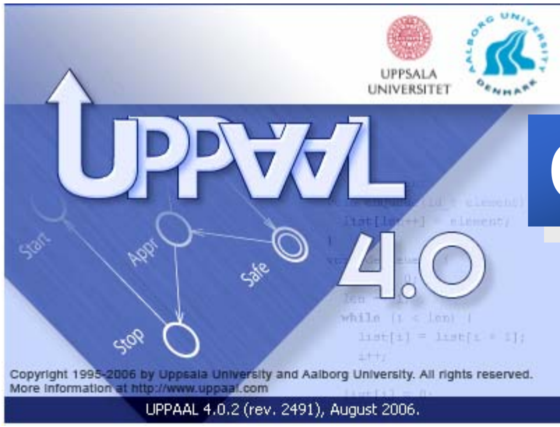


CORA

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

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

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

BRICS
Basic Research
in Computer Science

AMETIST

advanced methods for timed systems

OBJECTIVES

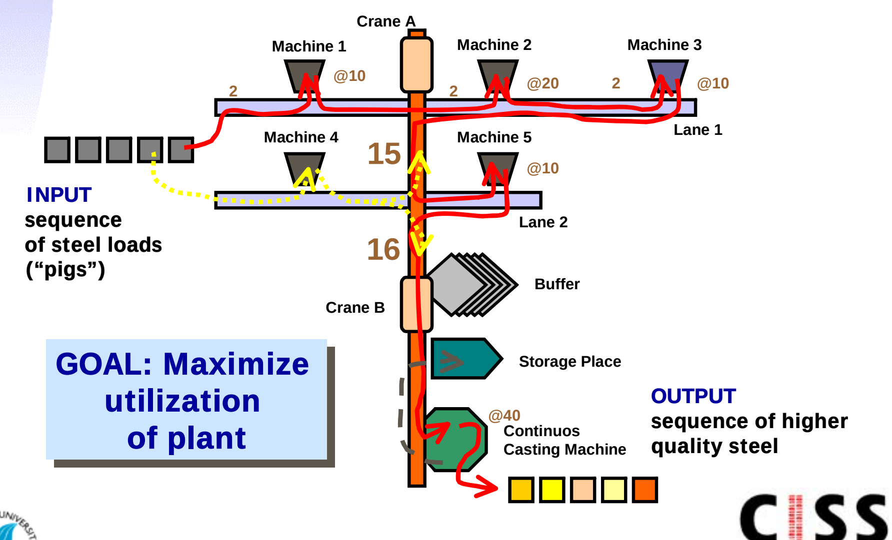
- powerful, unifying mathematical modelling
- efficient computerized problem-solving tools

- distributed real-time systems
- time-dependent behaviour and dynamic resource allocation

- **TIMED AUTOMATA**

BRICS Basic Research in Computer Science

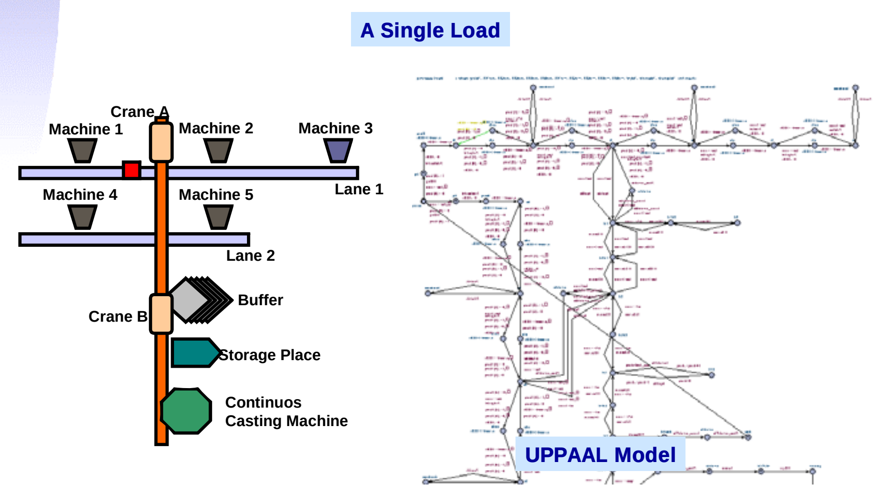
SIDMAR Overview



CISS

LTR Project VHS (Verification of Hybrid systems)

SIDMAR Modelling



CISS

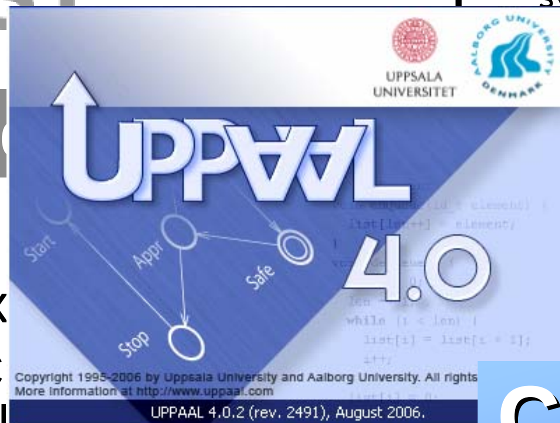
AMETIST

advanced methods for timed

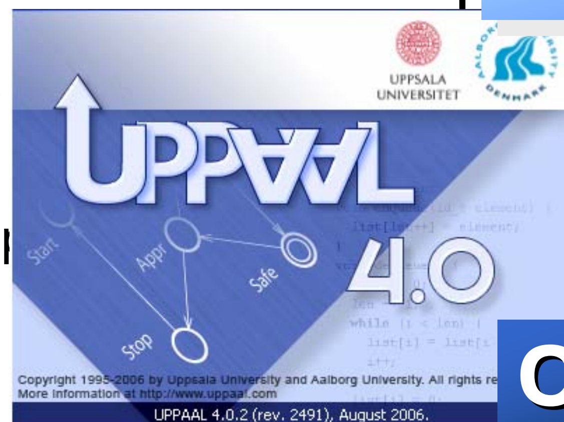
Case Stu

- Cybernetix
  - Smart C
  Personalization
- Terma:
  - Memory
- Bosch:
  - Car Peri
  Sensing
- AXXOM:
  - Lacquer Production
- Benchmarks

BRICS
Basic Research
in Computer Science

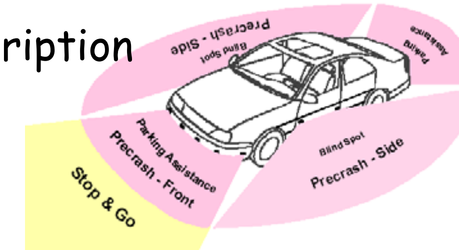


CLASSIC



CORA

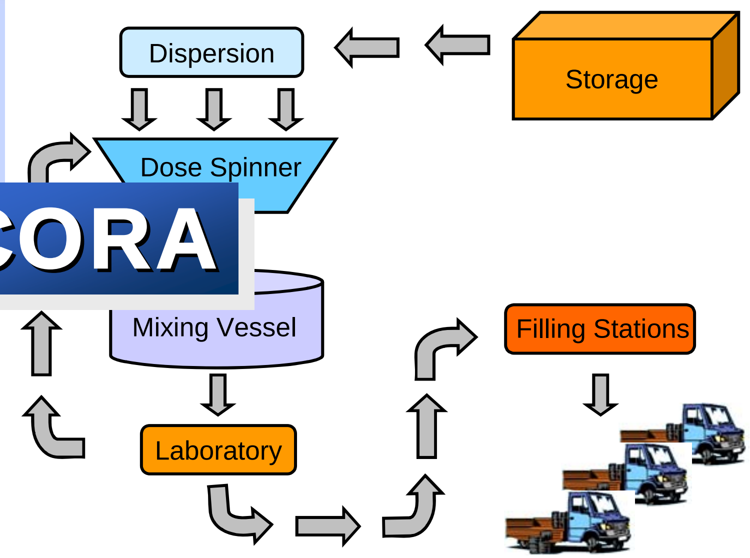
CPS: Informal description



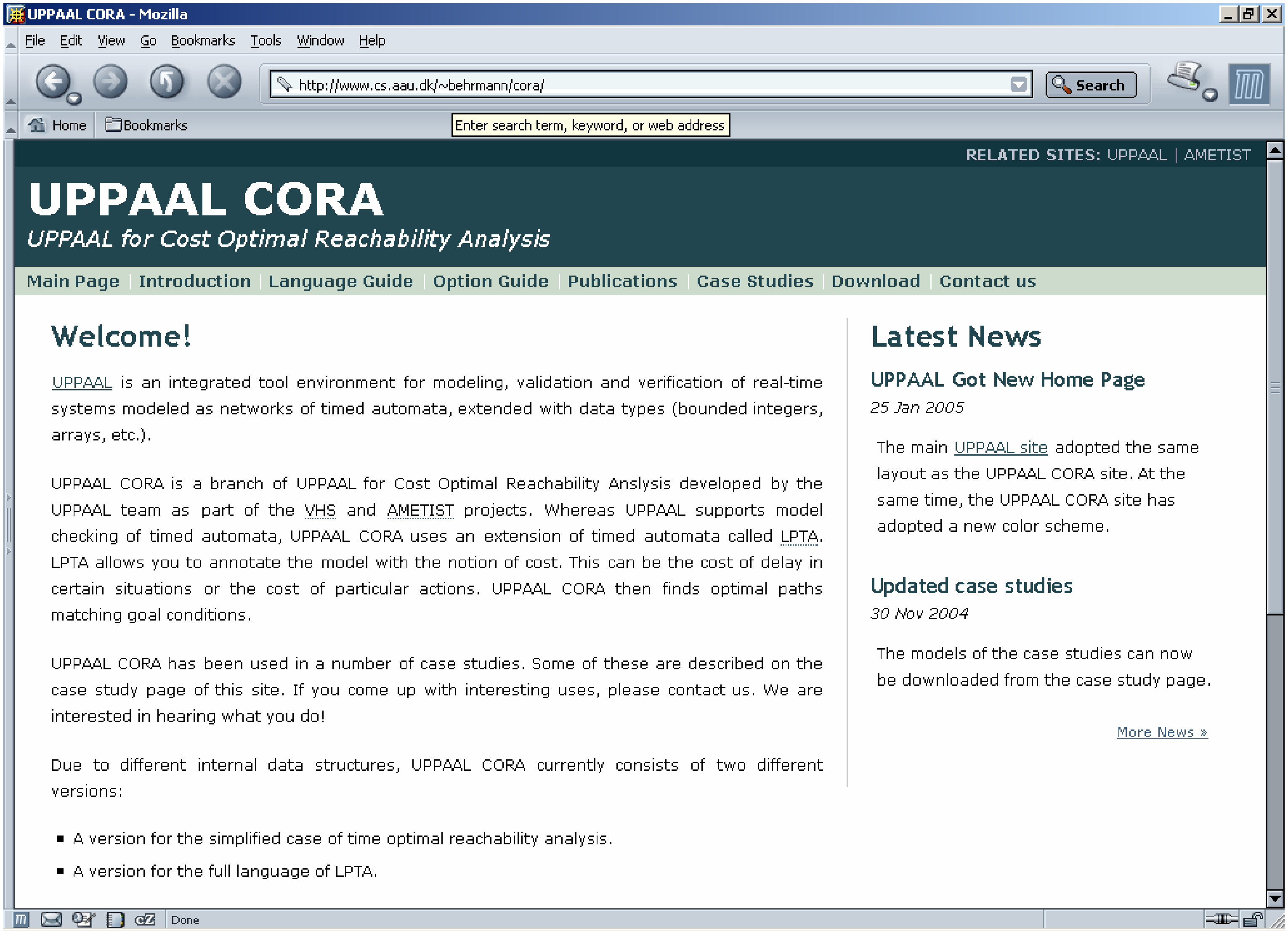
- CPS obtains and makes available for other systems information about ...vironment of a car. This ...formation may be used ...r:
  - Parking assistance
  - Pre-crash detection
  - Blind spot supervision
  - Lane change assistance
  - Stop & go
  - Etc
- The CPS considered in this case study
  - One sensor group only (currently 2 sensors)
  - Only the front sensors and corresponding controllers
  - Application: pre-crash detection, parking assistance, stop & go

Product flow of a Product

axxom



CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER



UPPAAL CORA - Mozilla

File Edit View Go Bookmarks Tools Window Help

http://www.cs.aau.dk/~behrmann/cora/ Search

Home Bookmarks Enter search term, keyword, or web address

RELATED SITES: UPPAAL | AMETIST

UPPAAL CORA

UPPAAL for Cost Optimal Reachability Analysis

Main Page | Introduction | Language Guide | Option Guide | Publications | Case Studies | Download | Contact us

Welcome!

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.).

UPPAAL CORA is a branch of UPPAAL for Cost Optimal Reachability Anslysis developed by the UPPAAL team as part of the VHS and AMETIST projects. Whereas UPPAAL supports model checking of timed automata, UPPAAL CORA uses an extension of timed automata called LPTA. LPTA allows you to annotate the model with the notion of cost. This can be the cost of delay in certain situations or the cost of particular actions. UPPAAL CORA then finds optimal paths matching goal conditions.

UPPAAL CORA has been used in a number of case studies. Some of these are described on the case study page of this site. If you come up with interesting uses, please contact us. We are interested in hearing what you do!

Due to different internal data structures, UPPAAL CORA currently consists of two different versions:

- A version for the simplified case of time optimal reachability analysis.
- A version for the full language of LPTA.

Latest News

UPPAAL Got New Home Page

25 Jan 2005

The main UPPAAL site adopted the same layout as the UPPAAL CORA site. At the same time, the UPPAAL CORA site has adopted a new color scheme.

Updated case studies

30 Nov 2004

The models of the case studies can now be downloaded from the case study page.

More News »

Done

BRICS
Basic Research
in Computer Science

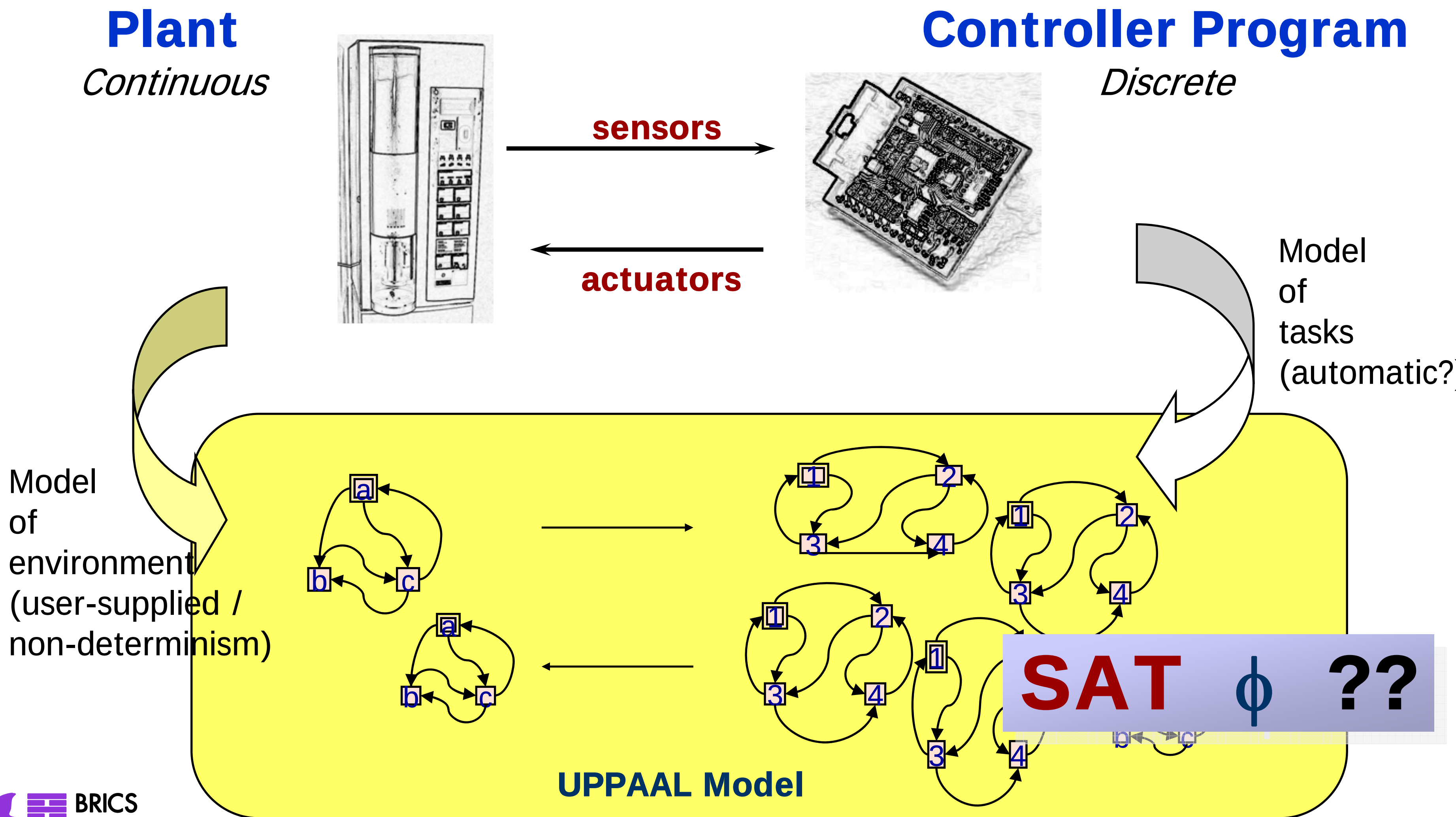
Overview

- Timed Automata & Scheduling
- Priced Timed Automata and Optimal Scheduling
- Optimal Infinite Scheduling
- Optimal Conditional Scheduling
- Applications

Optimal Scheduling Using Priced Timed Automata.
G. Behrmann, K. G. Larsen, J. I. Rasmussen,
ACM SIGMETRICS Performance Evaluation Review

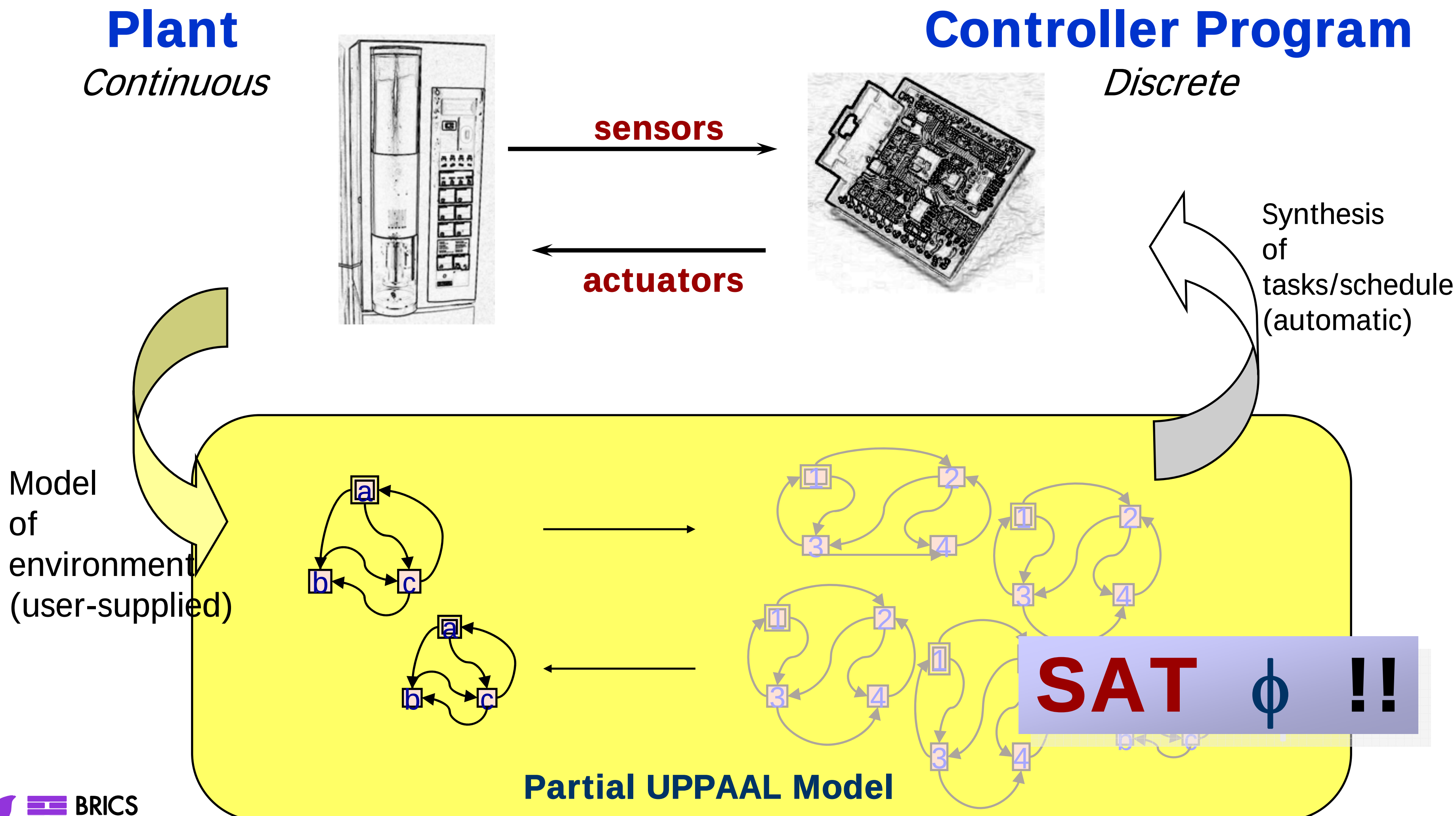
CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Real Time Model Checking



BRICS
Basic Research in Computer Science

Real Time Scheduling & Control Synthesis

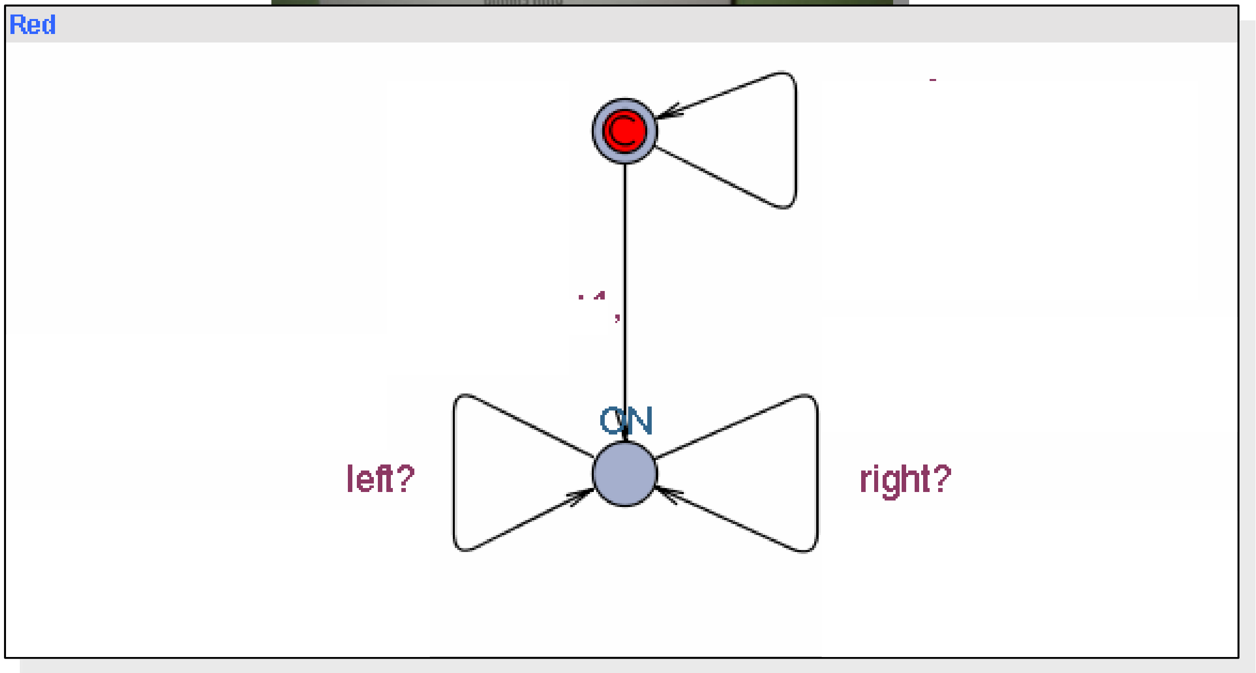


BRICS Basic Research in Computer Science

Rush Hour

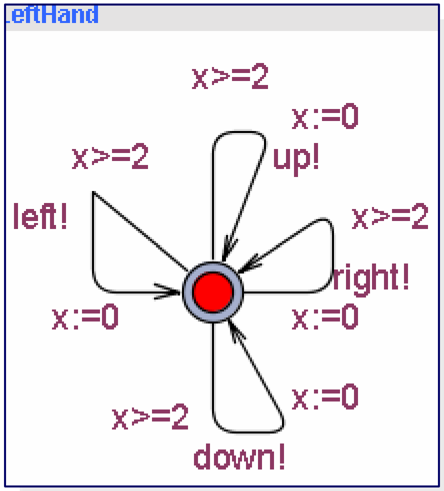
OBJECTIVE:
Get your CAR out

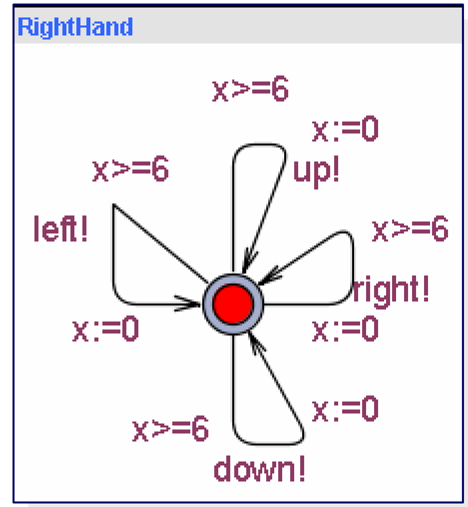
Rush Hour

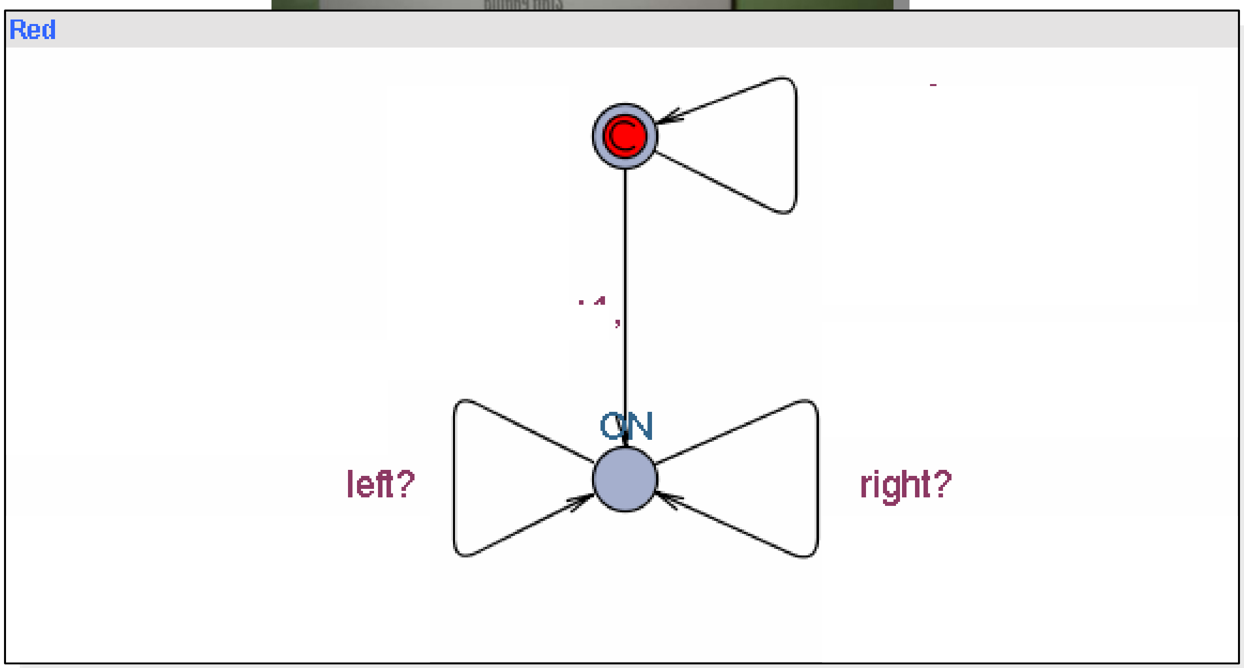


CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Rush Hour





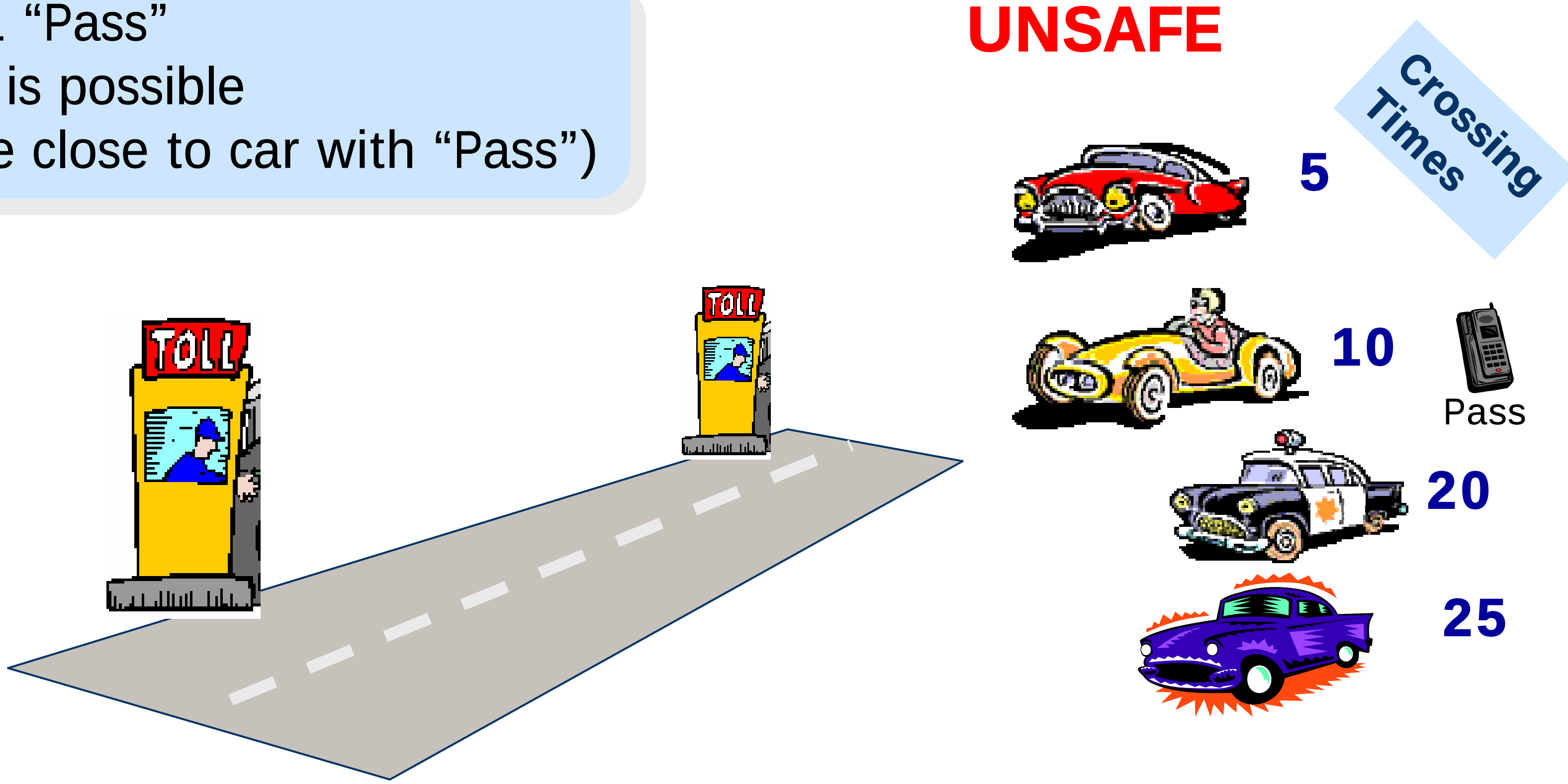


BRICS
Basic Research
in Computer Science

Real Time Scheduling

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

UNSAFE



SAFE

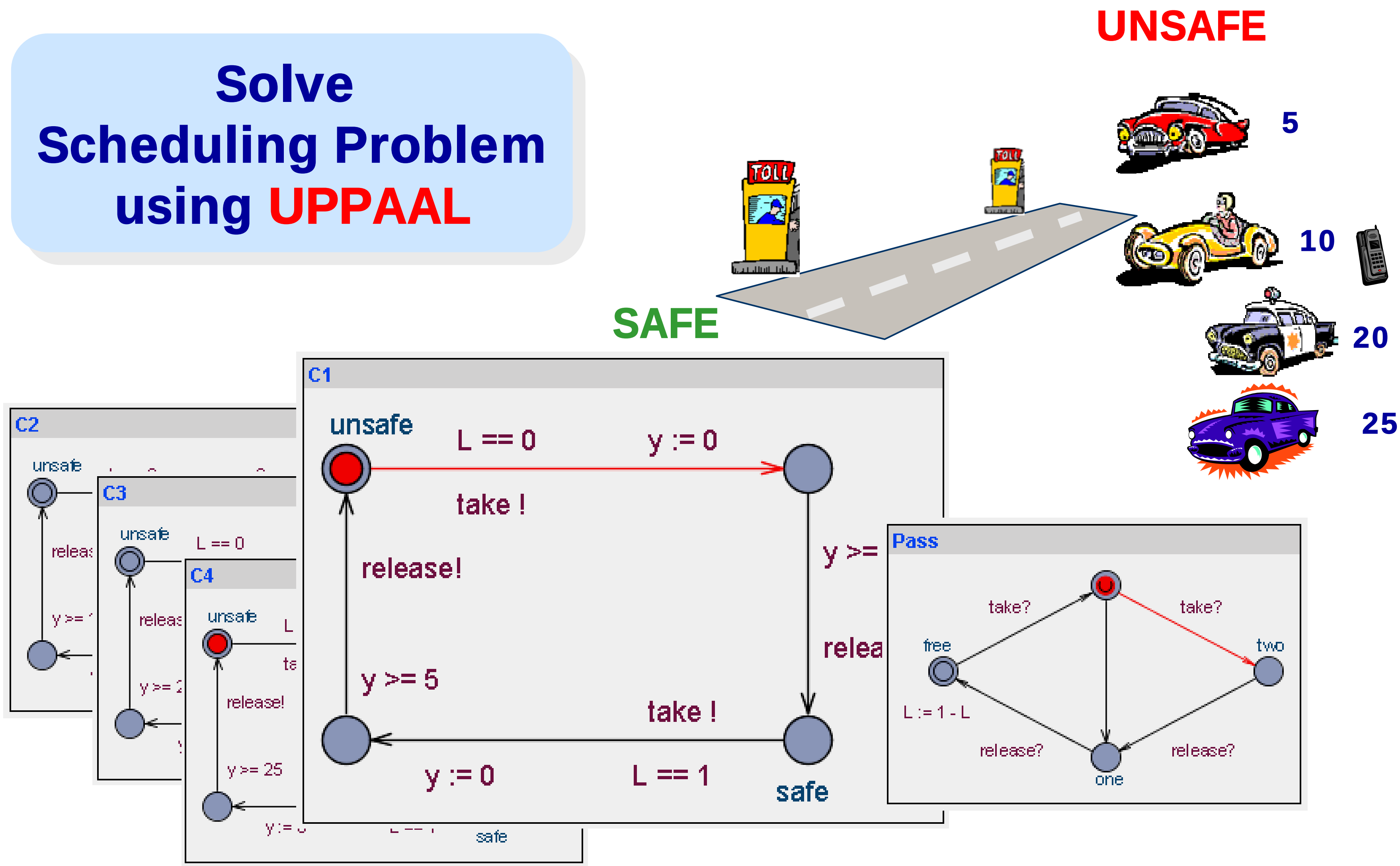
CAN THEY MAKE IT TO SAFE WITHIN 70 MINUTES ???

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Real Time Scheduling

Solve Scheduling Problem using UPPAAL

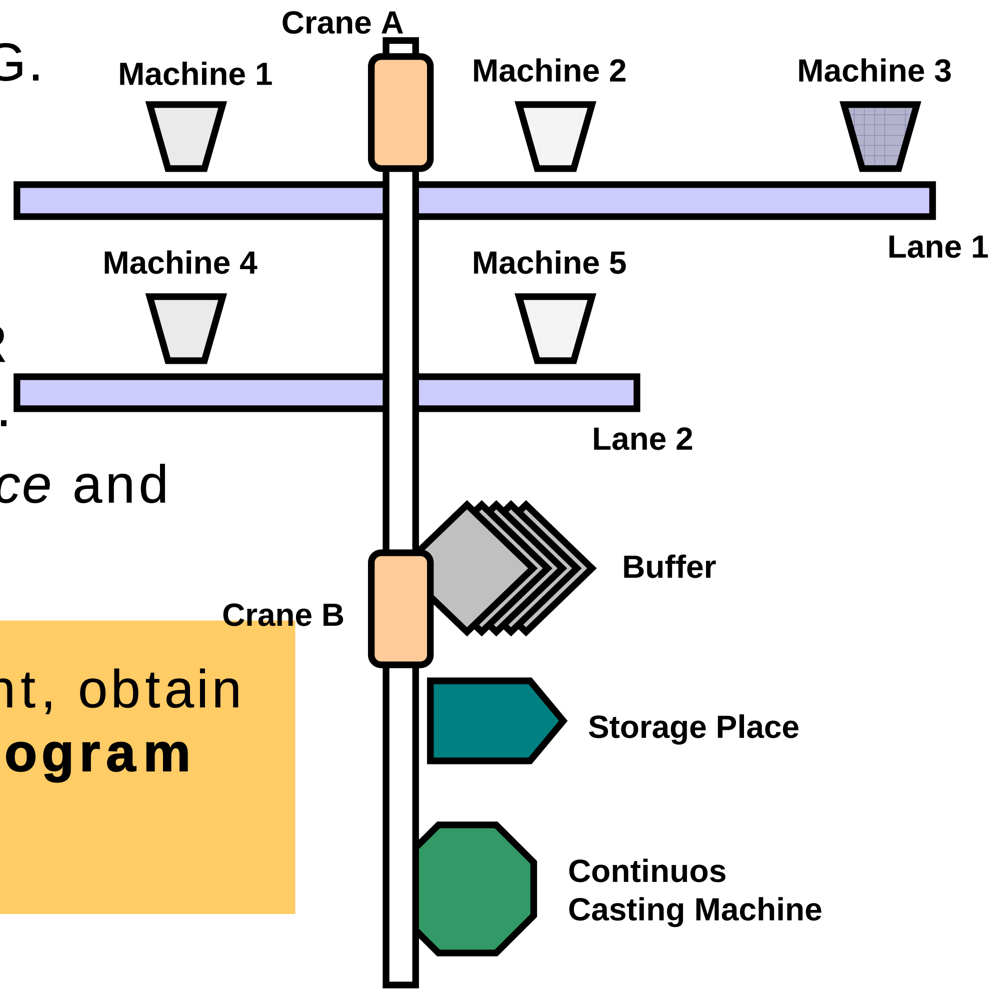


BRICS
Basic Research
in Computer Science

Steel Production Plant

- A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson
- Case study of Esprit-LTR project 26270 VHS
- Physical plant of SIDMAR located in Gent, Belgium.
- Part between *blast furnace* and *hot rolling mill*.

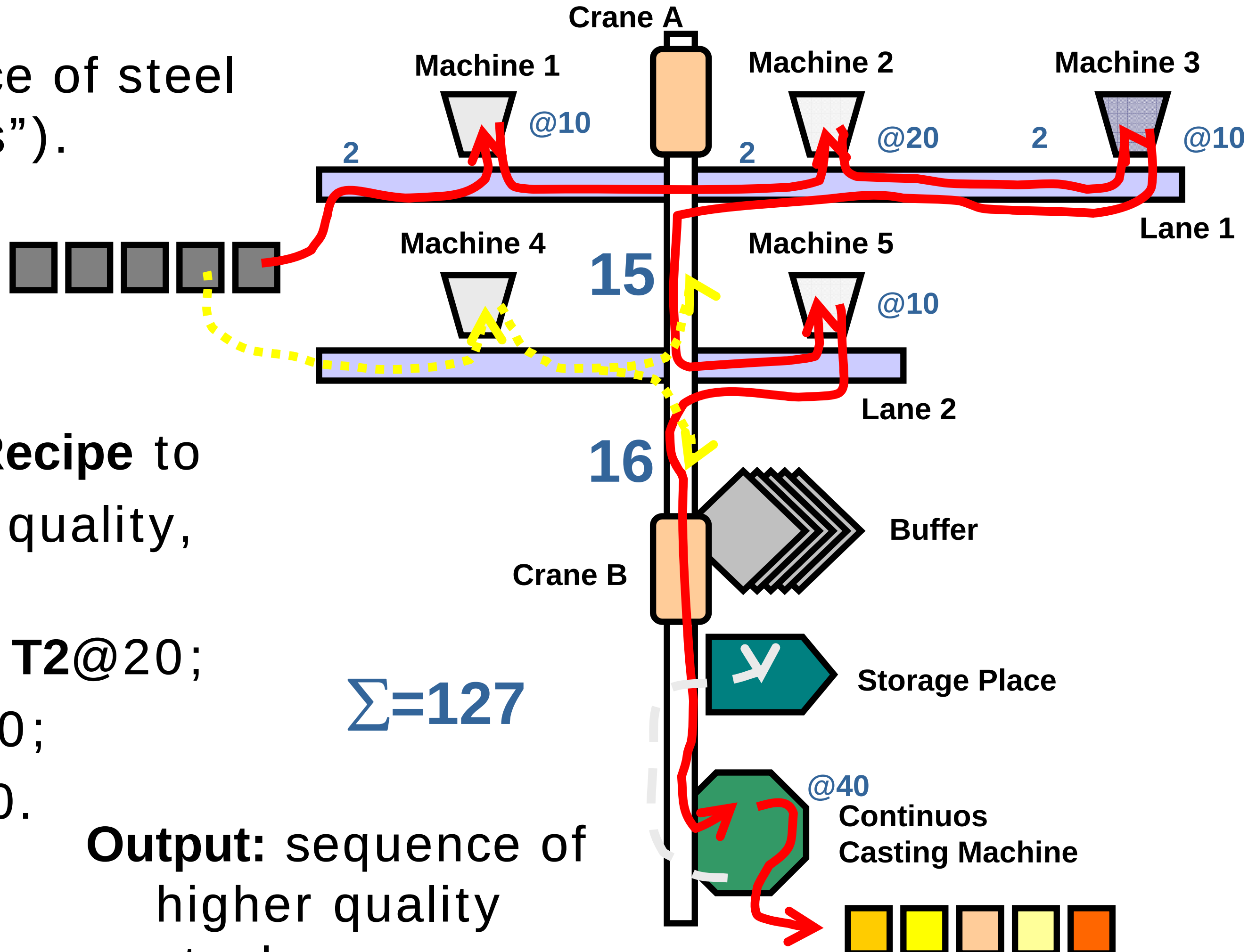
Objective: model the plant, obtain **schedule** and control **program** for plant.



BRICS Basic Research in Computer Science

Steel Production Plant

Input: sequence of steel loads (“pigs”).



Load follows **Recipe** to obtain certain quality, e.g:

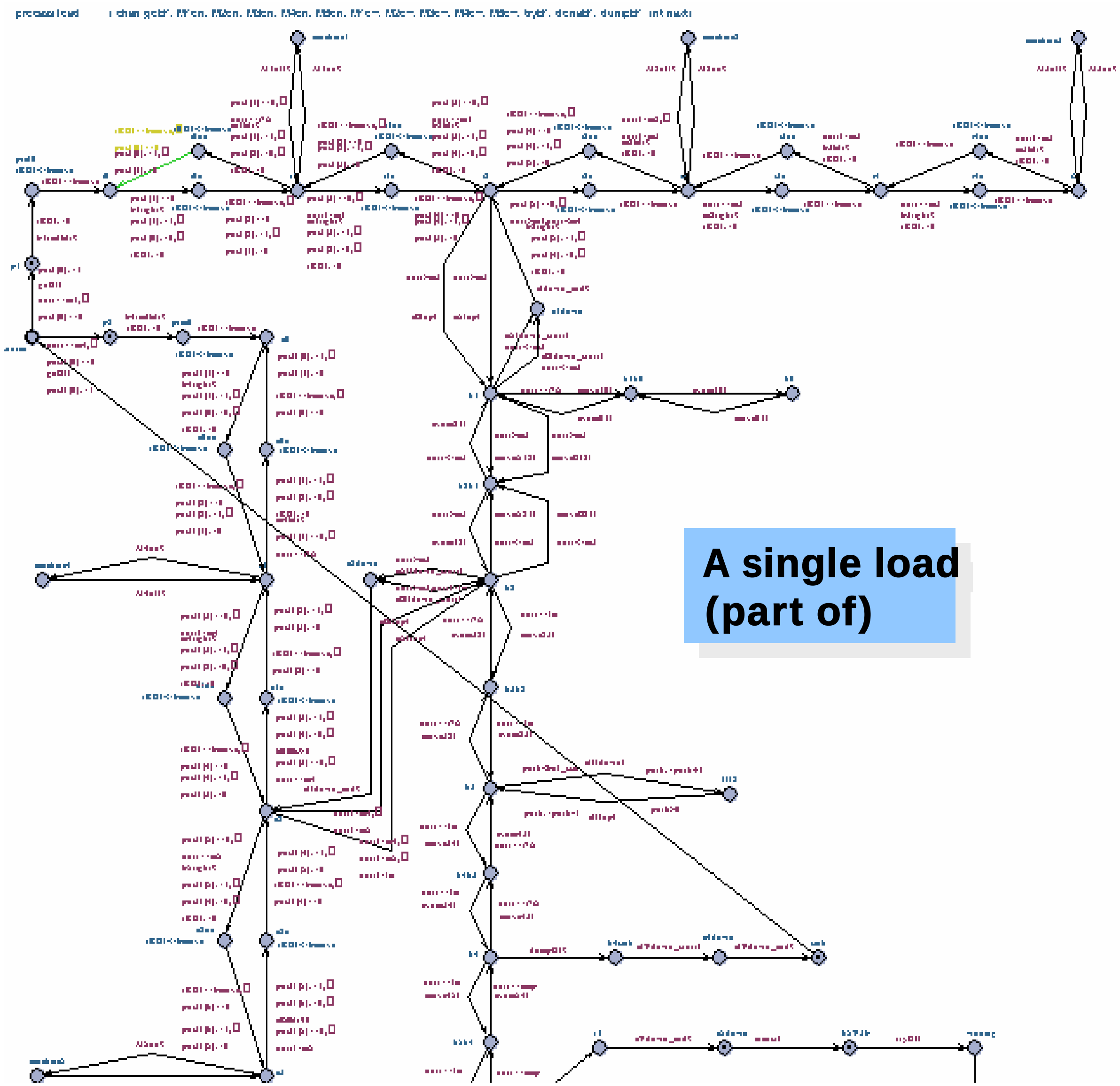
start; **T1**@10; **T2**@20; **T3**@10; **T2**@10;

end within 120.

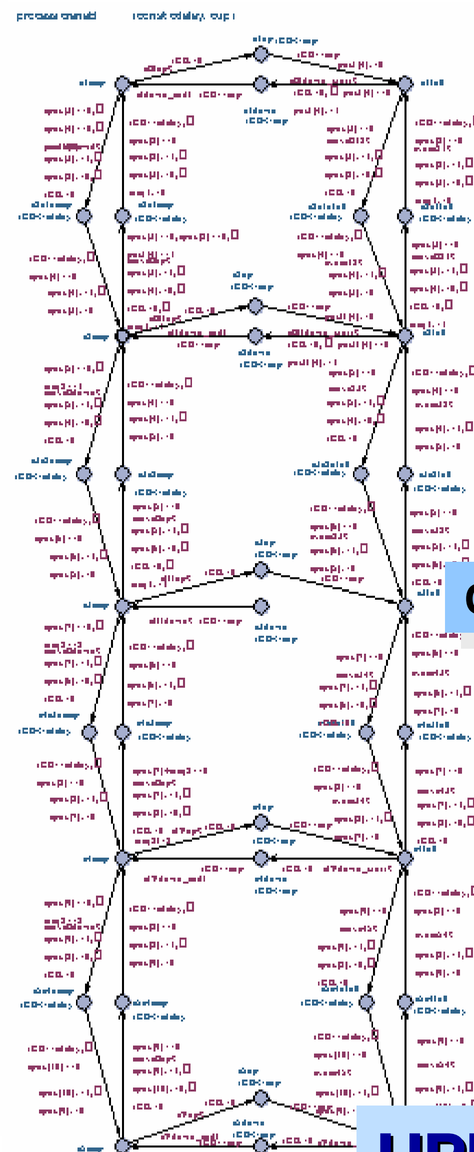
$\sum$=127

Output: sequence of higher quality steel.

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER



A single load (part of)



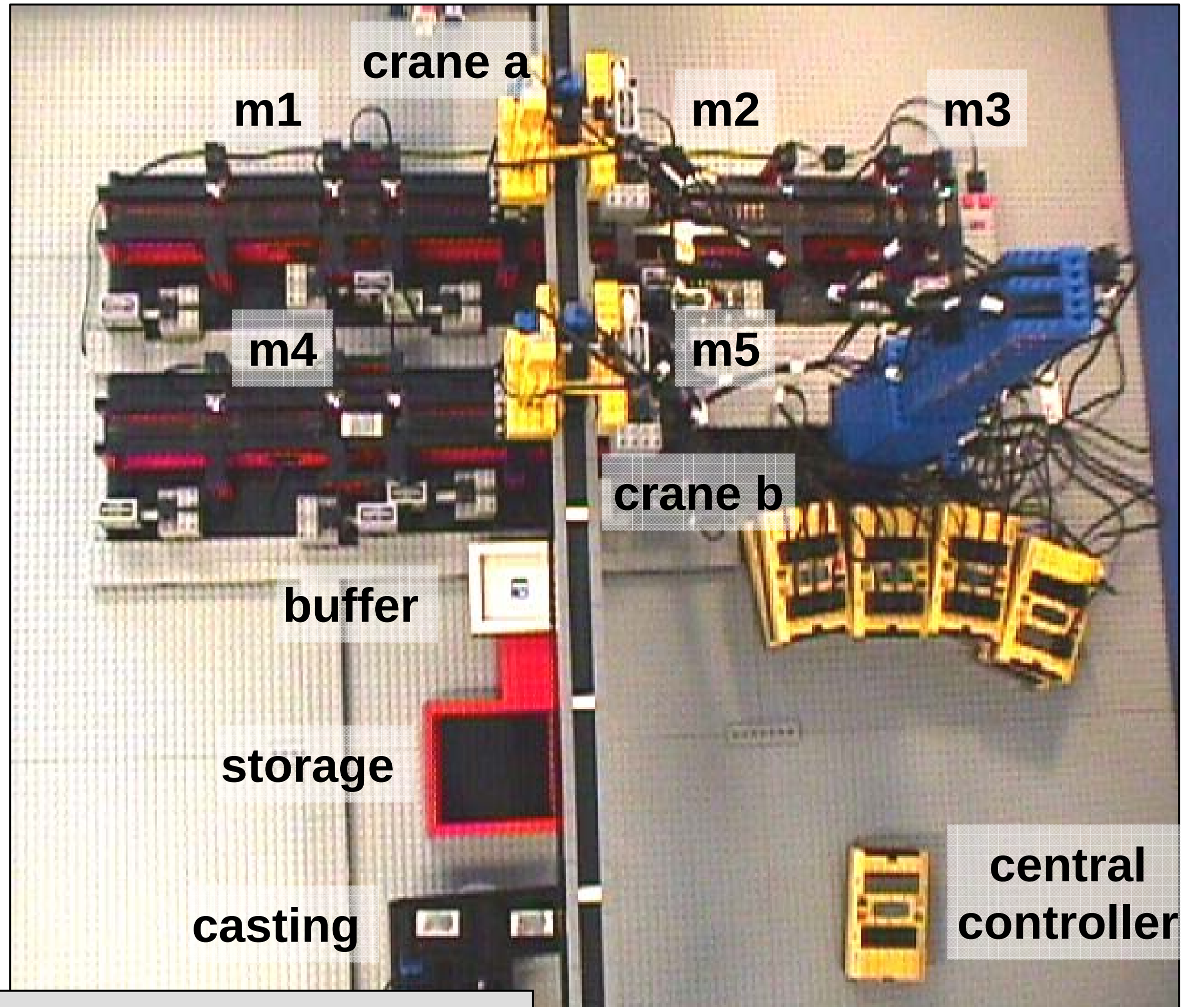
Crane B

UPPAAL

BRICS
Basic Research
in Computer Science

Controller Synthesis for LEGO Model

- LEGO RCX Mindstorms.
- Local controllers with control programs.
- IR protocol for remote invocation of programs.
- Central controller.



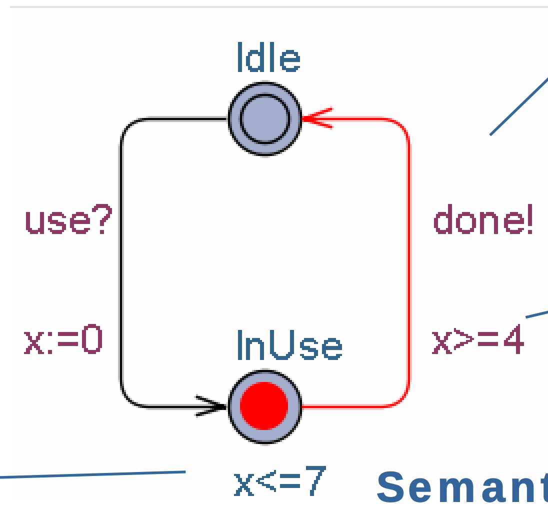
1971 lines of RCX code (n=5),
24860 - “ - (n=60).

Synthesis

BRICS
Basic Research in Computer Science

Timed Automata [Alur & Dill'89]

Resource

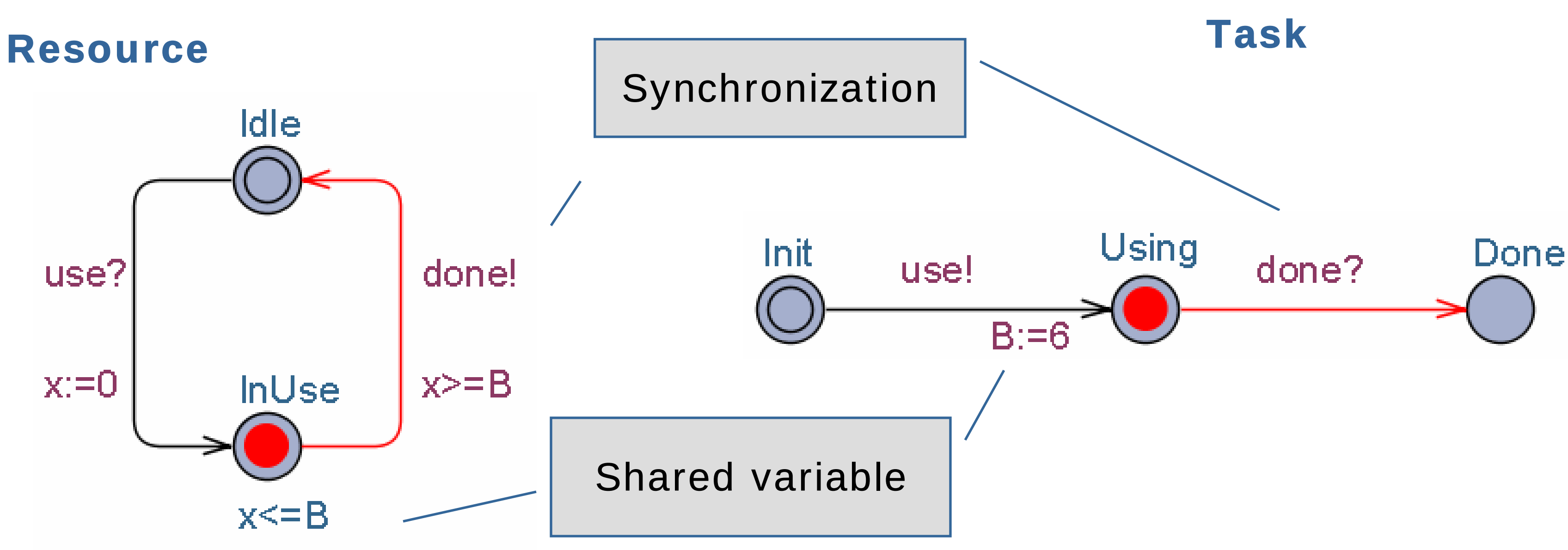


Semantics:

(Idle , x=0)

→ (Idle , x=2.5)	d(2.5)
→ (InUse , x=0)	use?
→ (InUse , x=5)	d(5)
→ (Idle , x=5)	done!
→ (Idle , x=8)	d(3)
→ (InUse , x=0)	use?

Composition



Semantics:

(Idle , Init , B=0, x=0)

→ (Idle , Init , B=0 , x=3.1415) d(3)

→ (InUse , Using , B=6, x=0) use

→ (InUse , Using , B=6, x=6) d(6)

→ (Idle , Done , B=6 , x=6) done

Jobshop Scheduling

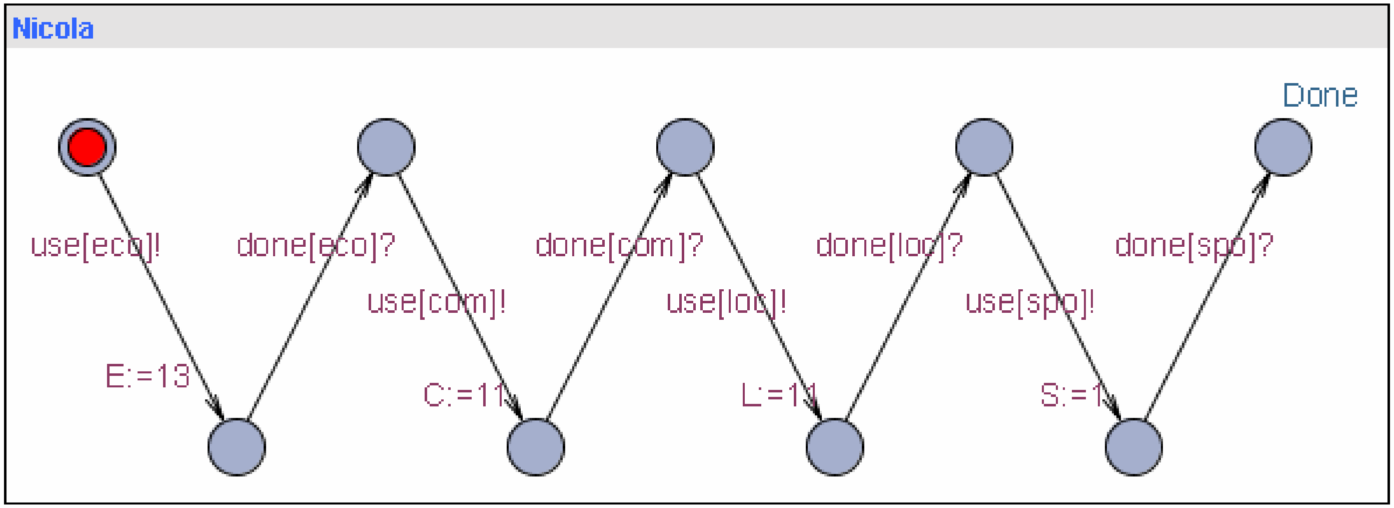
RE SOURCES

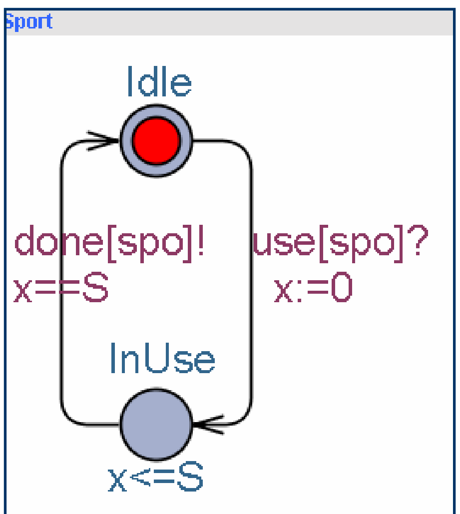
JOBS

	Sport	Economy	Local News	Comic Stip
Kim	2. 5 min	4. 1 min	3. 3 min	1. 10 min
Maria	1. 10 min	2. 20 min	3. 1 min	4. 1 min
Nicola	4. 1 min	1. 13 min	3. 11 min	2. 11 min

Problem: compute the minimal **MAKESPAN**

Jobshop Scheduling in UPPAAL





	Sport	Economy	Local News	Comic Stip
Kim	2. 5 min	4. 1 min	3. 3 min	1. 10 min
Maria	1. 10 min	2. 20 min	3. 1 min	4. 1 min
Nicola	4. 1 min	1. 13 min	3. 11 min	2. 11 min

BRICS
Basic Research
in Computer Science

Experiments

[TACAS'2001]

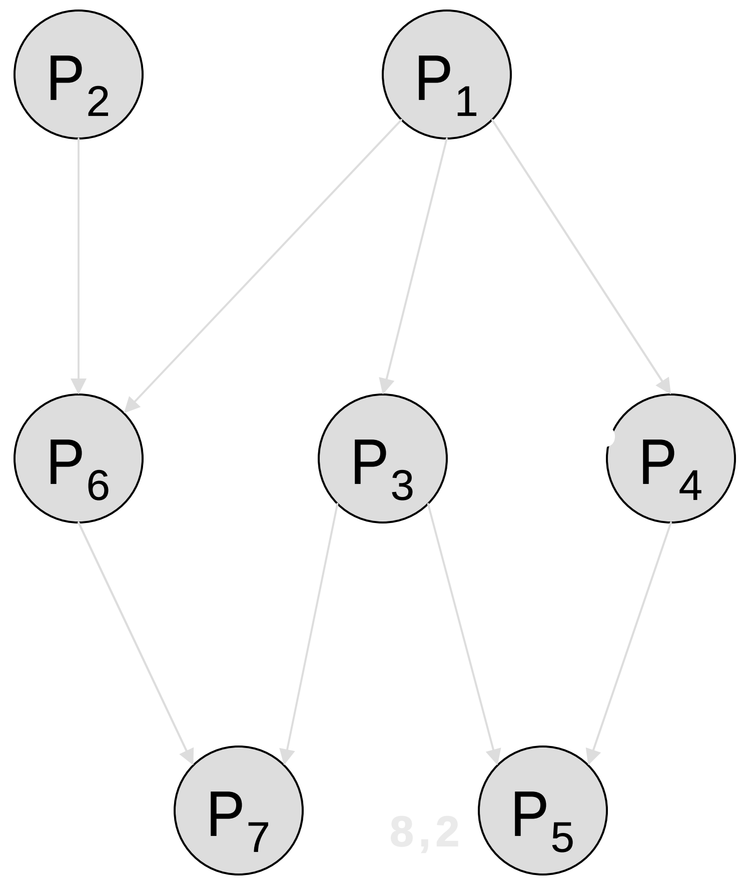
B-&-B algorithm running for 60 sec.

m	j	problem instance	BF cost	BF states	MC cost	MC states	MC+ cost	MC+ states	DF cost	DF states	RDF cost	RDF states	comb. heur. cost	comb. heur. states	minimal makespan
m=5	j=10	la01	-	-	-	-	-	-	2466	-	842	-	666	292	666
		la02	-	-	-	-	-	-	2360	-	806	-	672	-	655
		la03	-	-	-	-	-	-	2094	-	769	-	626	-	597
		la04	-	-	-	-	-	-	2212	-	783	-	639	-	590
		la05	-	-	-	-	593	9791	1955	-	696	-	593	284	593
	j=15	la06	-	-	-	-	-	-	3656	-	1076	-	926	480	926
		la07	-	-	-	-	-	-	3410	-	1113	-	890	-	890
		la08	-	-	-	-	-	-	3520	-	1009	-	863	400	863
		la09	-	-	-	-	-	-	3984	-	1154	-	951	425	951
		la10	-	-	-	-	-	-	3681	-	1063	-	958	454	958
	j=20	la11	-	-	-	-	-	-	4974	-	1303	-	1222	642	1222
		la12	-	-	-	-	-	-	4557	-	1271	-	1039	633	1039
		la13	-	-	-	-	-	-	4846	-	1227	-	1150	662	1150
		la14	-	-	-	-	1292	10653	5145	-	1377	-	1292	688	1292
		la15	-	-	-	-	-	-	5264	-	1459	-	1289	-	1207
m=10	j=10	la16	-	-	-	-	-	-	4849	-	1298	-	1022	-	945
		la17	-	-	-	-	-	-	4299	-	938	-	786	-	784
		la18	-	-	-	-	-	-	4763	-	1034	-	922	-	848
		la19	-	-	-	-	-	-	4566	-	1140	-	904	-	842
		la20	-	-	-	-	-	-	5056	-	1378	-	964	-	902
	j=15	la21	-	-	-	-	-	-	7608	-	1326	-	1149	-	(1040,1053)
		la22	-	-	-	-	-	-	6920	-	1413	-	1047	-	927
		la23							7676	-	1357	-	1075	-	1032
		la24							7237	-	1346	-	1061	-	935
		la25	-	-	-	-	-	-	7141	-	1290	-	1070	-	977

Lawrence Job Shop Problems

Task Graph Scheduling

Optimal Static Task Scheduling

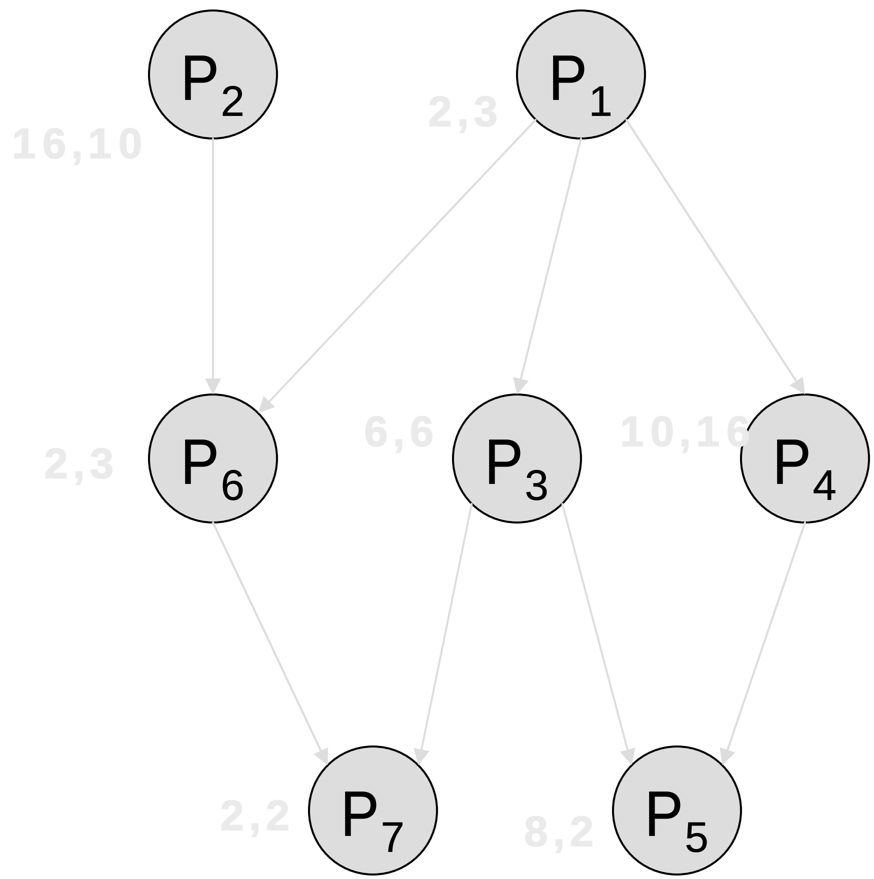


$\mathbf{M} = \{M_1, M_2\}$

Task Graph Scheduling

Optimal Static Task Scheduling

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



M = $\{M_1,M_2\}$

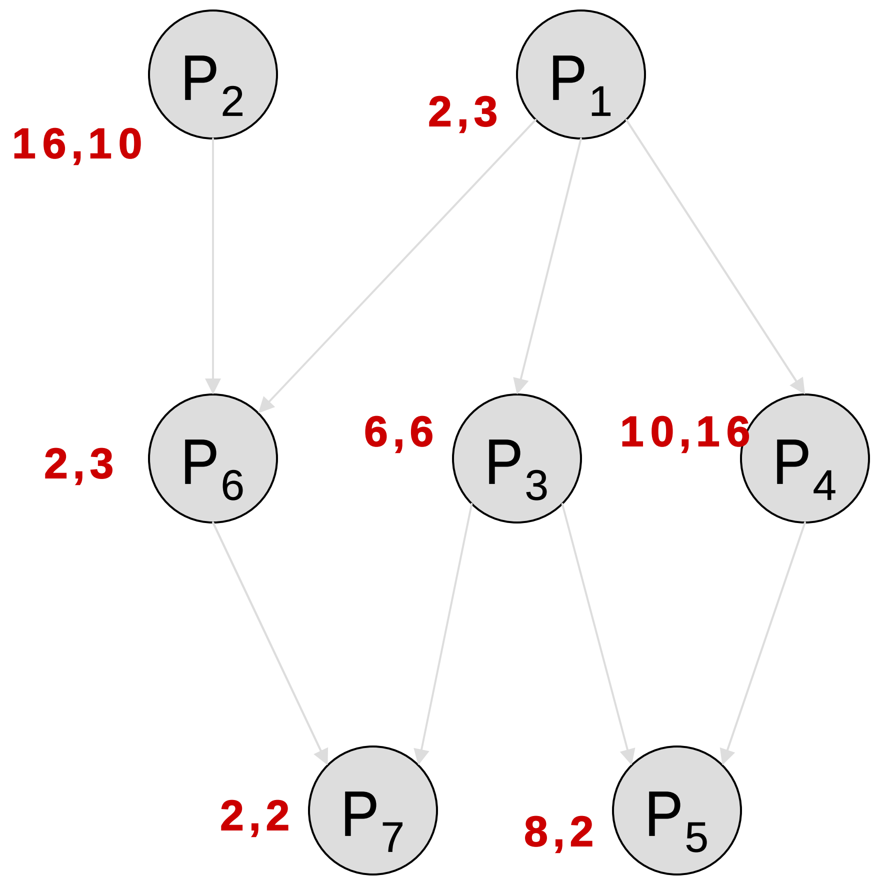
BRICS Basic Research in Computer Science

CISS CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Task Graph Scheduling

Optimal Static Task Scheduling

- Task $\mathbf{P}=\{P_1,..,P_m\}$
- Machines $\mathbf{M}=\{M_1,..,M_n\}$
- Duration $\Delta : (\mathbf{P}\times\mathbf{M}) \rightarrow \mathbf{N}_\infty$

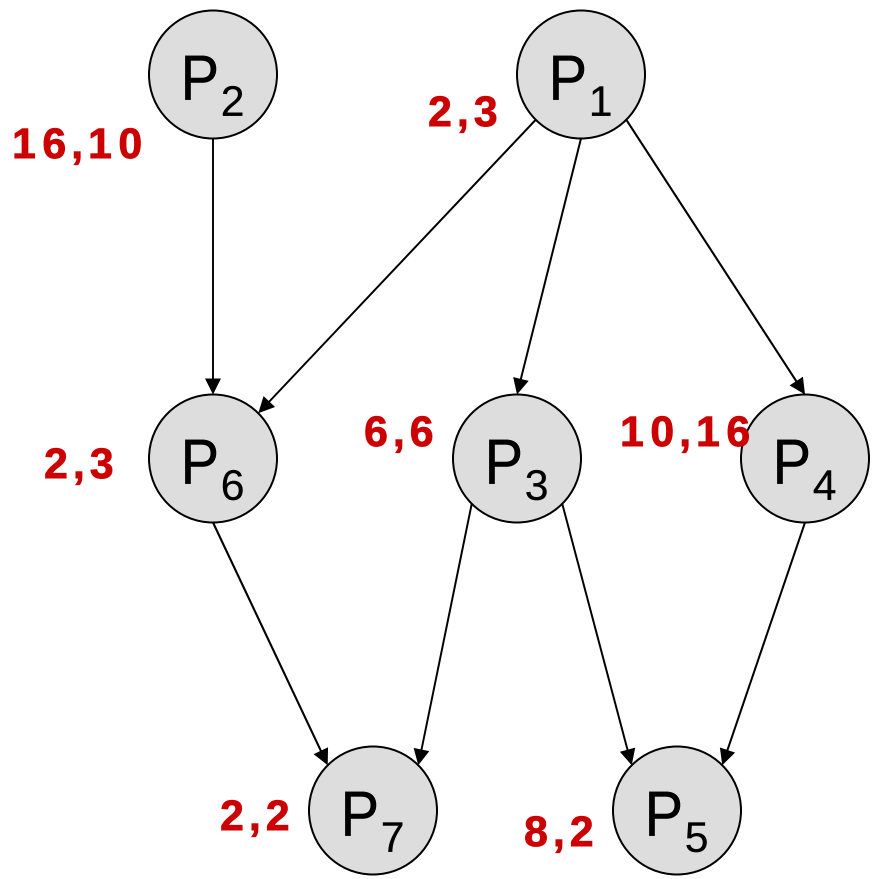


$\mathbf{M} = \{M_1,M_2\}$

Task Graph Scheduling

Optimal Static Task Scheduling

- Task $\mathbf{P} = \{P_1,.., P_m\}$
- Machines $\mathbf{M} = \{M_1,..,M_n\}$
- Duration $\Delta : (\mathbf{P} \times \mathbf{M}) \rightarrow \mathbf{N}_\infty$
- $<$: p.o. on $\mathbf{P}$ (pred.)
-
-
-

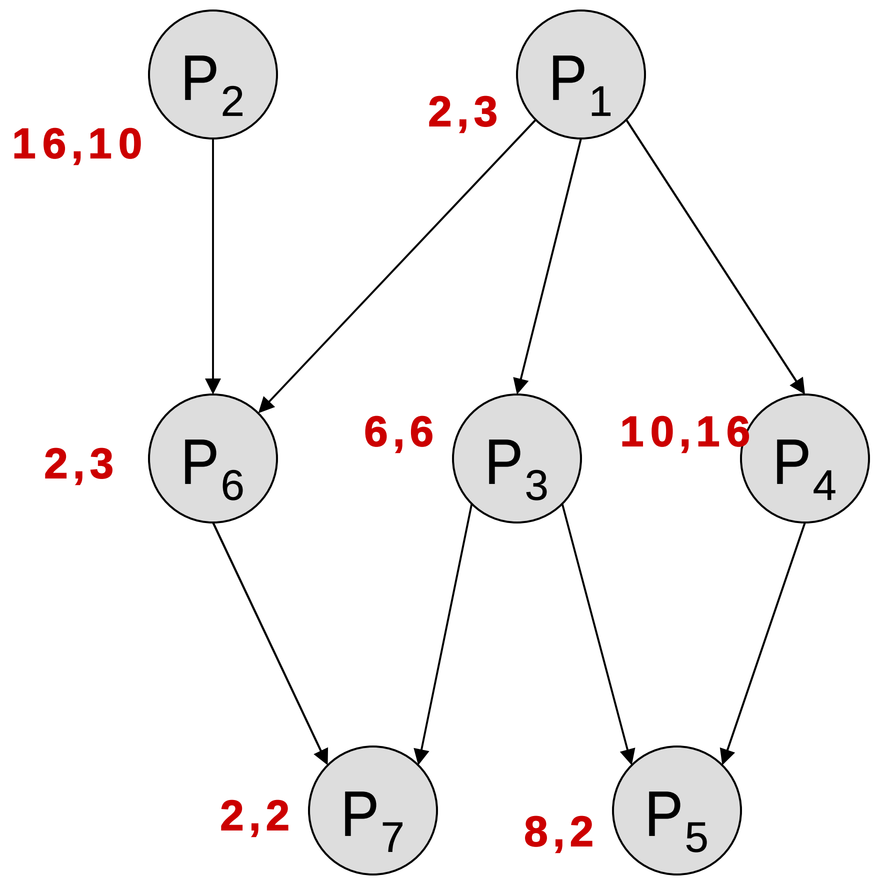


$\mathbf{M} = \{M_1, M_2\}$

Task Graph Scheduling

Optimal Static Task Scheduling

- Task $\mathbf{P} = \{P_1, .., P_m\}$
- Machines $\mathbf{M} = \{M_1, .., M_n\}$
- Duration $\Delta : (\mathbf{P} \times \mathbf{M}) \rightarrow \mathbf{N}_\infty$
- $<$: p.o. on $\mathbf{P}$ (pred.)



- A task can be executed only if all predecessors have completed
- Each machine can process at most one task at a time
- Task cannot be preempted.

$\mathbf{M} = \{M_1, M_2\}$

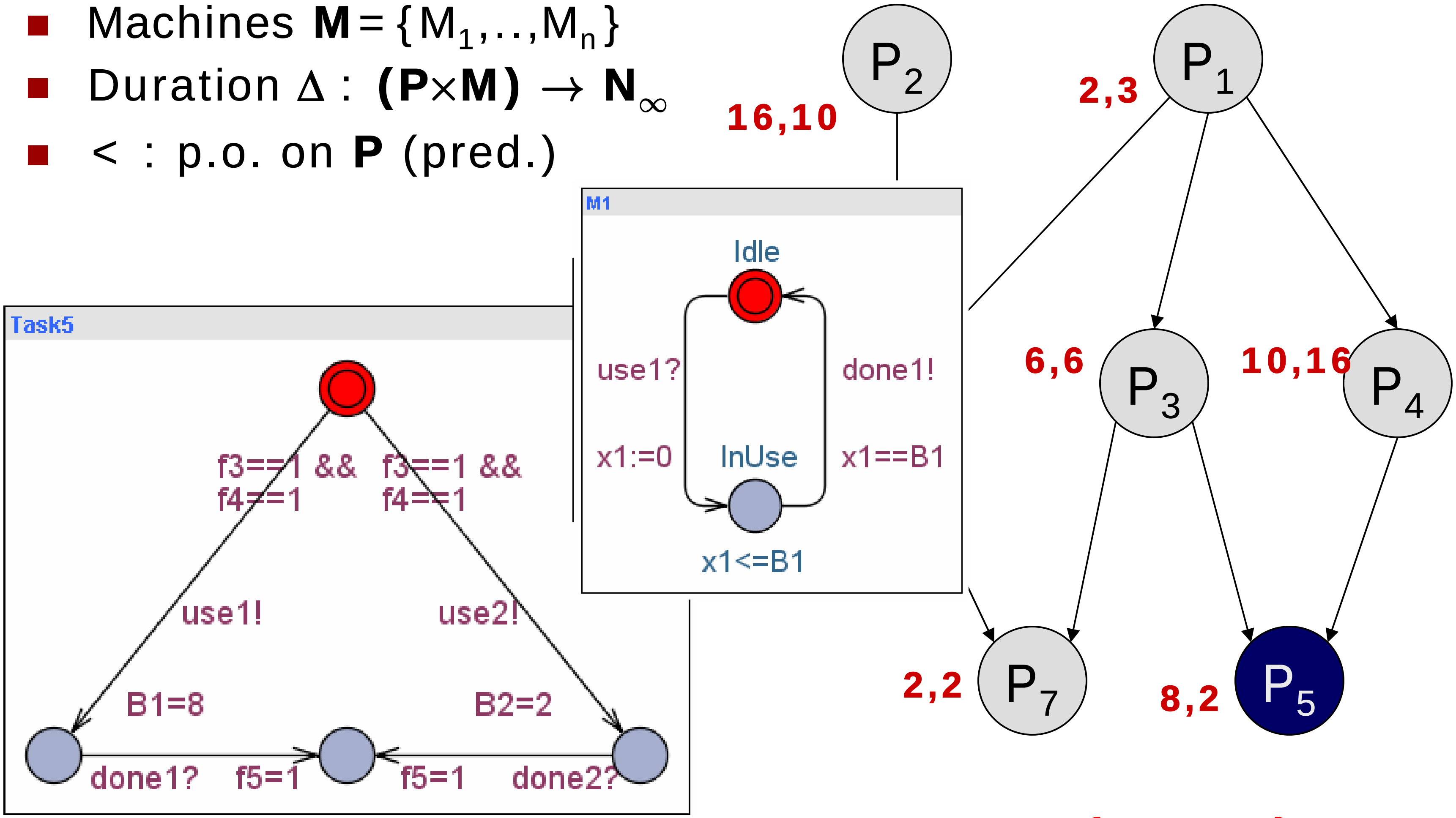
BRICS Basic Research in Computer Science

CISS CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Task Graph Scheduling

Optimal Static Task Scheduling

- Task $\mathbf{P} = \{P_1, .., P_m\}$
- Machines $\mathbf{M} = \{M_1, .., M_n\}$
- Duration $\Delta : (\mathbf{P} \times \mathbf{M}) \rightarrow \mathbf{N}_\infty$
- $<$: p.o. on $\mathbf{P}$ (pred.)



$\mathbf{M} = \{M_1, M_2\}$

Experimental Results

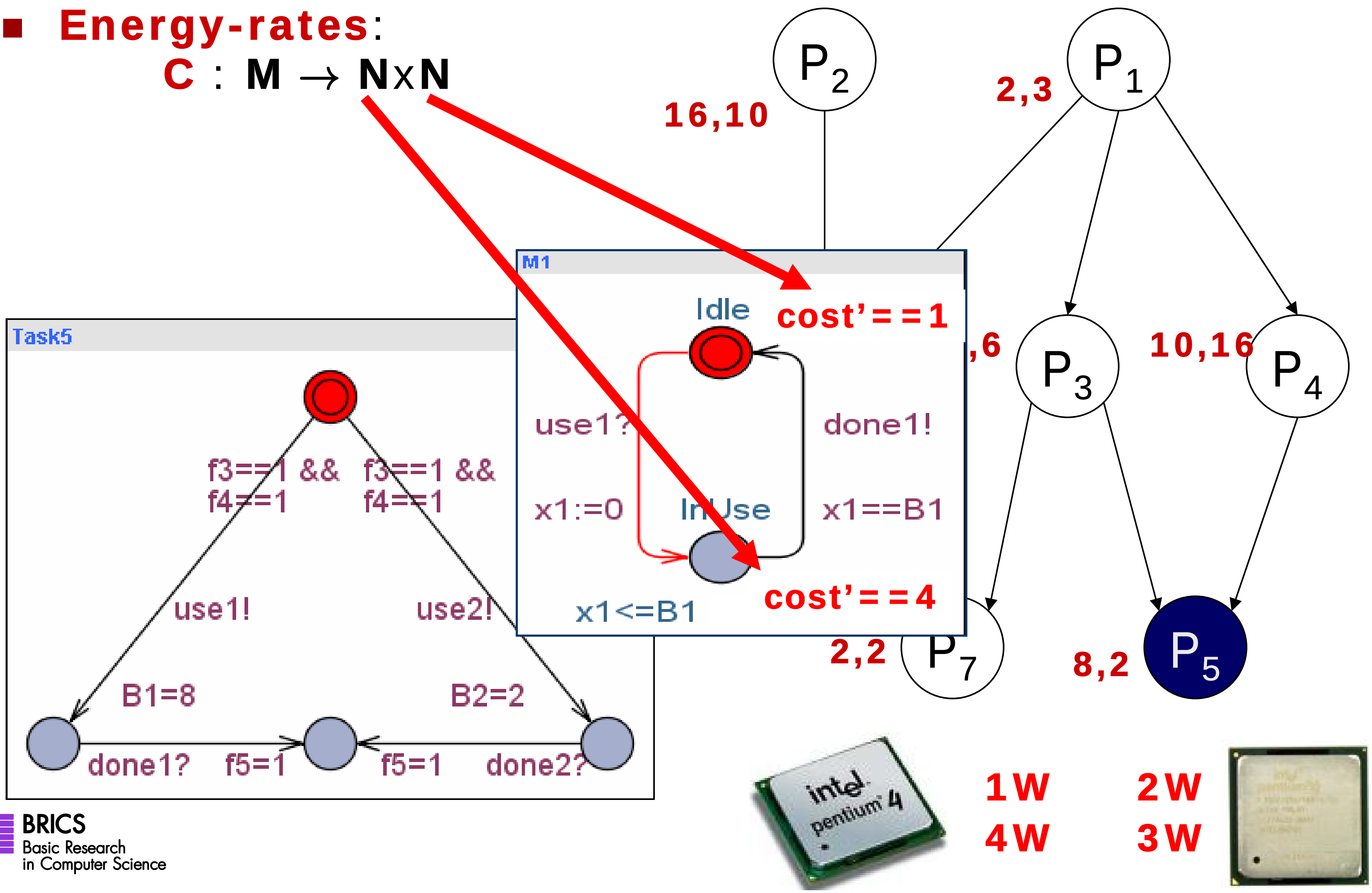
name	#tasks	#chains	# machines	optimal	TA
001	437	125	4	1178	1182
000	452	43	20	537	537
018	730	175	10	700	704
074	1007	66	12	891	894
021	1145	88	20	605	612
228	1187	293	8	1570	1574
071	1193	124	20	629	634
271	1348	127	12	1163	1164
237	1566	152	12	1340	1342
231	1664	101	16	t.o.	1137
235	1782	218	16	t.o.	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

Abdeddaïm, Kerbaa, Maler

Optimal Task Graph Scheduling

Power-Optimality

- **Energy-rates**:
  $C : M \rightarrow N \times N$

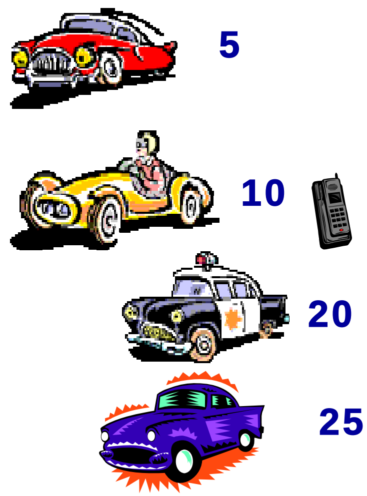


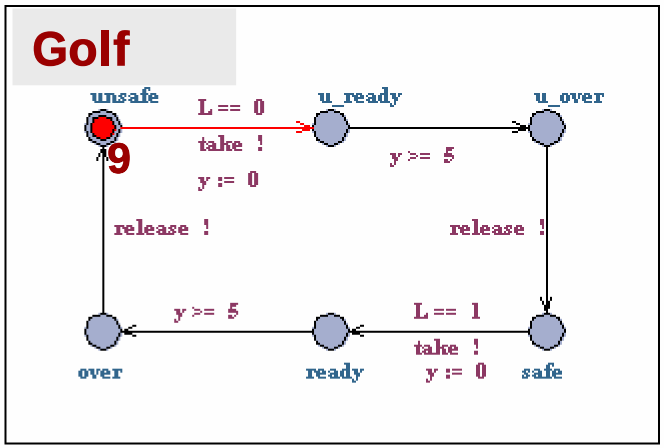
Priced Timed Automata

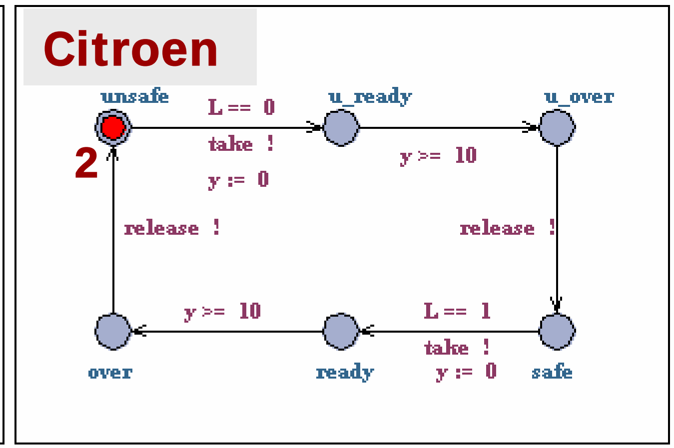
Optimal Scheduling

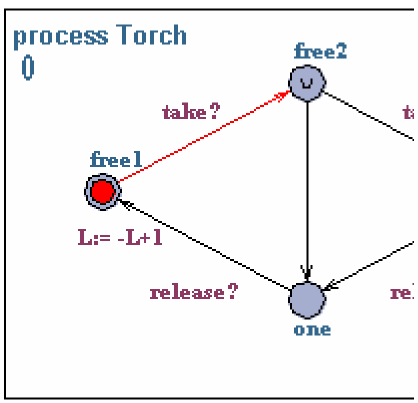
with Paul Pettersson, Thomas Hune, Judi Romijn,
Ansgar Fehnker, Ed Brinksma, Frits Vaandrager,
Patricia Bouyer, Franck Cassez, Henning Dierks
Emmanuel Fleury, Jacob Rasmussen,..

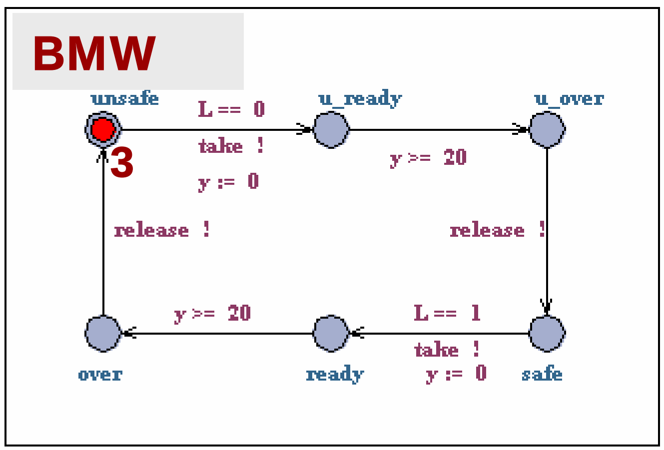
EXAMPLE: Optimal rescue plan for cars with different subscription rates for city driving !

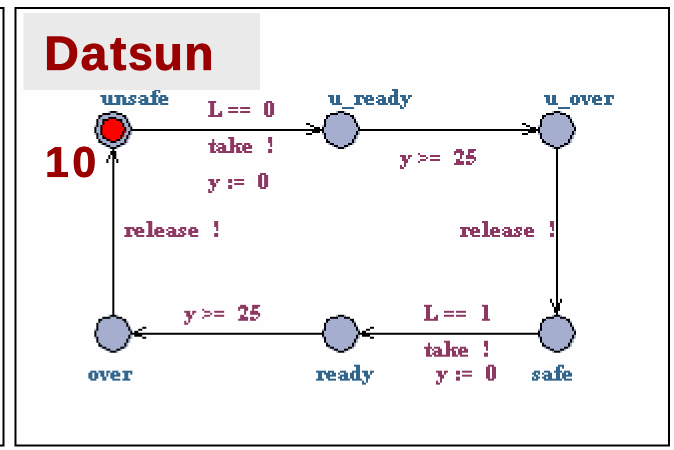












OPTIMAL PLAN HAS ACCUMULATED COST=195 and TOTAL TIME=65!

Experiments

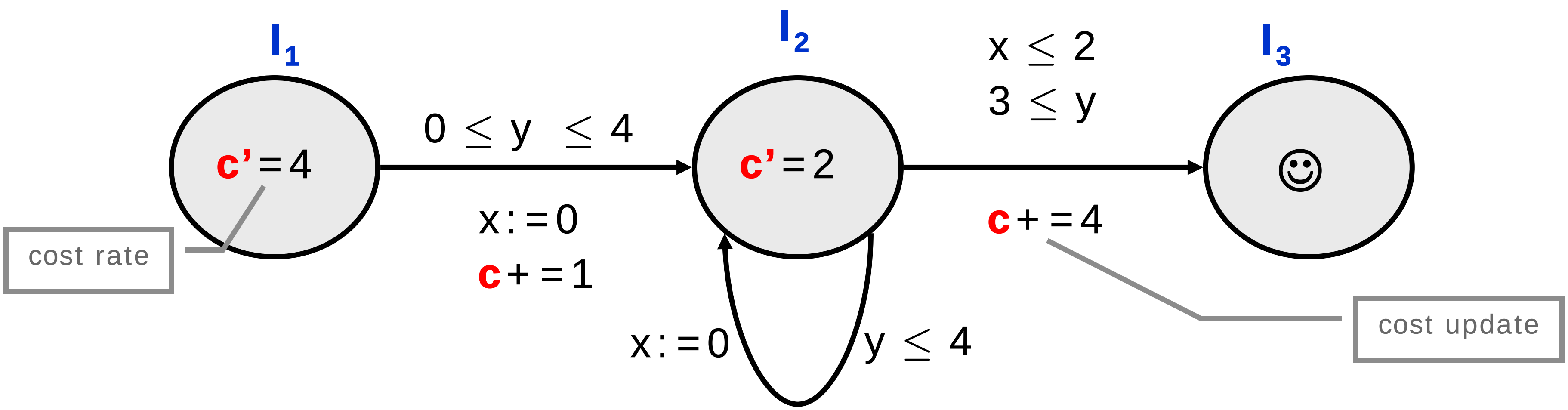
COST-rates				SCHEDULE	COST	TIME	#Expl	#Pop'd
G	C	B	D					
Min Time				CG> G< BD> C< CG>		60	1762 1538	2638
1	1	1	1	CG> G< BG> G< GD>	55	65	252	378
9	2	3	10	GD> G< CG> G< BG>	195	65	149	233
1	2	3	4	CG> G< BD> C< CG>	140	60	232	350
1	2	3	10	CD> C< CB> C< CG>	170	65	263	408
1	20	30	40	BD> B< CB> C< CG>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

Priced Timed Automata

Timed Automata + **COST** variable

Behrmann, Fehnker, et all (HSCC'01)

Alur, Torre, Pappas (HSCC'01)

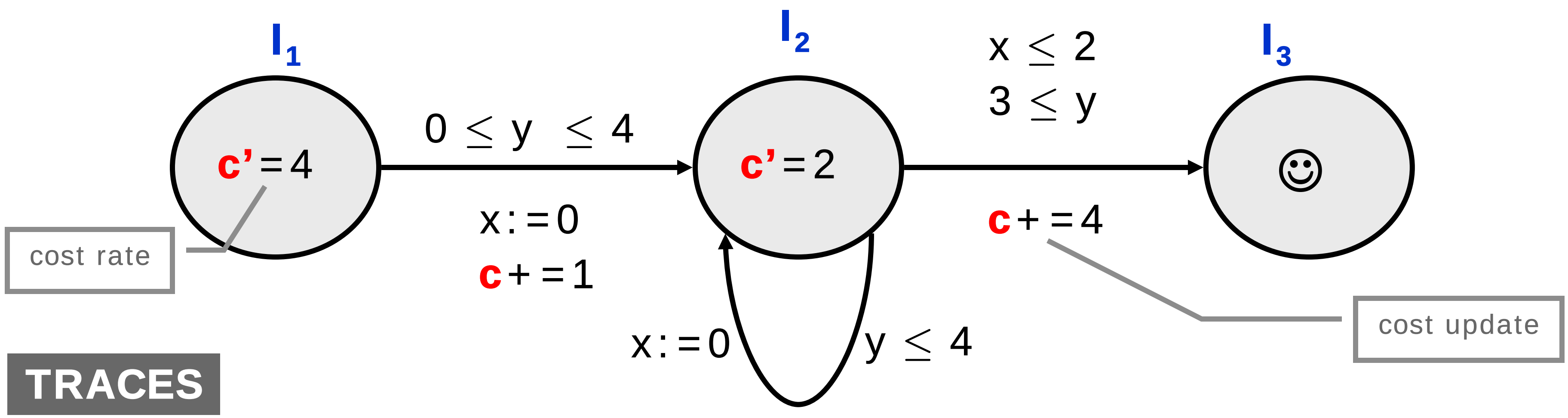


Priced Timed Automata

Timed Automata + **COST** variable

Behrmann, Fehnker, et all (HSCC'01)

Alur, Torre, Pappas (HSCC'01)



TRACES

$$(l_1, x=y=0) \overset{\varepsilon(3)}{\underset{12}{\dashrightarrow}} (l_1, x=y=3) \underset{1}{\longrightarrow} (l_2, x=0, y=3) \underset{4}{\longrightarrow} (l_3, \_, \_)$$

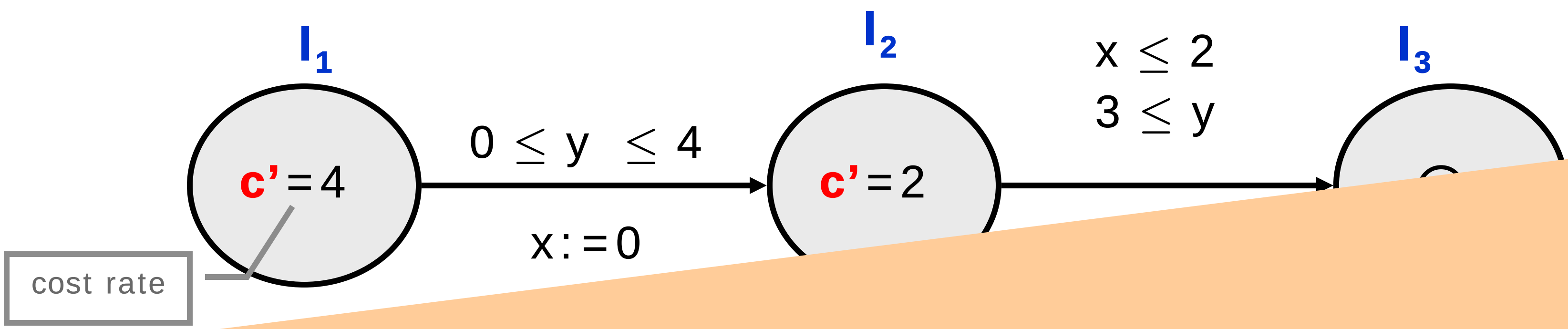
$\sum c = 17$

Priced Timed Automata

Timed Automata + **COST** variable

Behrmann, Fehnker, et all (HSCC'01)

Alur, Torre, Pappas (HSCC'01)



TRA

Problem :

Find the minimum cost of reaching location l_3

Efficient Implementation:

CAV'01 and TACAS'04

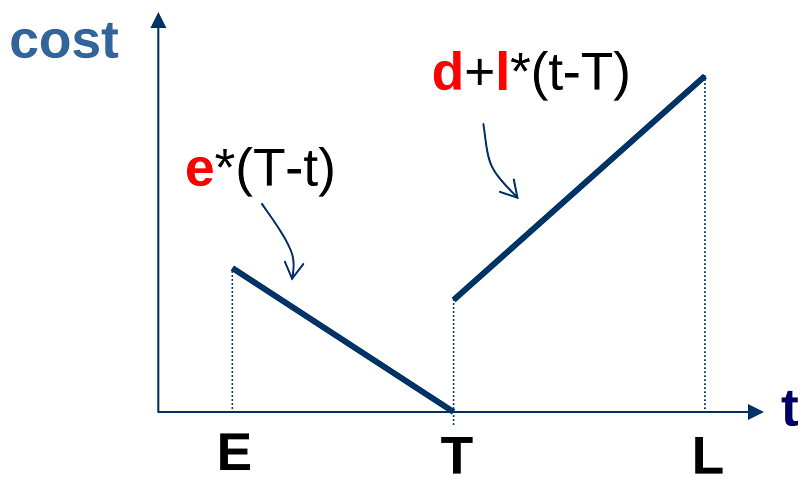
$(l_1$ … $(l_3,\_,\_)$ $\sum c=16$

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

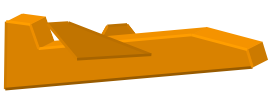
$\sum c=11$

BRICS
Basic Research
in Computer Science

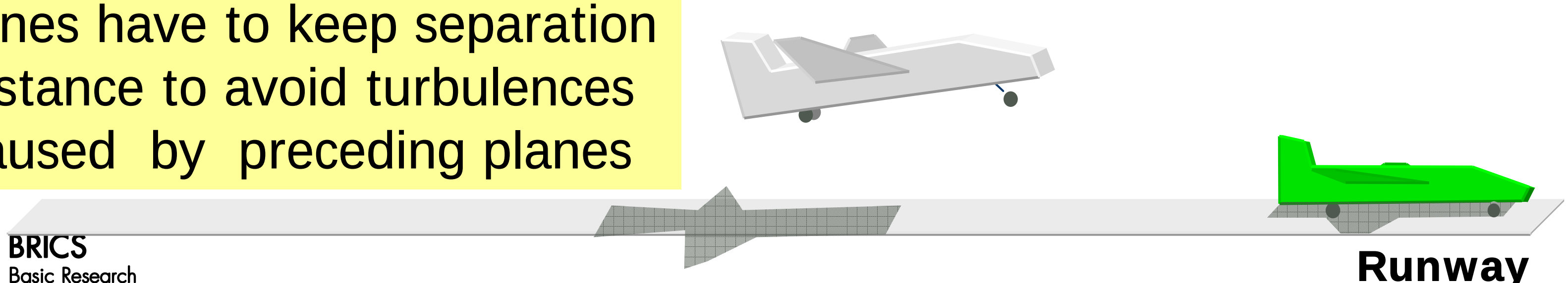
Aircraft Landing Problem



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



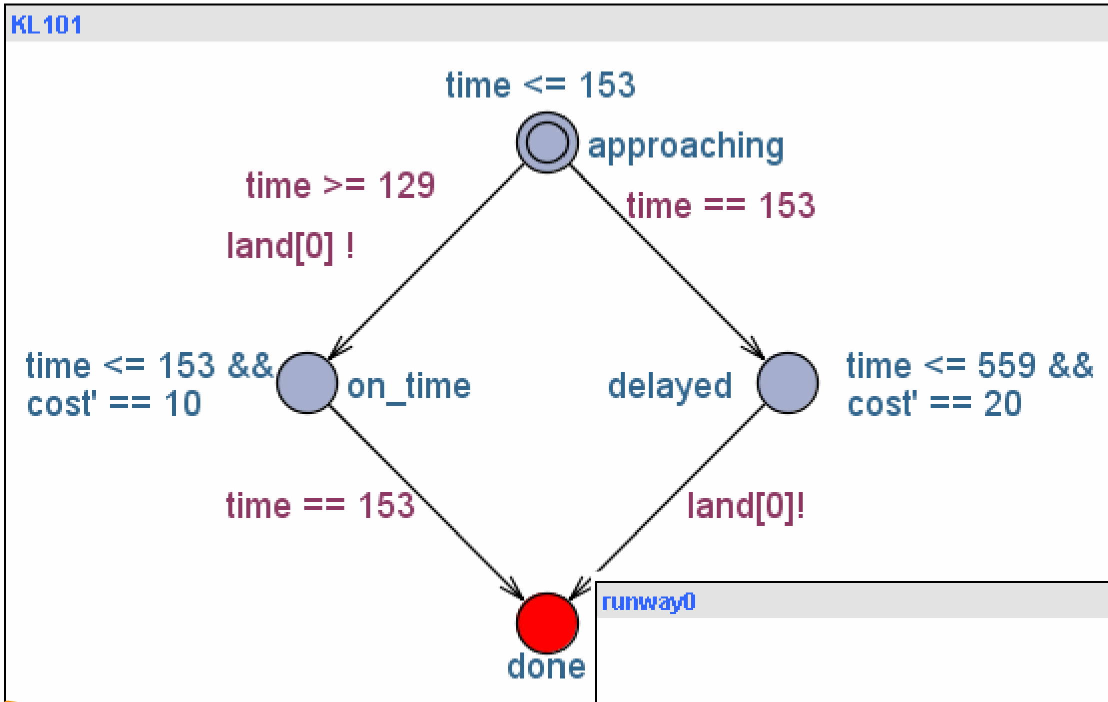
Planes have to keep separation distance to avoid turbulences caused by preceding planes

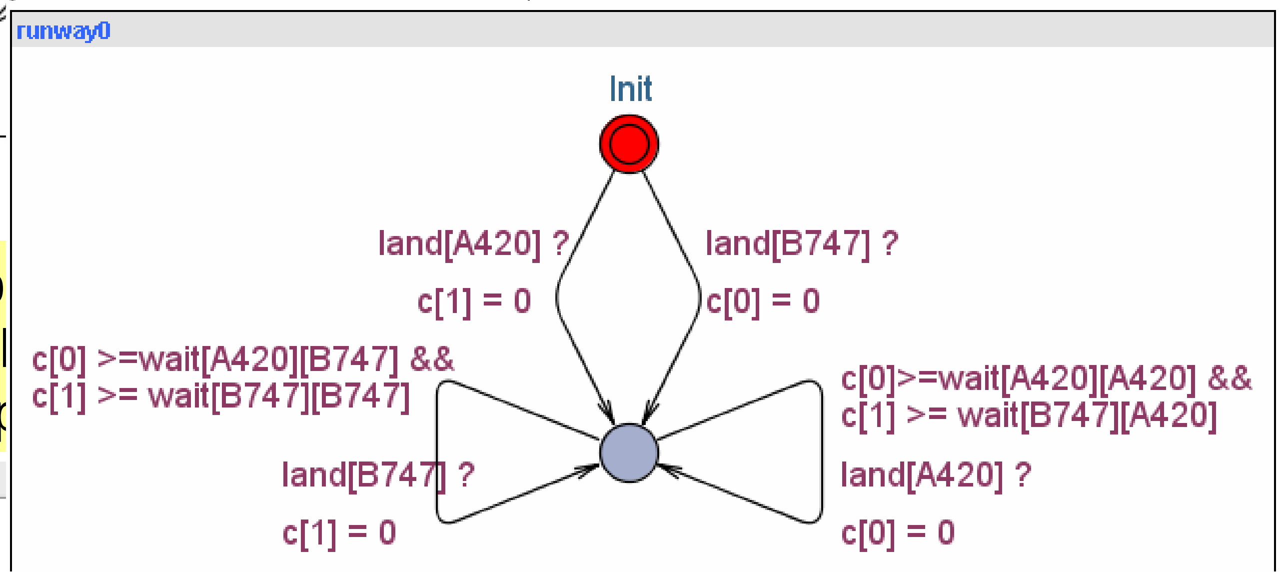


Modeling ALP with PTA

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

BRICS Basic Research in Computer Science

Symbolic "A*"

Zones

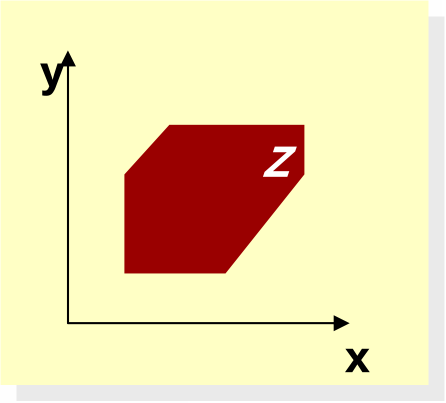
Definition

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.



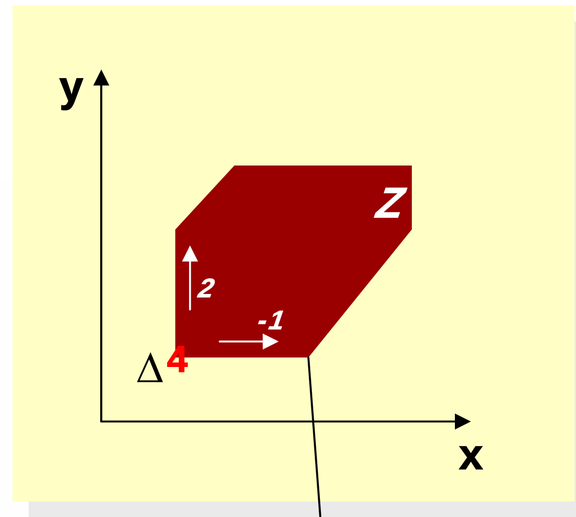
Operations

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



Definition

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

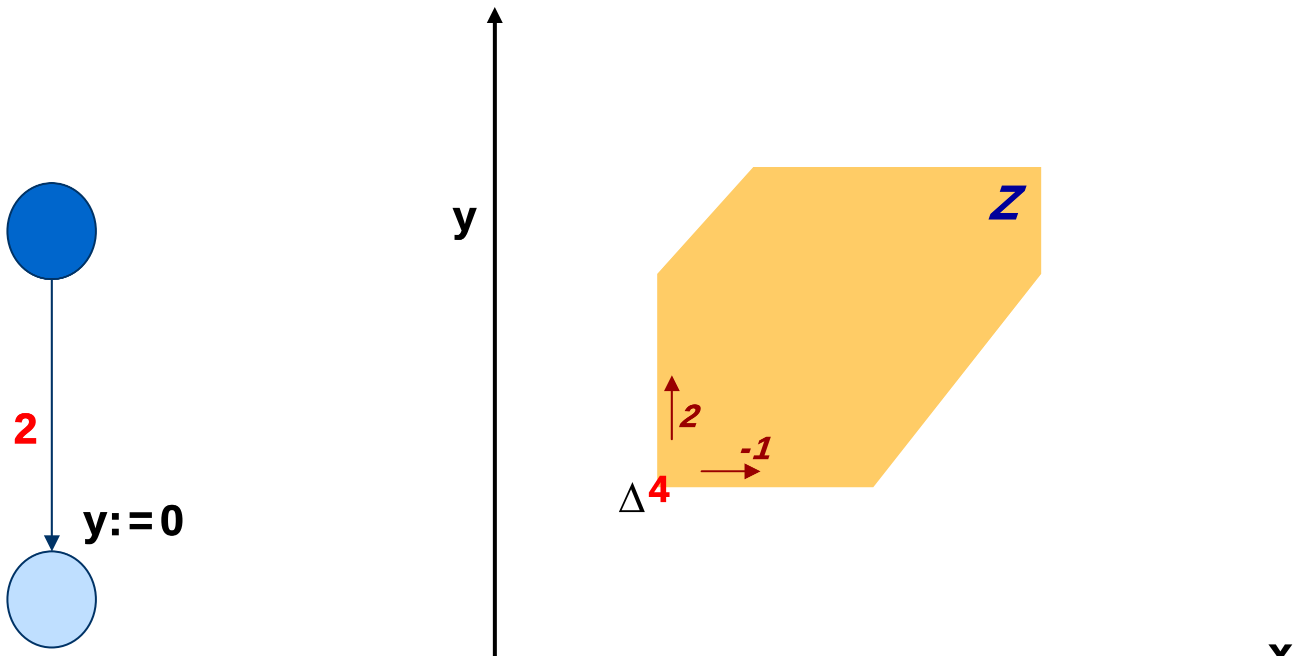
- Z is a zone
- $c \in \mathbb{N}$ describes the cost of Δ_Z
- $r : C \to \mathbb{Z}$ gives a *rate* for any clock $x \in C$.

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

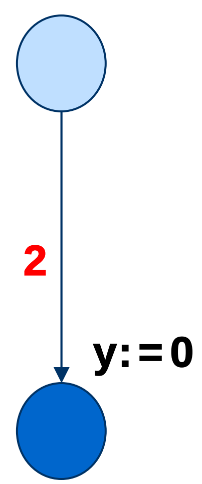
$$\text{Cost}(u, P) = c + \sum_{x \in C} r(x) \cdot (u(x) - \Delta_Z(x))$$

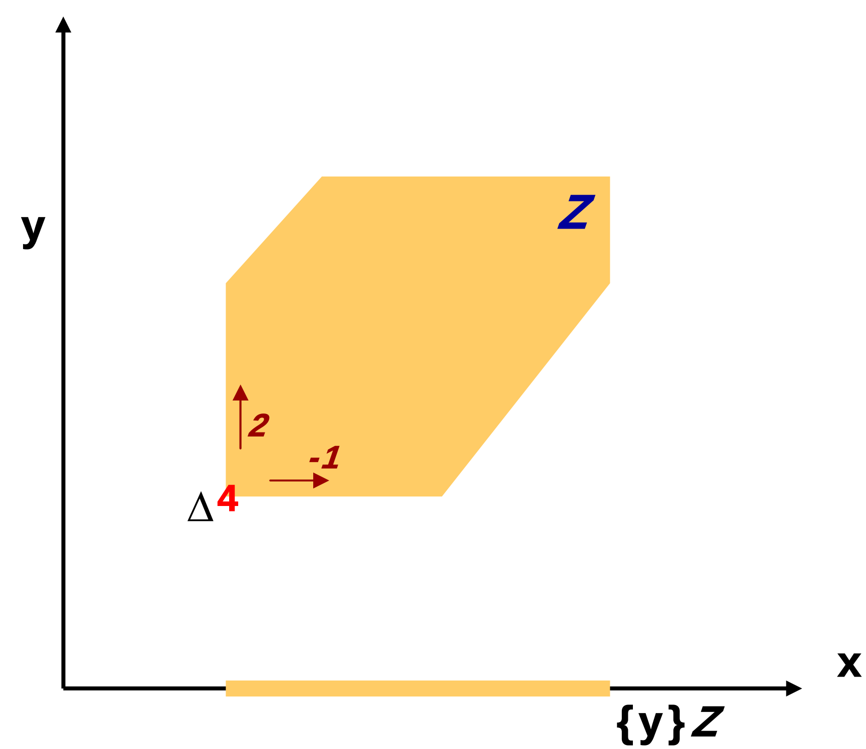
$$Cost(x, y) = 2y - x + 2$$

Reset

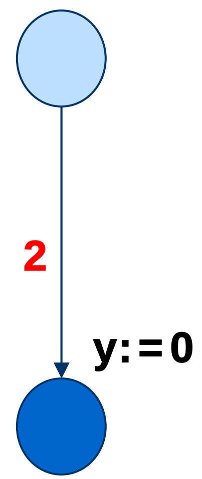


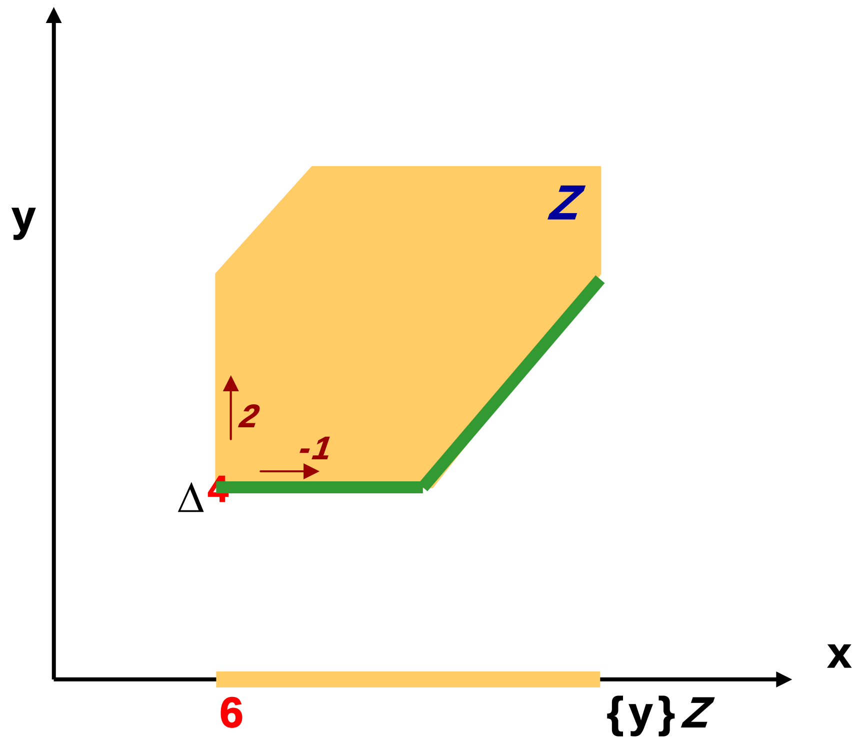
Reset





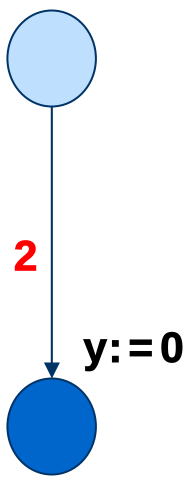
Reset

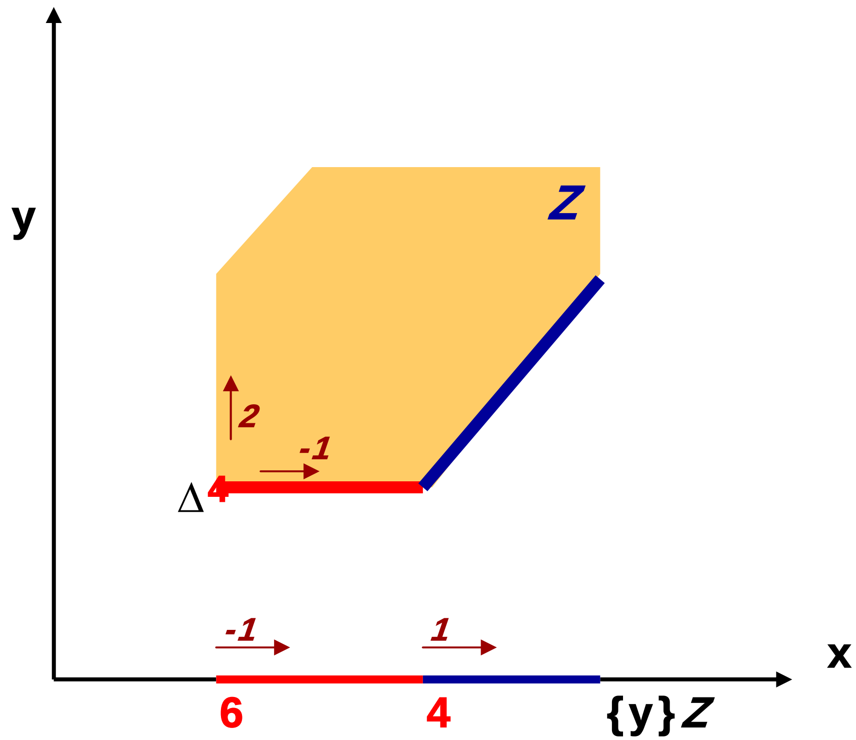




BRICS
Basic Research
in Computer Science

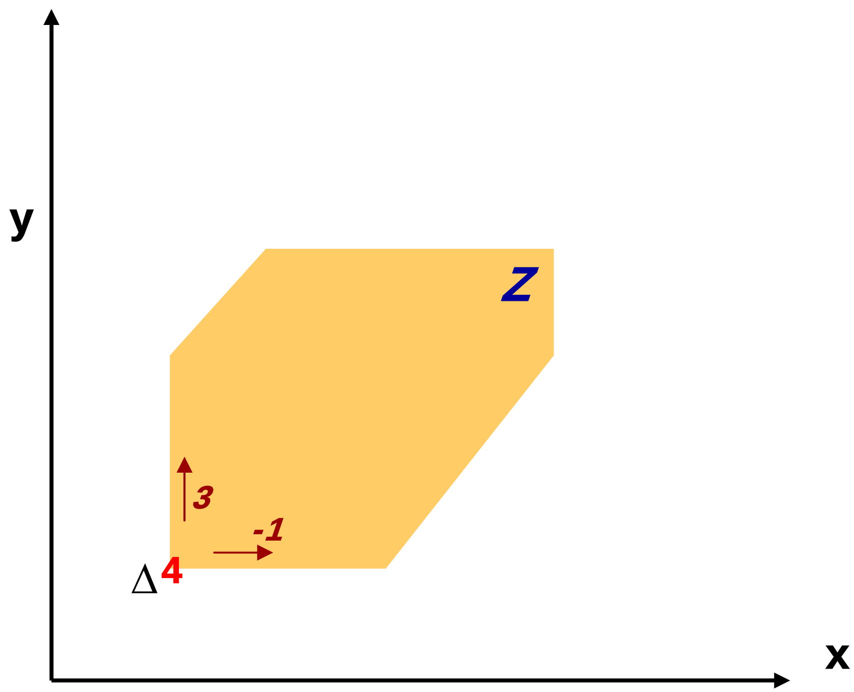
Reset

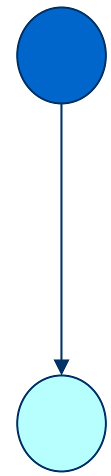




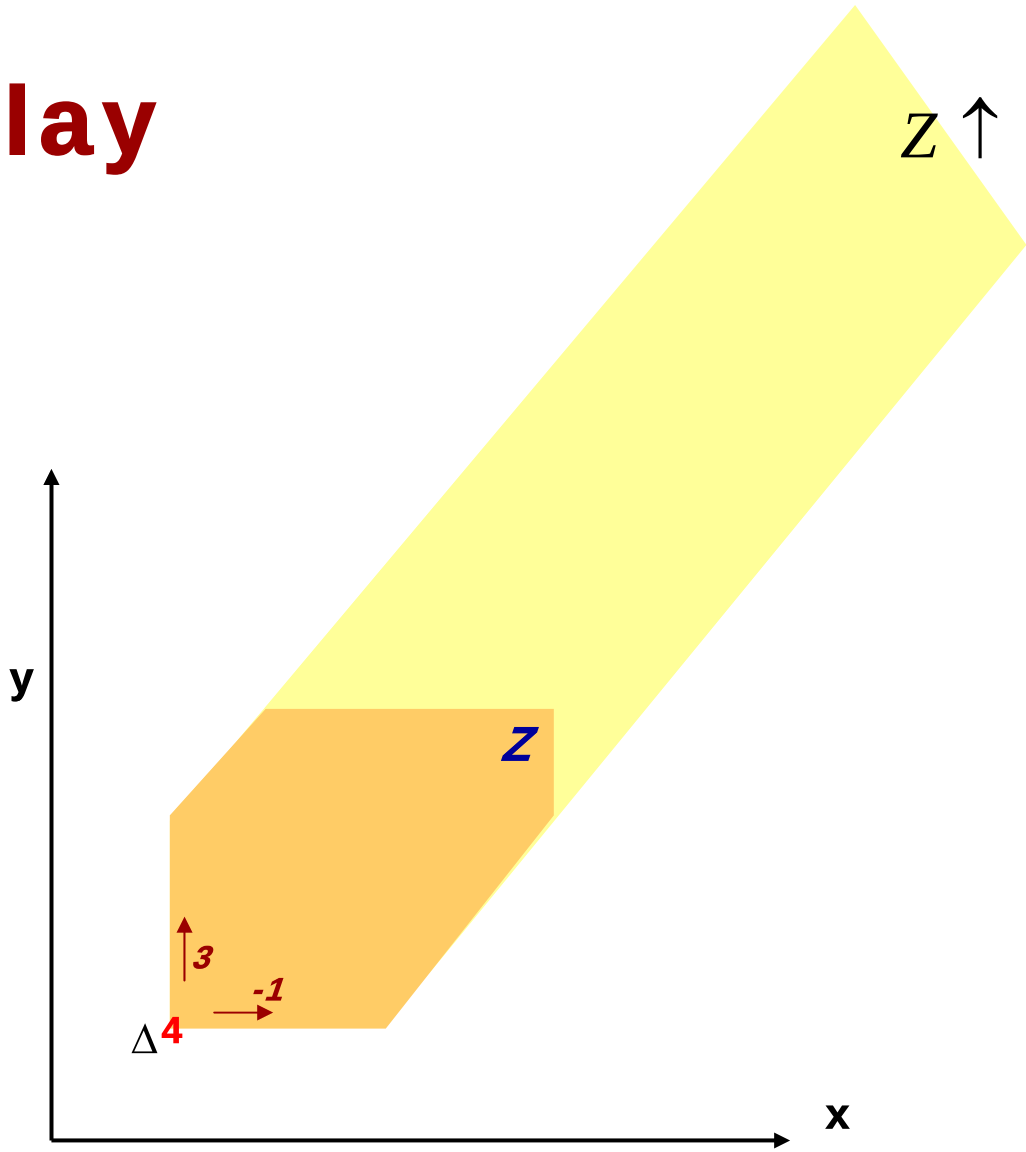
A split of {y}*Z*

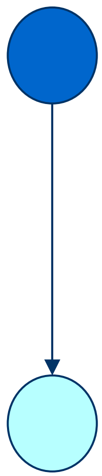
Delay



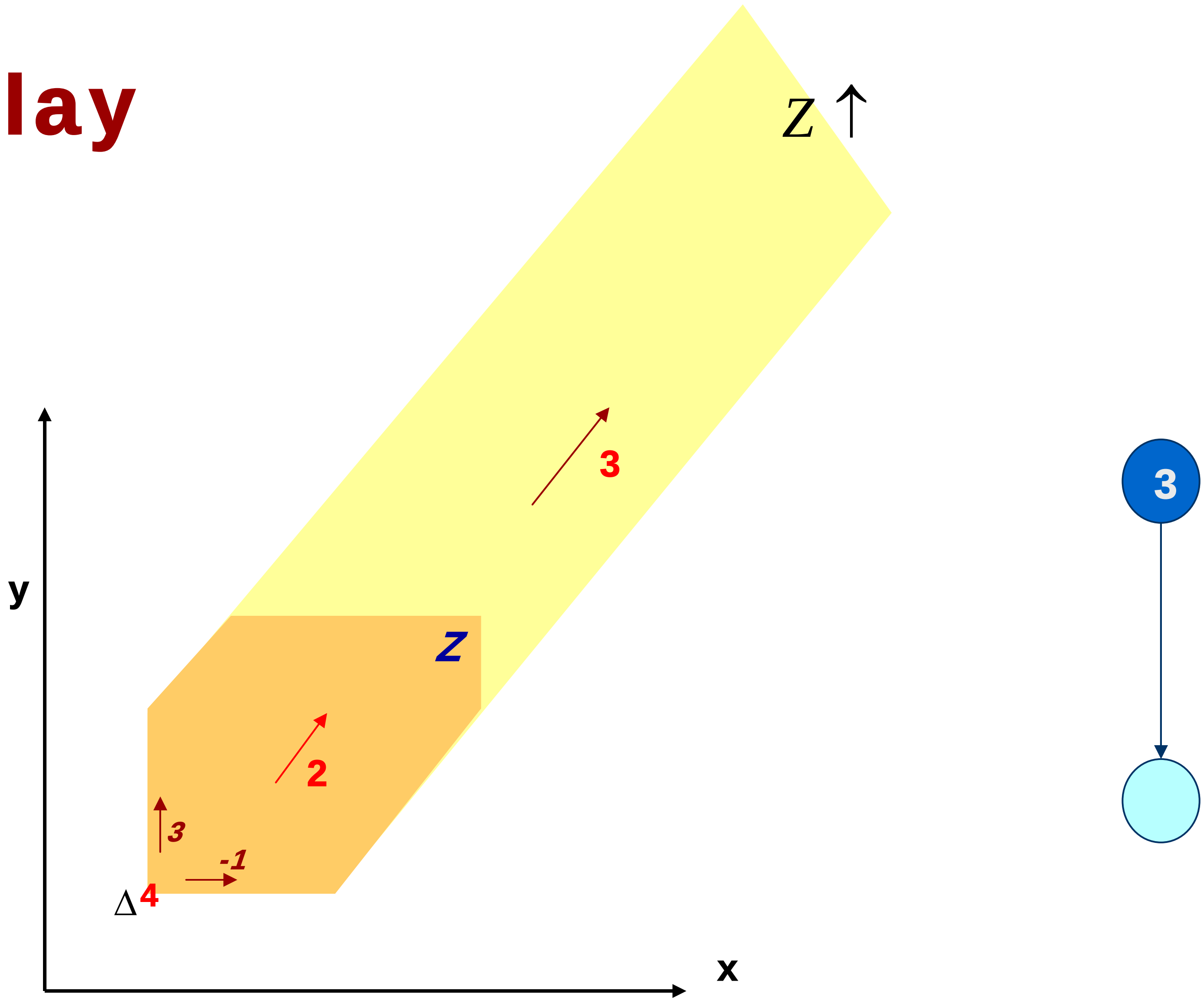


Delay



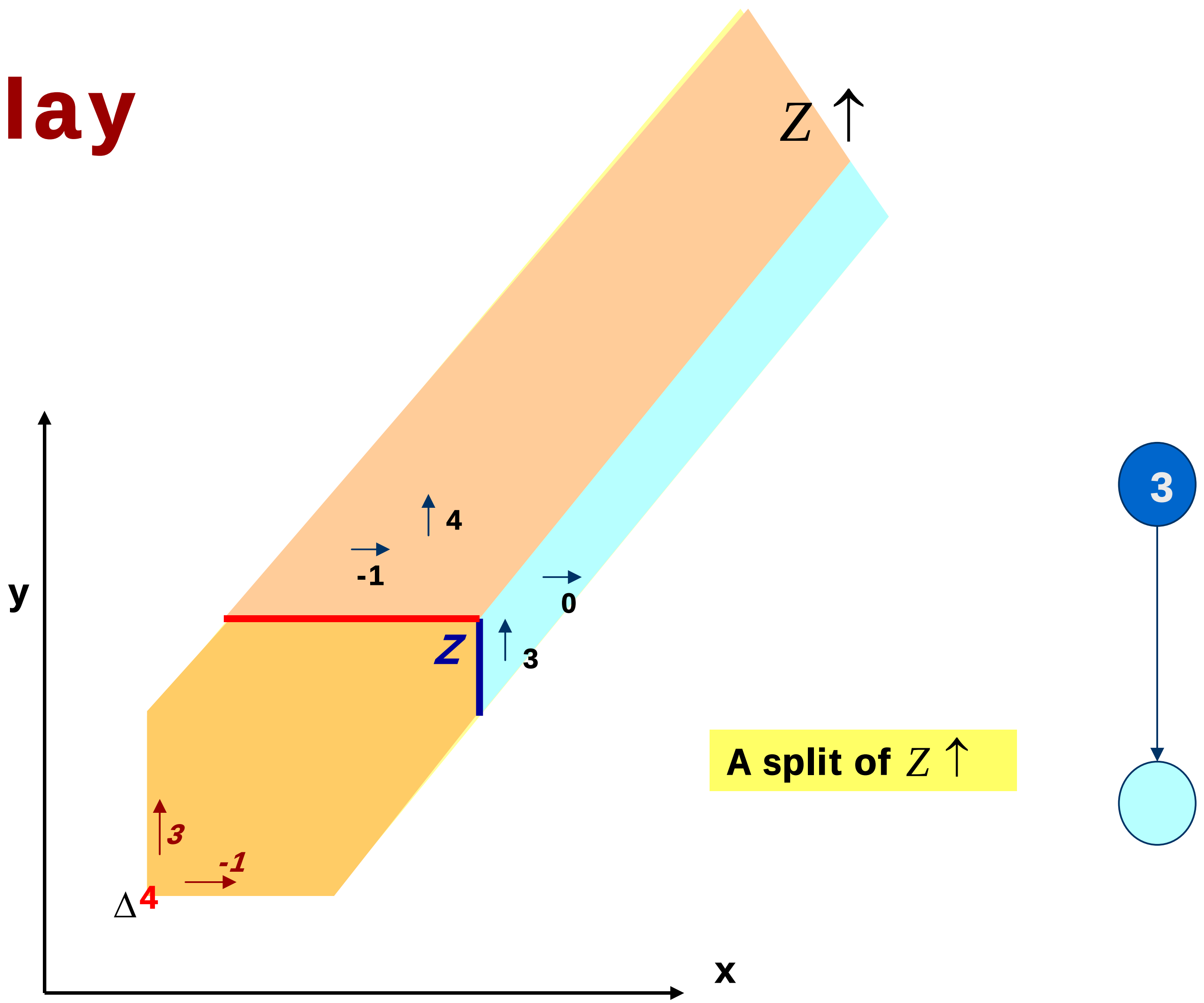


Delay



BRICS Basic Research in Computer Science

Delay



BRICS Basic Research in Computer Science

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) + Rem$_{(l,Z)} \geq$ Cost **then** break

  if for all (l, Z') in Passed: $Z' \not\preceq Z$ **then**

    add (l, Z) to Passed

    add all (l', Z') with $(l, Z) \rightarrow (l', Z')$ to Waiting

return Cost

Branch & Bound Algorithm

$\boxed{\begin{array}{l}\text{Cost} := \infty \\ \text{Passed} := \emptyset \\ \text{Waiting} := \{(l_0, Z_0)\}\end{array}}$

while Waiting $\neq \emptyset$ **do**

  select (l, Z) from Waiting

  if $l = l_g$ and minCost(Z) < Cost **then**

    Cost := minCost(Z)

  if minCost$(Z) +$ Rem$_{(l,Z)} \geq$ Cost **then** break

  if for all (l, Z') in Passed: $Z' \not\preceq Z$ **then**

    add (l, Z) to Passed

    add all (l', Z') with $(l, Z) \rightarrow (l', Z')$ to Waiting

return Cost

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) + Rem$_{(l,Z)} \geq$ Cost **then** break

  if for all (l, Z') in Passed: $Z' \not\preceq Z$ **then**

    add (l, Z) to Passed

    add all (l', Z') with $(l, Z) \rightarrow (l', Z')$ to Waiting

return Cost

BRICS Basic Research in Computer Science

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) + 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

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)

  $\boxed{\textbf{if } \text{minCost}(Z) + \text{Rem}_{(l,Z)} \geq \text{Cost } \textbf{then} \text{ break}}$

  if for all (l, Z') in Passed: $Z' \not\subseteq Z$ **then**

    add (l, Z) to Passed

    add all (l', Z') with $(l, Z) \rightarrow (l', Z')$ to Waiting

return Cost

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$(Z) + \text{Rem}_{(l,Z)} \geq$ C... **th**...

  if for all (l, Z') in Passed: $Z' \not\leq Z$ **then**

    add (l, Z) to Passed

    add all (l', Z') with $(l, Z) \rightarrow (l', ...$

return Cost

$Z' \leq Z$

Z'* is bigger & cheaper than *Z

$\leq$ is a well-quasi ordering which guarantees termination!

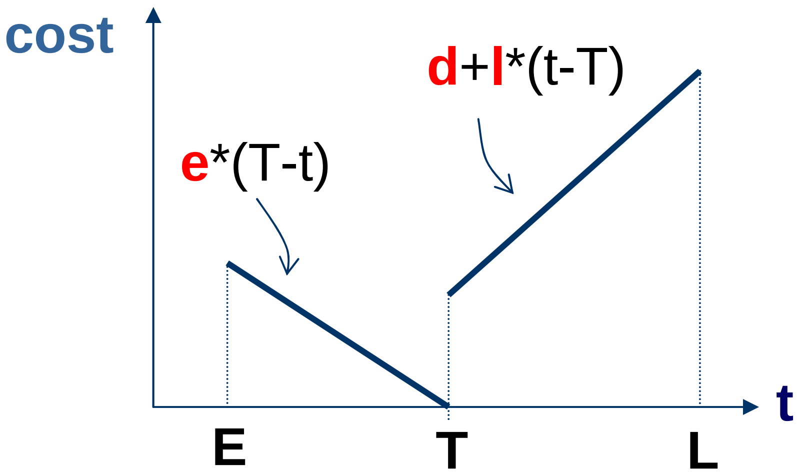
Experimental Results

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

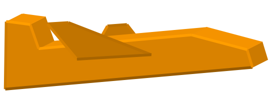
Experiments *MC Order*

COST-rates				SCHEDULE	COST	TIME	#Expl	#Pop'd
G_5	C_{10}	B_{20}	D_{25}					
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

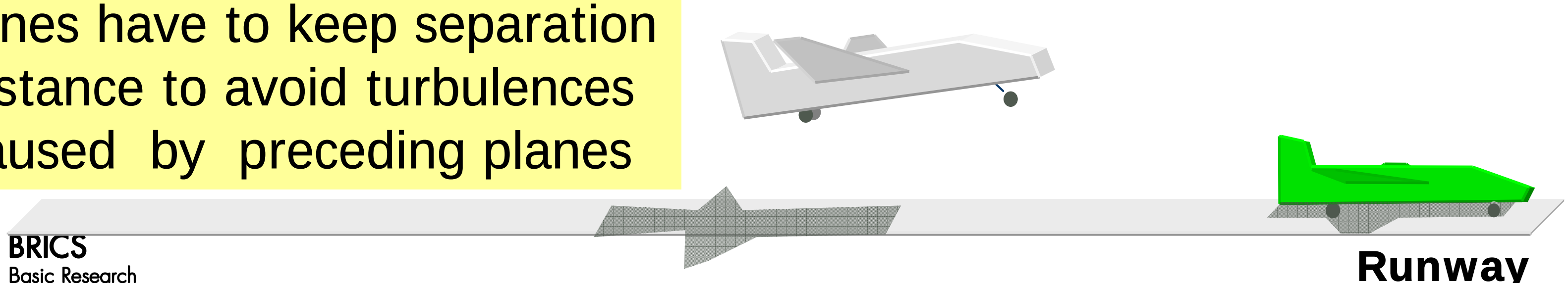
Example: Aircraft Landing



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

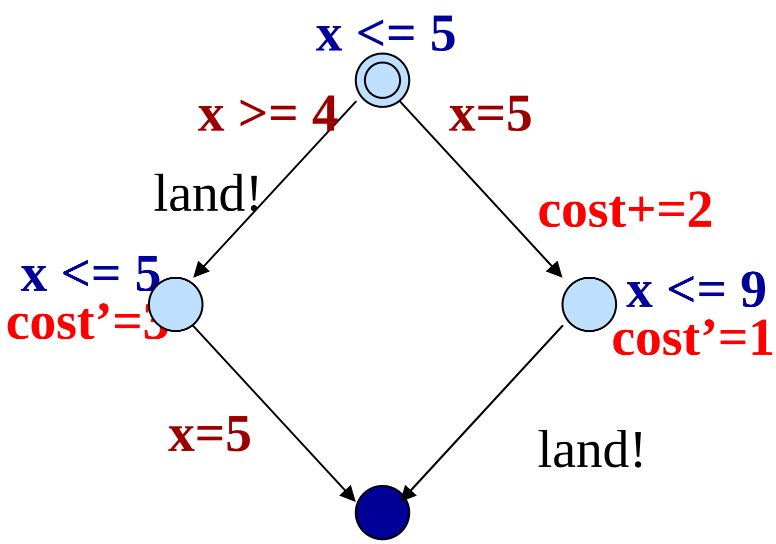


Planes have to keep separation distance to avoid turbulences caused by preceding planes

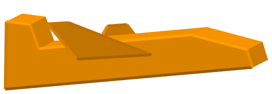


BRICS
Basic Research
in Computer Science

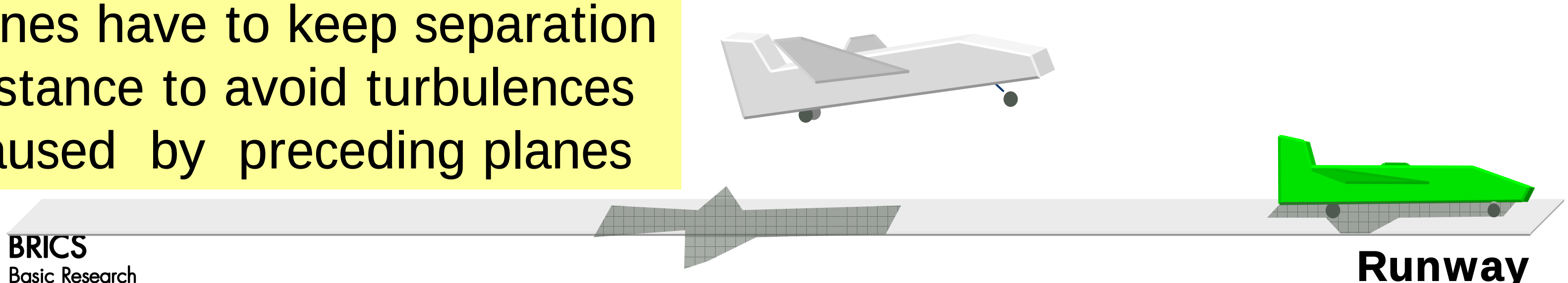
Example: Aircraft Landing



4 earliest landing time
5 target time
9 latest time
3 cost rate for being early
1 cost rate for being late
2 fixed cost for being late



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



BRICS Basic Research in Computer Science

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		

BRICS Basic Research in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Branch & Bound Algorithm

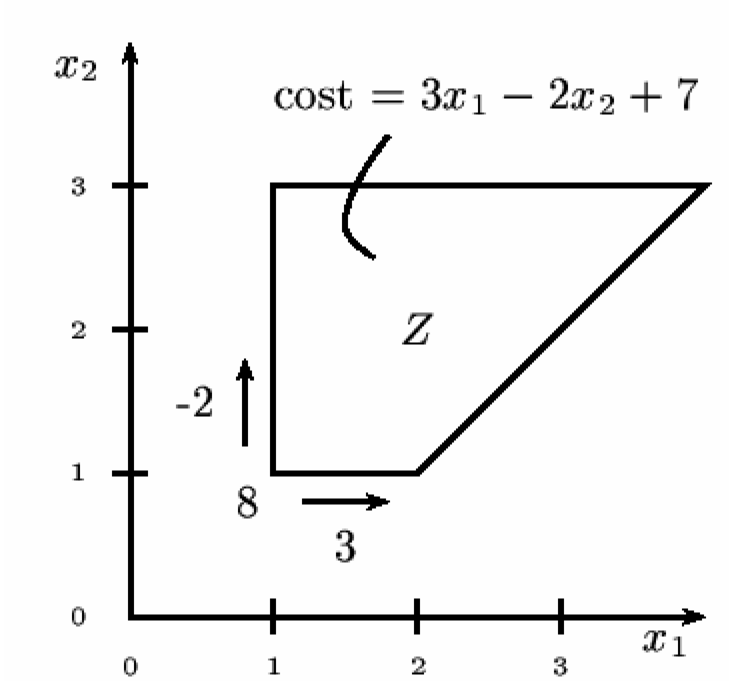
Zone based Linear Programming Problems

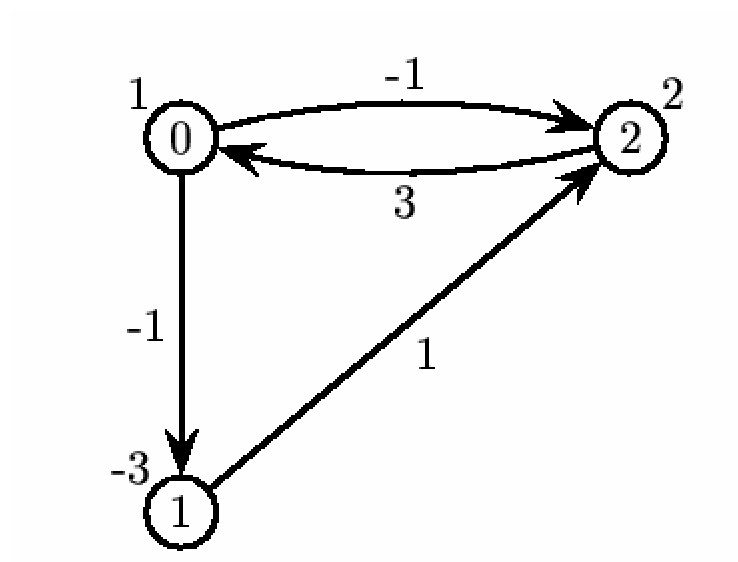
Cost := ∞
Passed := $\emptyset$
Waiting := $\{(l_0, Z_0)\}$
while Waiting $\neq \emptyset$ **do**
    select (l, Z) from Waiting
    if $l = l_g$ and minCost$(Z) <$ Cost **then**
        Cost := minCost(Z)
    if minCost(Z) + Rem$_{(l,Z)} \geq$ Cost **then** break
    if for all (l, Z') in Passed: $Z'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

BRICS
Basic Research
in Computer Science

Zone LP → Min Cost Flow

Exploiting duality





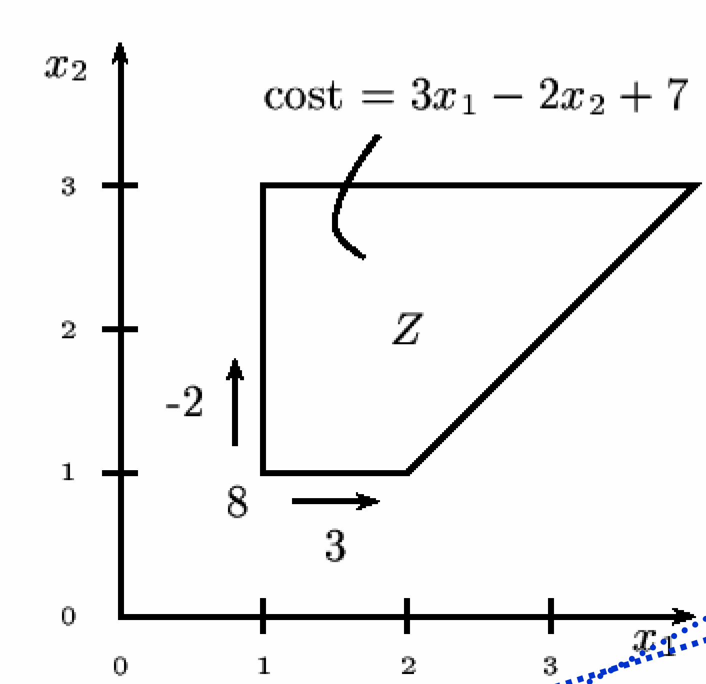
minimize $\mathbf{3x_1 - 2x_2 + 7}$
when $x_1 - x_2 \leq 1$
$1 \leq x_2 \leq 3$
$x_2 \geq 1$

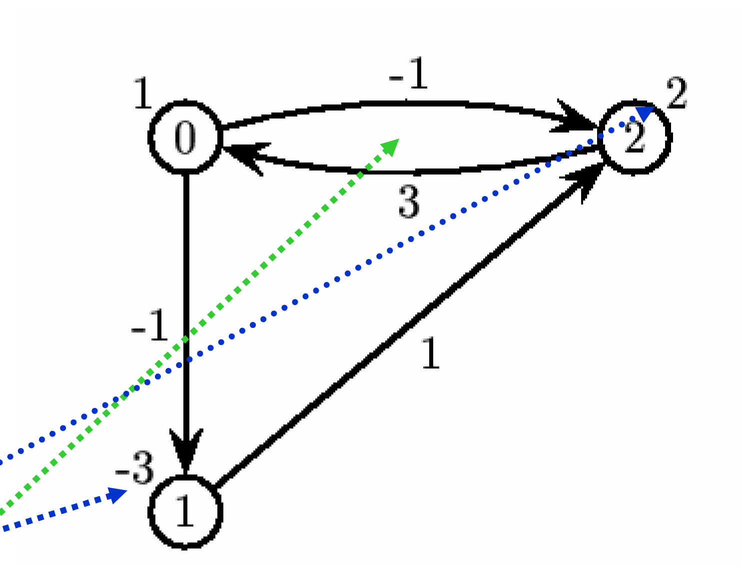
minimize $\mathbf{3y_{2,0} - y_{0,2} + y_{1,2} - y_{0,1}}$
when $y_{2,0} - y_{0,1} - y_{0,2} = 1$
$y_{0,2} + y_{1,2} = 2$
$y_{0,1} - y_{1,2} = -3$

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Zone LP → Min Cost Flow

Exploiting duality





minimize $\mathbf{3x_1 - 2x_2 + 7}$

when $x_1 - x_2 \leq 1$

$1 \leq x_2 \leq 3$

$x_2 \geq 1$

minimize $\mathbf{3y_{2,0} - y_{0,2} + y_{1,2} - y_{0,1}}$

when $y_{2,0} - y_{0,1} - y_{0,2} = 1$

$y_{0,2} + y_{1,2} = 2$

$y_{0,1} - y_{1,2} = -3$

Aircraft Landing

Using MCF / Netsimplex

Rasmussen, Larsen, Subramani
TACAS04

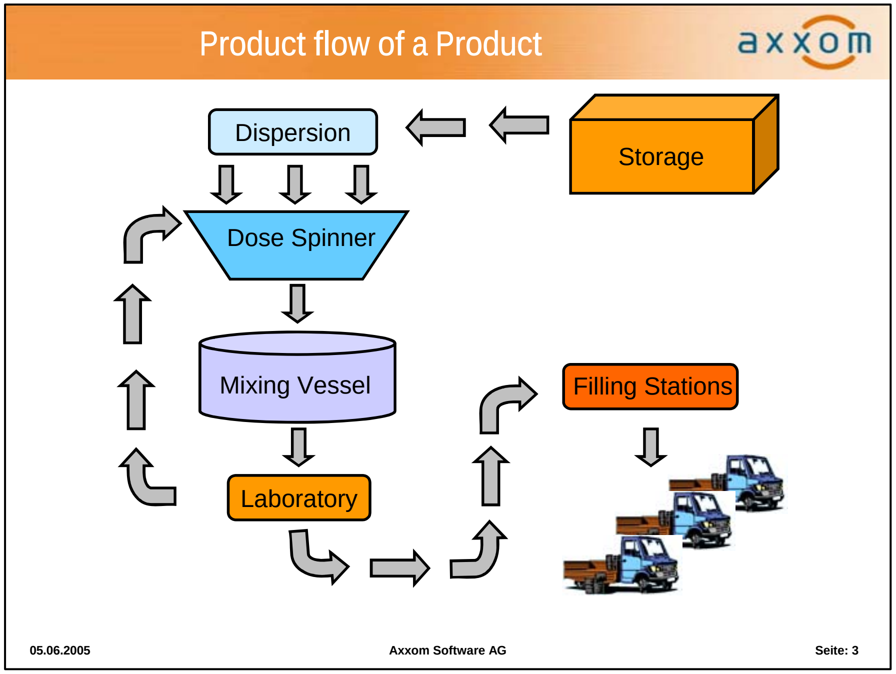
RW	Planes	10	15	20	20	20	30	44
	Types	2	2	2	2	2	4	2
1	simplex	0.844s	5.210s	2.135s	17.888s	44.878s	0.451s	0.670s
	netsimplex	0.156s	0.657s	0.369s	2.363s	5.503s	0.127s	0.322s
factor		**5.41**	**7.93**	**5.79**	**7.57**	**8.16**	**3.55**	**2.08**
2	simplex	2.577s	7.436s	2.175s	94.357s	120.004s	2.322s	0.264s
	netsimplex	0.332s	1.036s	0.436s	13.376s	18.033s	0.600s	0.179s
factor		**8.00**	**7.18**	**4.99**	**7.054**	**6.65**	**3.87**	**1.474**
3	simplex	0.120s	0.181s	0.357s	740.100s	516.678s	0.166s	N/A
	netsimplex	0.064s	0.104s	0.129s	170.176s	124.805s	0.079s	N/A
factor		**1.87**	**1.74**	**2.77**	**4.34**	**4.14**	**2.10**	
4	simplex	N/A	N/A	N/A	1.603s	0.318s	N/A	N/A
	netsimplex	N/A	N/A	N/A	0.378s	0.093s	N/A	N/A
factor					**4.24**	**3.42**		

A. Loebel (2000). MCF Version 1.2 - A network simplex implementation. (http://www.zib.de)

AXXOM Case study

Laquer Production Scheduling

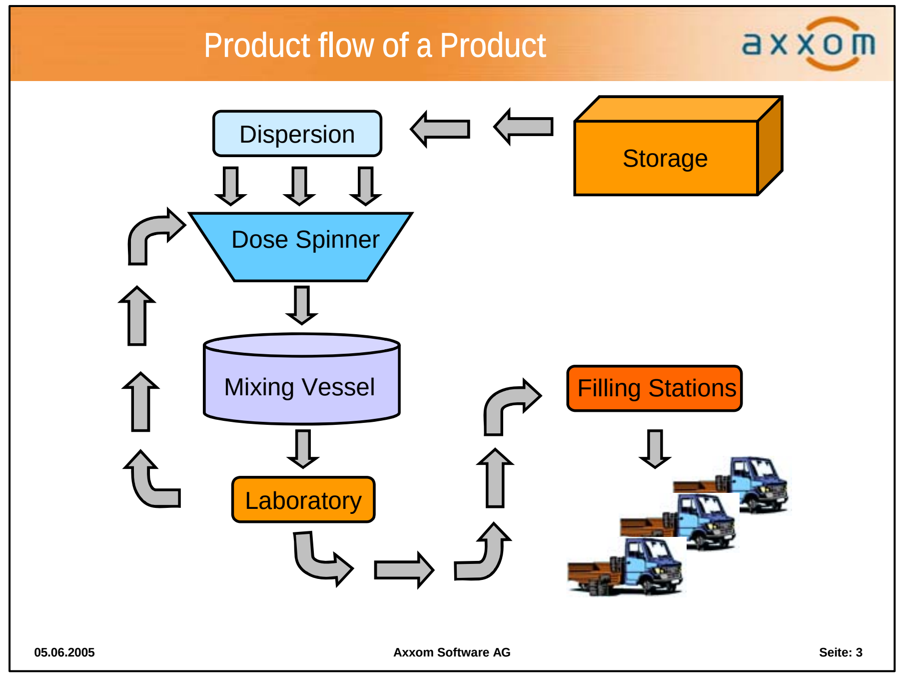
- 3 types of recipes
  - for uni/metallic/bronce
  - use of resources, processing times, timing
- 29 (73, 219) orders:
  - start time, due date, recipe
- extensions:
  - delay cost, storage cost, setup cost
  - weekend, nights



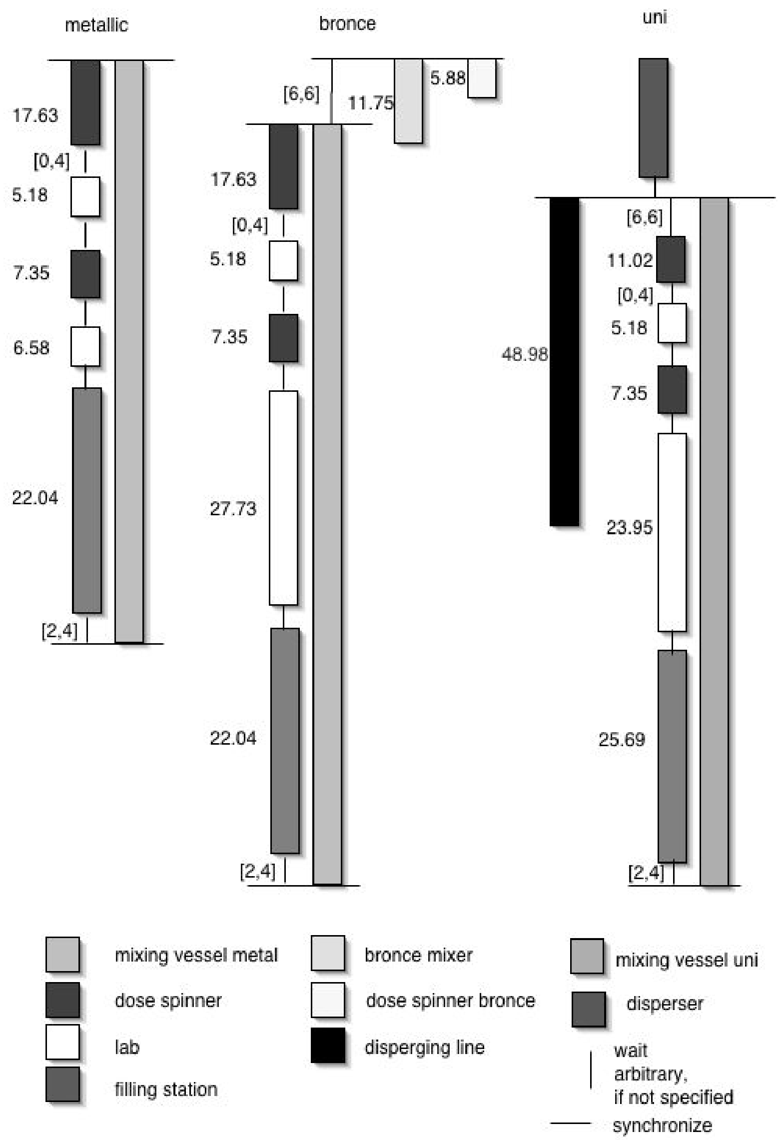
Behrmann, Brinksma, Hendriks, Mader
16th IFAC World Congress

Resources

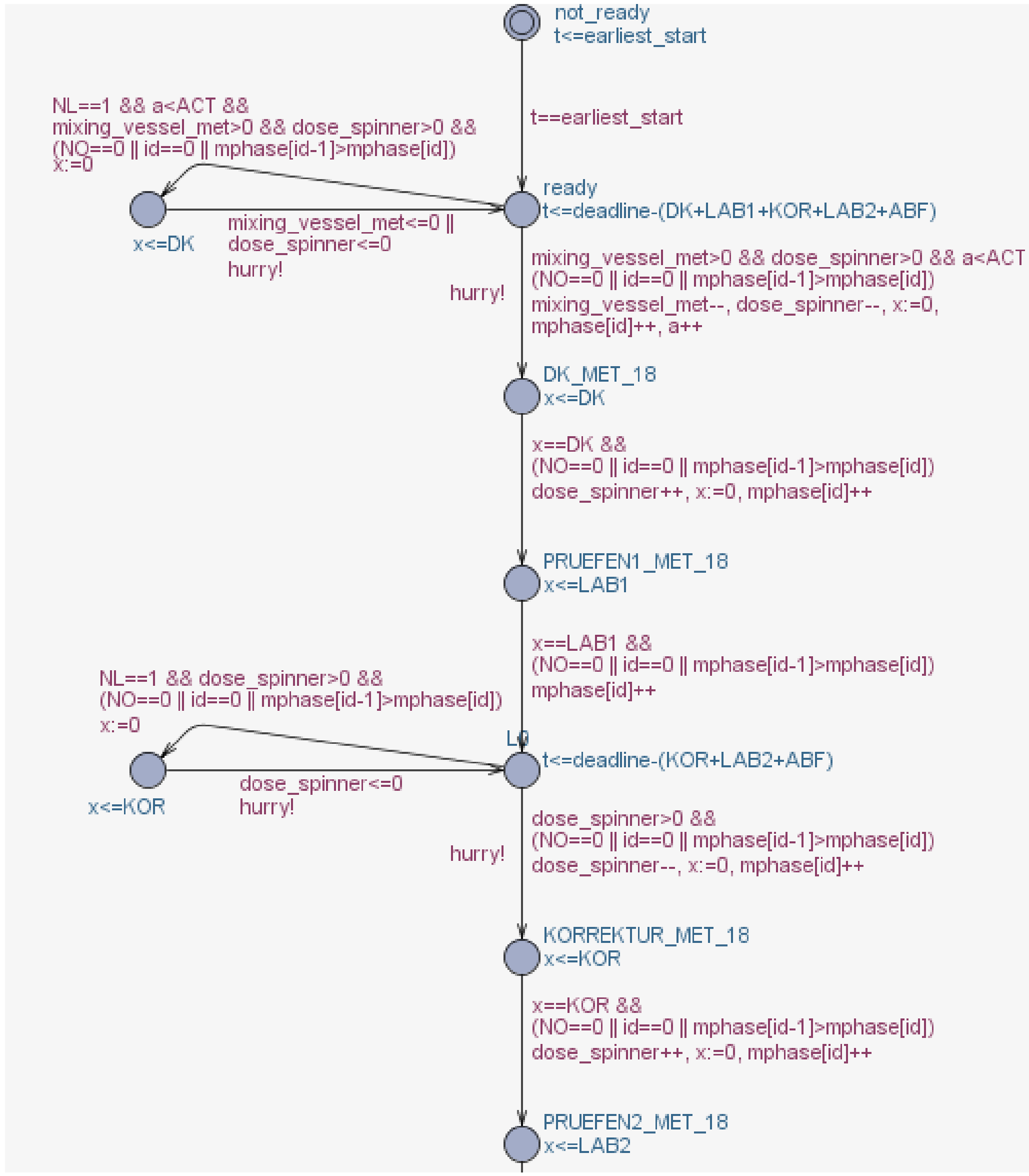
- 2 mixing vessels for uni lacquers
- 3 mixing vessels for metallic/bronce
- 2 dose spinners
- 1 dose spinner bronce
- 1 disperging line
- 1 predisperser
- 1 bronce mixer
- 2 filling lines
- lab (unlimitted)



Recipes



UPPAAL template for metal



Orders

order	product ID	quantity	unit	delay costs per day per kg	earliest start	delivery date
1204081.FE	10009.ABF.MET.18	19000	kg	51	05.04.2002 19:12	19.04.2002 19:12
4147108.FE	10009.ABF.MET.18	19000	kg	51	20.04.2002 06:00	04.05.2002 06:00
4152856.FE	10009.ABF.MET.18	19000	kg	51	29.03.2002 07:12	12.04.2002 07:12
4161304.FE	10001.ABF.UNI.85	23000	kg	51	08.04.2002 04:00	22.04.2002 04:00
4173559.FE	10001.ABF.UNI.85	22000	kg	0,51	01.04.2002 22:48	15.04.2002 22:48
4173830.FE	10001.ABF.UNI.85	22000	kg	0,51	15.04.2002 02:48	29.04.2002 02:48
4174631.FE	10009.ABF.MET.18	19000	kg	51	22.04.2002 22:48	06.05.2002 22:48
4179351.FE	10009.ABF.MET.18	19000	kg	51	01.04.2002 23:59	15.04.2002 23:59
4179360.FE	10009.ABF.MET.18	19000	kg	51	29.03.2002 19:12	12.04.2002 19:12
4187241.FE	10001.ABF.UNI.85	22000	kg	0,51	15.04.2002 22:48	29.04.2002 22:48
4187676.FE	10009.ABF.MET.18	19000	kg	51	12.04.2002 19:12	26.04.2002 19:12
4187688.FE	10009.ABF.MET.18	19000	kg	51	12.04.2002 23:59	26.04.2002 23:59
4189165.FE	10009.ABF.MET.18	19000	kg	51	01.04.2002 04:00	15.04.2002 04:00
4189694.FE	10001.ABF.UNI.85	22000	kg	51	03.04.2002 10:24	17.04.2002 10:24
4196932.FE	10001.ABF.UNI.85	22000	kg	51	25.03.2002 04:39	08.04.2002 04:39
4196968.FE	10001.ABF.UNI.85	22000	kg	51	02.05.2002 14:24	16.05.2002 14:24
4197088.FE	10009.ABF.MET.18	19000	kg	51	27.03.2002 04:48	10.04.2002 04:48
4197955.FE	10001.ABF.UNI.85	22000	kg	51	03.04.2002 14:24	17.04.2002 14:24
4200741.FE	10009.ABF.MET.18	19000	kg	51	16.04.2002 04:48	30.04.2002 04:48
4206483.FE	10037.ABF.MET.11	19000	kg	51	15.04.2002 23:59	29.04.2002 23:59
4206683.FE	10037.ABF.MET.11	19000	kg	51	15.04.2002 04:00	29.04.2002 04:00
4209227.FE	10009.ABF.MET.18	19000	kg	51	12.04.2002 14:24	26.04.2002 14:24
4210646.FE	10001.ABF.UNI.85	22000	kg	51	26.04.2002 14:24	10.05.2002 14:24
4210669.FE	10037.ABF.MET.11	19000	kg	51	08.04.2002 04:00	22.04.2002 04:00
4213014.FE	10009.ABF.MET.18	19000	kg	51	23.03.2002 02:24	06.04.2002 02:24
4213016.FE	10009.ABF.MET.18	19000	kg	51	20.04.2002 02:24	04.05.2002 02:24
4220296.FE	10009.ABF.MET.18	19000	kg	51	27.03.2002 09:36	10.04.2002 09:36
4222941.FE	10037.ABF.MET.11	19000	kg	0,51	03.04.2002 06:00	17.04.2002 06:00
4223024.FE	10037.ABF.MET.11	19000	kg	0,51	18.04.2002 06:00	02.05.2002 06:00

Instantiated Model

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Heuristics

- **Nice heuristics**
  - non-overtaking

    orders of the same recipe cannot overtake each other
  - non-laziness

    a process that needs an available resource will not waste time if its is not claimed by others (a.k.a. active scheduling)
- **Cut-and-Pray heuristics**
  - greediness

    a process that needs an available resource will claim this resource immediately
  - reducing active orders

    the number of concurrent orders is restricted (number of critical resources can give an indication)

Experimental Results

#jobs	heuristic	max. orders	term. time
29	-	-	-
29	nl	-	1 s
73	nl, no	-	-
73	nl, no	3	7 s
219	g, no	4	8 s

uses clock optimization & optimized successor calculation

Results Extended Case

storage, delay and setup costs, working hours

#jobs	work hrs	isti	ma	
29				
2				
			-	14,875
2		o, g	-	2,263,496
29	ex	no	4	192,881,129

Competitive with Orion-pi results

Order of magnitude faster than MILP, GAMS/CPLEX

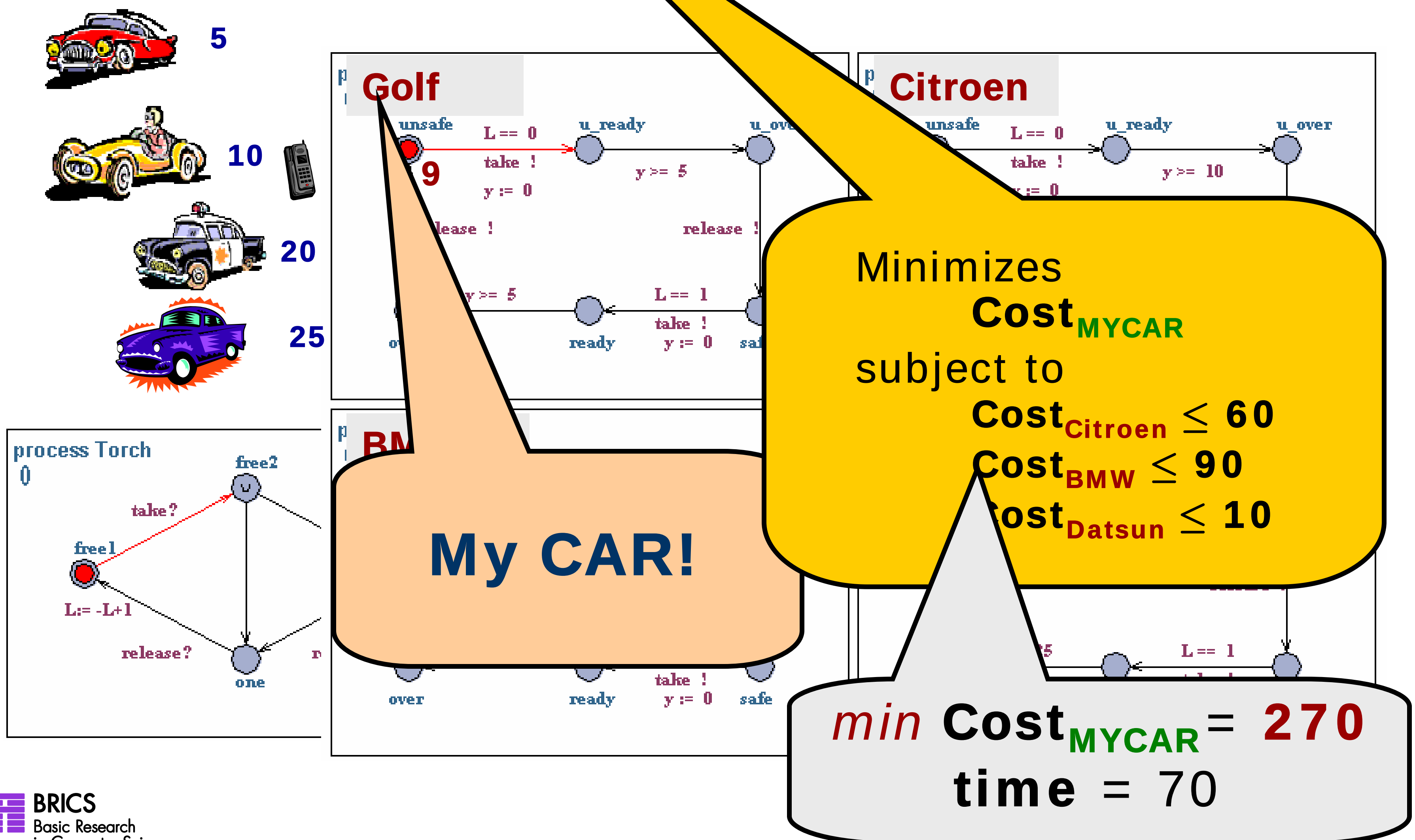
Optimal Conditional Reachability

with Jacob I. Rasmussen

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

CONDITIONAL

UNSAFE

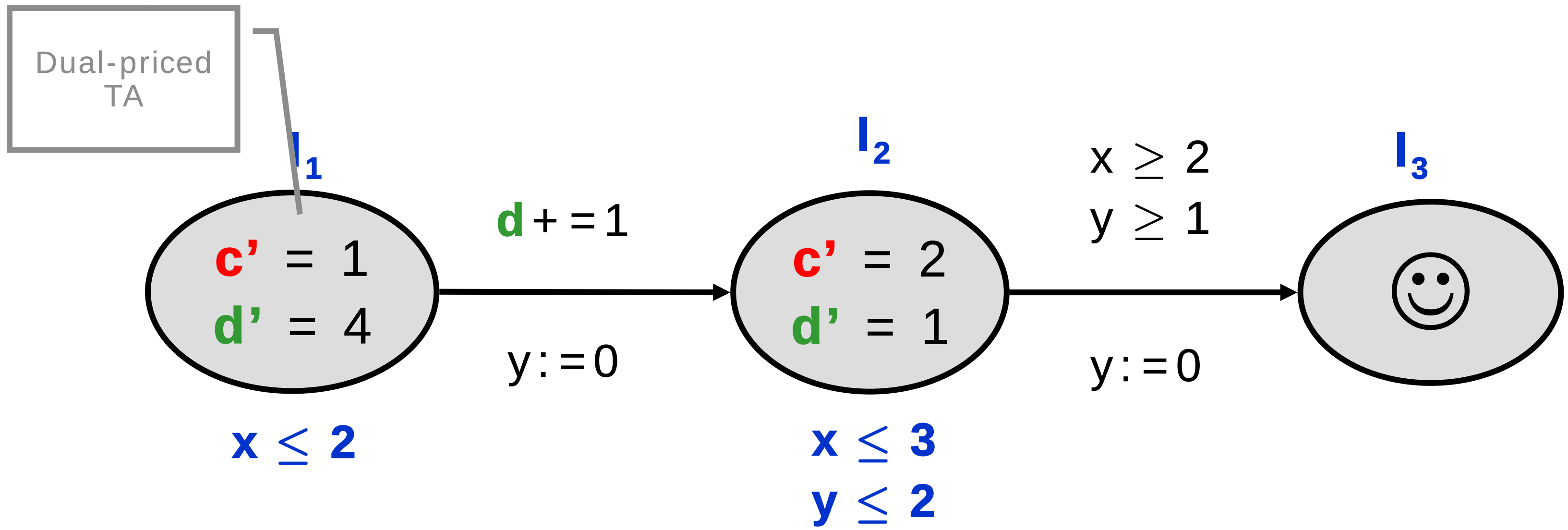


CISS CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

BRICS Basic Research in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Optimal *Conditional* Reachability



PROBLEM:

Reach l_3 in a way which minimizes c subject to $d \leq 4$

SOLUTION:

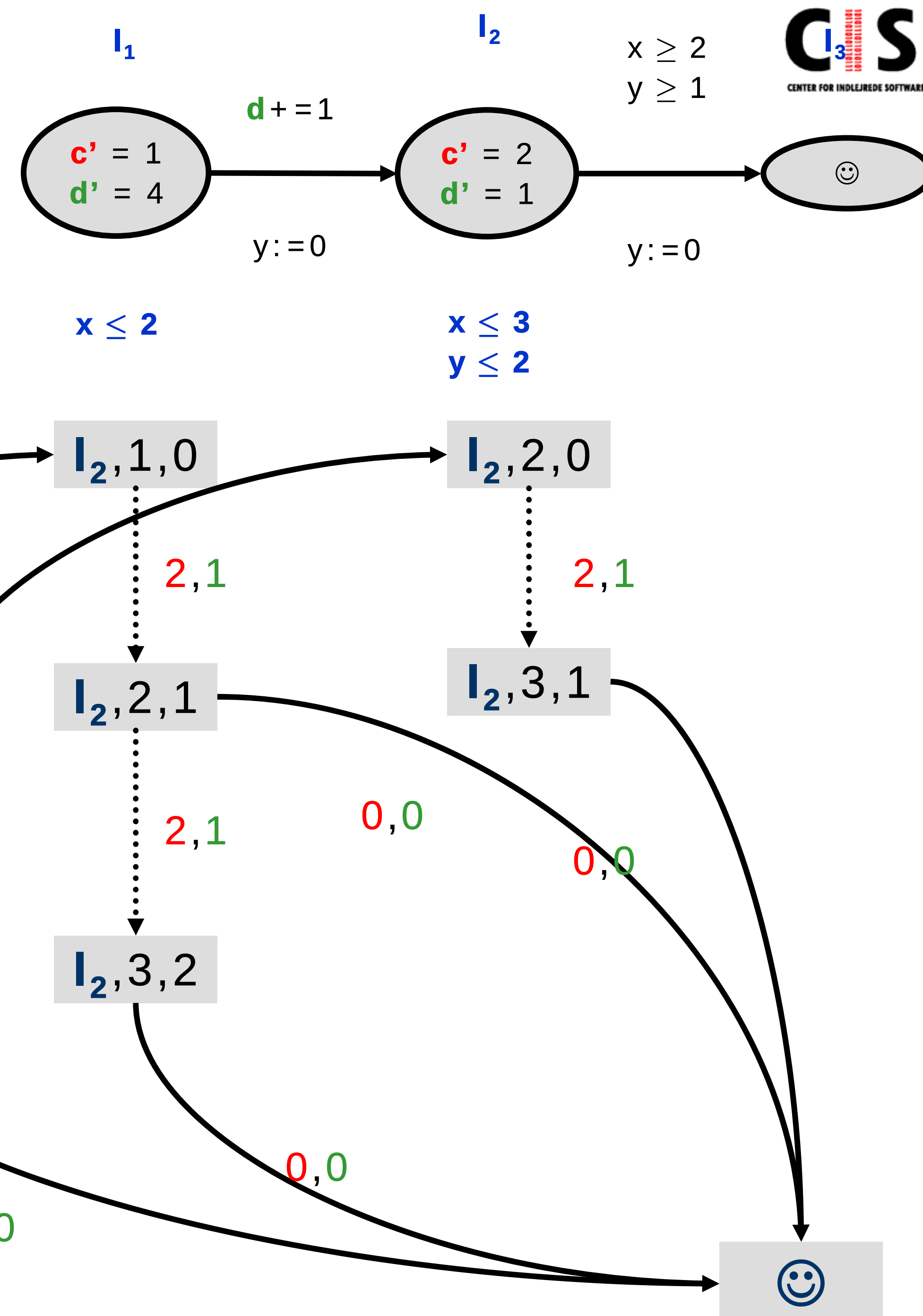
$c = 11/3$ →

wait 1/3 in l_1; goto l_2;

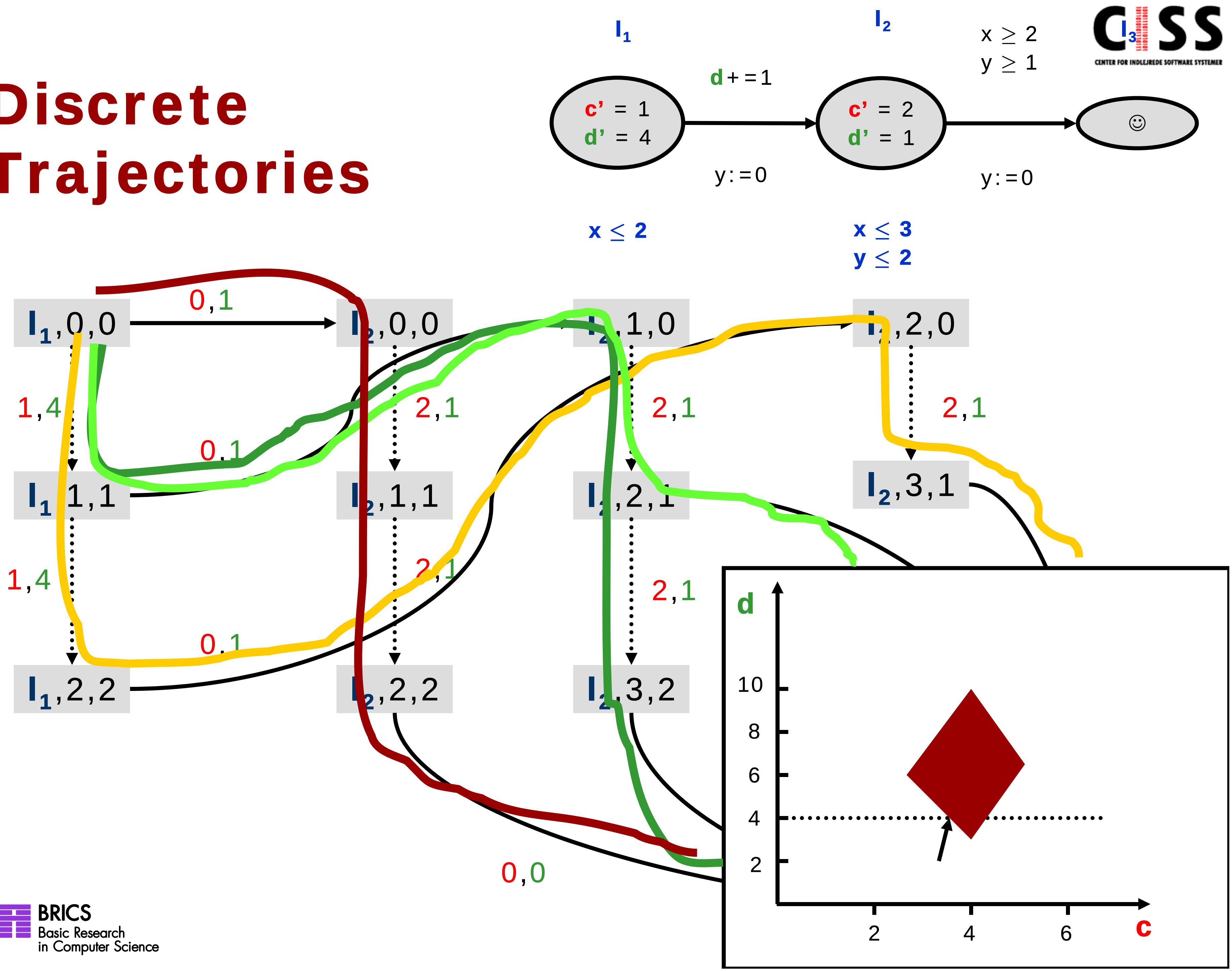
wait 5/3 in l_2; goto l_3

BRICS
Basic Research
in Computer Science

Discrete Trajectories



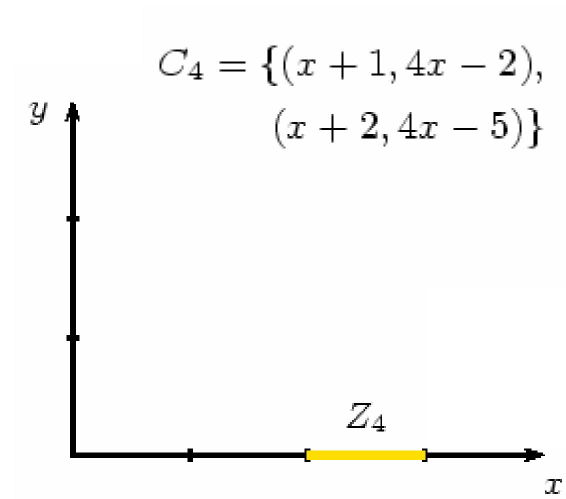
Discrete Trajectories



Dual Priced Zones

Definition

A dual-priced zone is a pair (Z, C), where Z is a zone and C is a set of pairs of affine cost functions $\{(e_1, d_1), \ldots, (e_n, d_n)\}$.

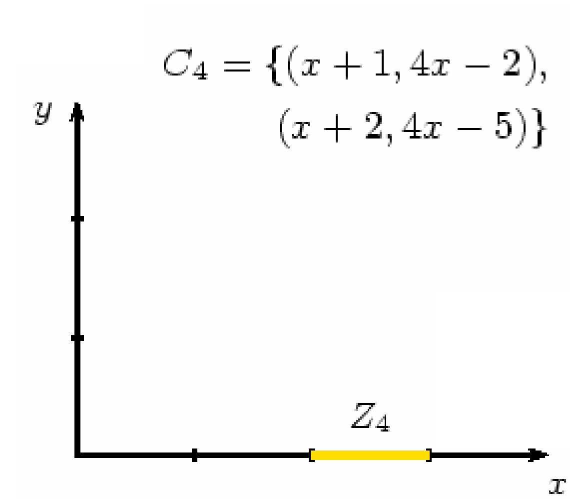


BRICS Basic Research in Computer Science

Dual Priced Zones

Definition

A dual-priced zone is a pair (Z, C), where Z is a zone and C is a set of pairs of affine cost functions $\{(e_1, d_1), \ldots, (e_n, d_n)\}$.

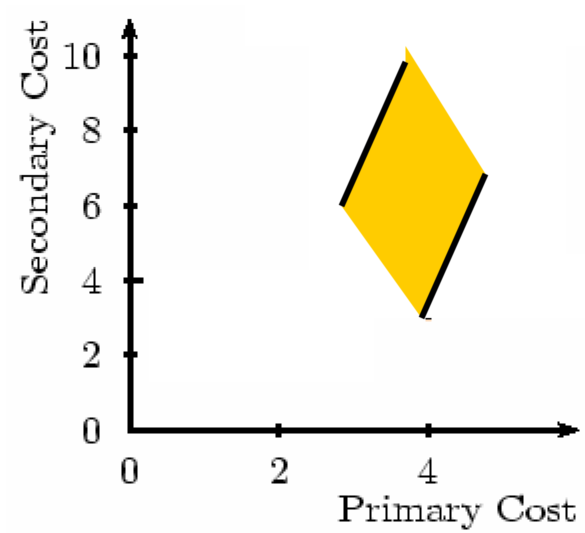


Definition

The dual-priced zone (Z, C) associates with a clock-valuation $u \in Z$ all cost pairs (c_1, c_2) such that:

$$(c_1, c_2) \in \lambda(C(u))$$

where $C(u) = \{(e_1(u), d_1(u)), \ldots, (e_n(u), d_n(u))\}$ and $\lambda()$ is convex-hull closure.

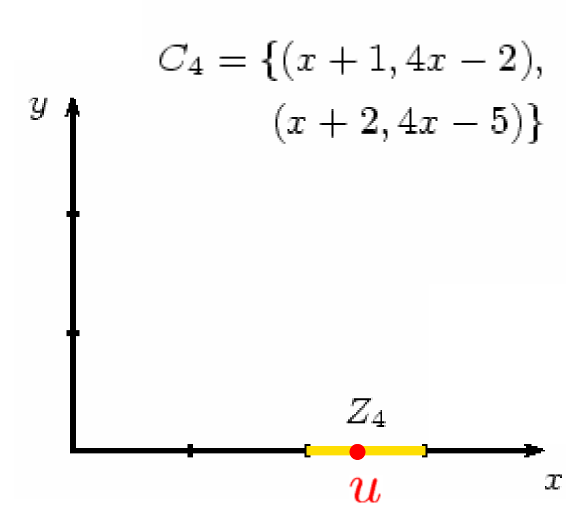


BRICS
Basic Research
in Computer Science

Dual Priced Zones

Definition

A dual-priced zone is a pair (Z, C), where Z is a zone and C is a set of pairs of affine cost functions $\{(e_1, d_1), \ldots, (e_n, d_n)\}$.

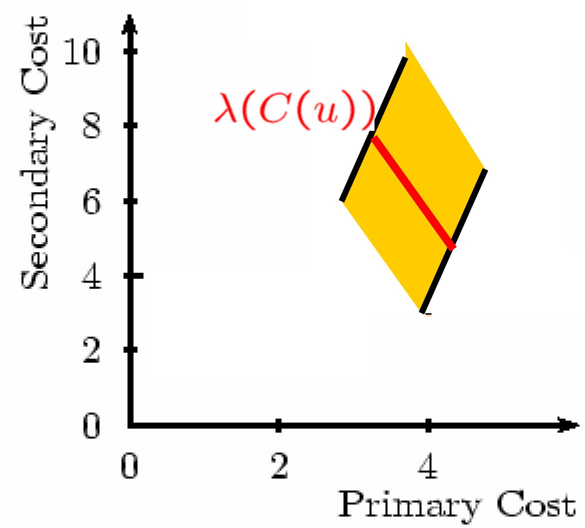


Definition

The dual-priced zone (Z, C) associates with a clock-valuation $u \in Z$ all cost pairs (c_1, c_2) such that:

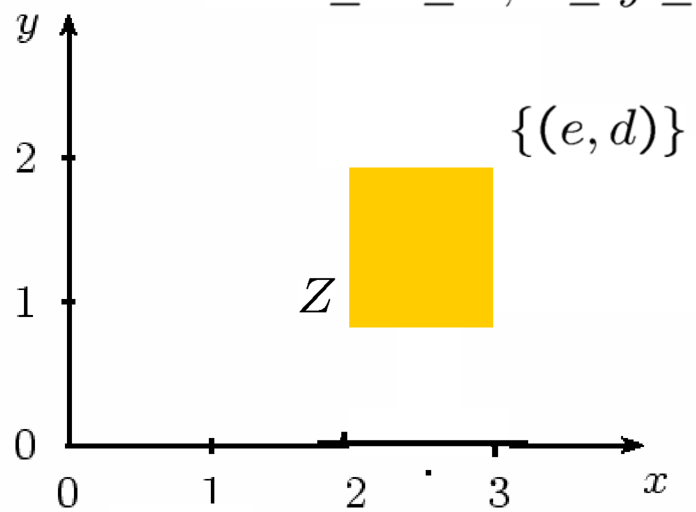
$$(c_1, c_2) \in \lambda(C(u))$$

where $C(u) = \{(e_1(u), d_1(u)), \ldots, (e_n(u), d_n(u))\}$ and $\lambda()$ is convex-hull closure.



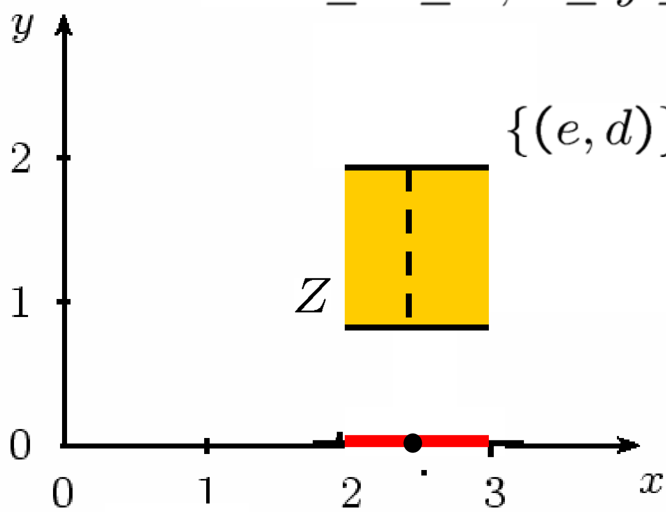
Reset

$$e = x + y$$
$$d = 4x - 3y + 1$$
$$Z : 2 \le x \le 3, 1 \le y \le 2$$



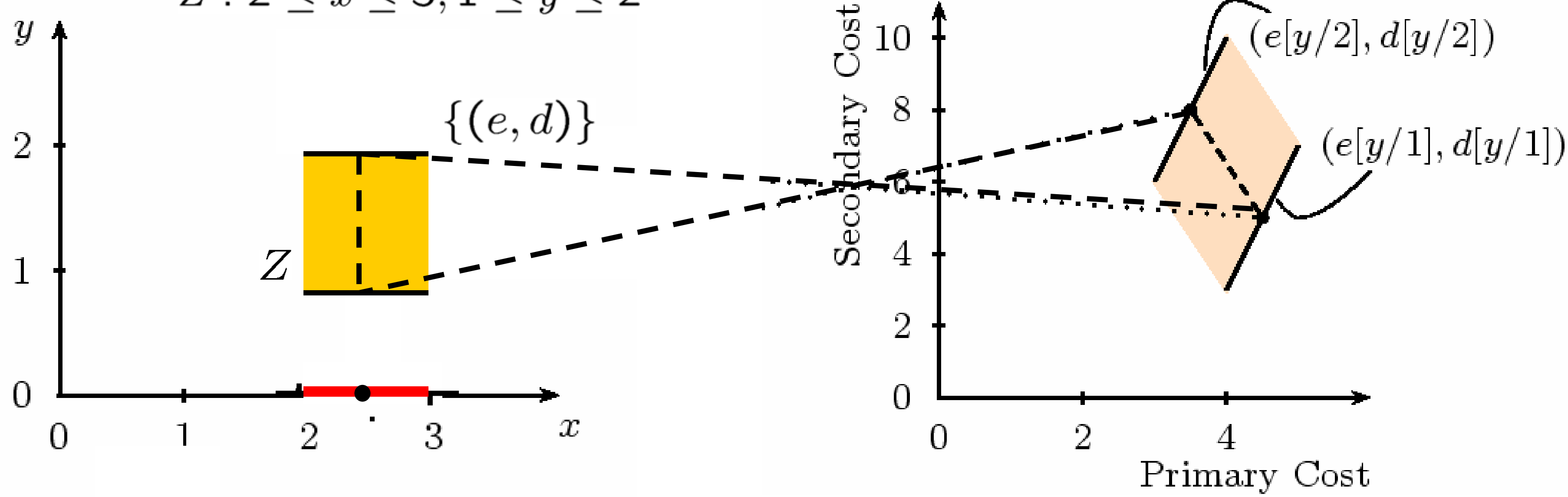
Reset

$$e = x + y$$
$$d = 4x - 3y + 1$$
$$Z : 2 \le x \le 3, 1 \le y \le 2$$



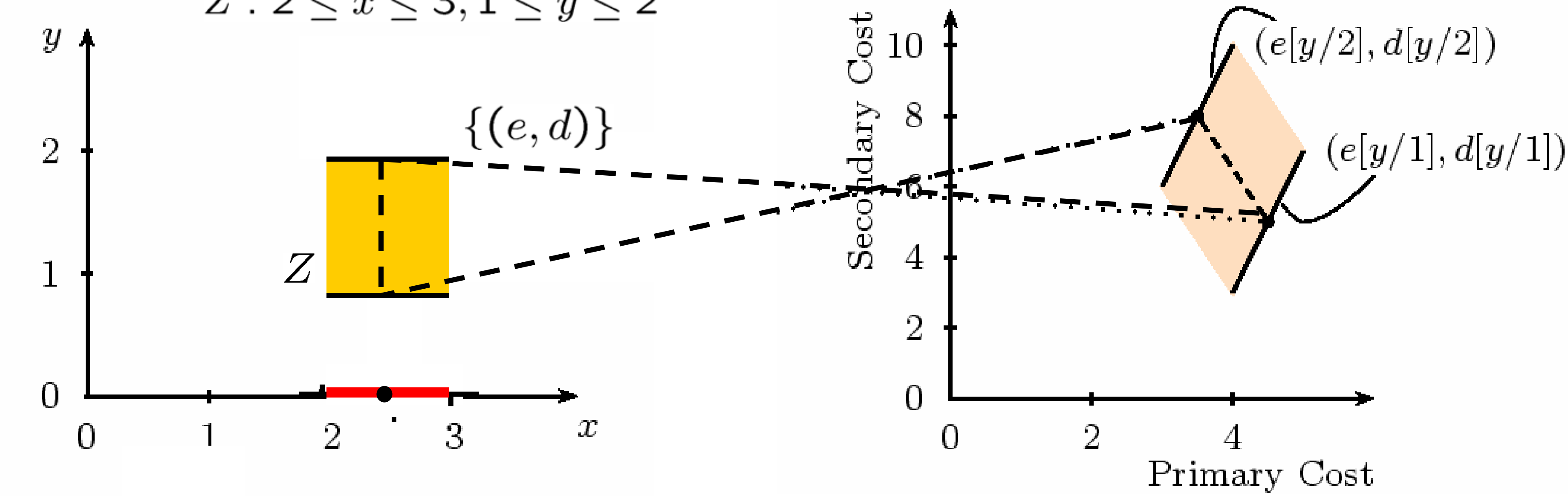
Reset

$$e = x + y$$
$$d = 4x - 3y + 1$$
$$Z : 2 \leq x \leq 3, 1 \leq y \leq 2$$



Reset

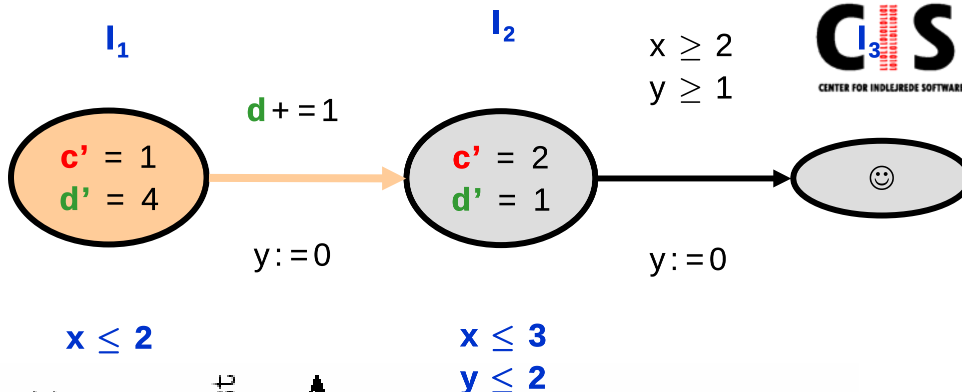
$$e = x + y$$
$$d = 4x - 3y + 1$$
$$Z : 2 \leq x \leq 3, 1 \leq y \leq 2$$



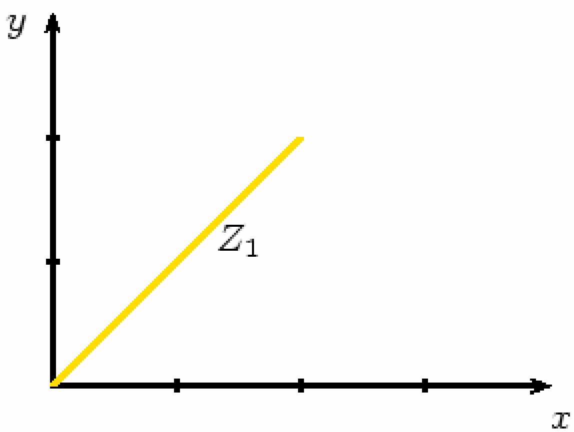
$$C = \{(x+1, 4x-2),$$
$$(x+2, 4x-5)\}$$

BRICS
Basic Research
in Computer Science

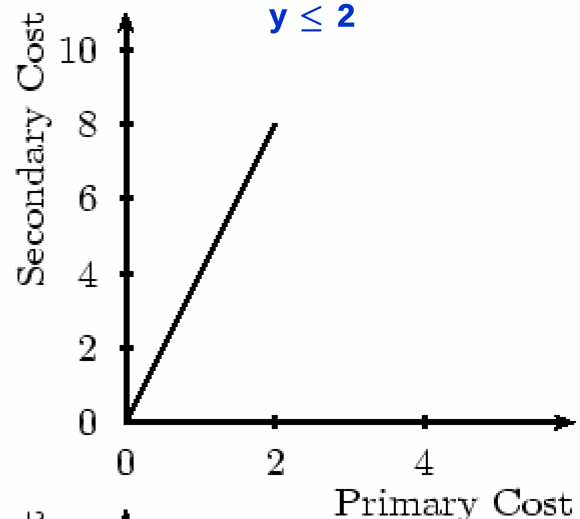
Exploration



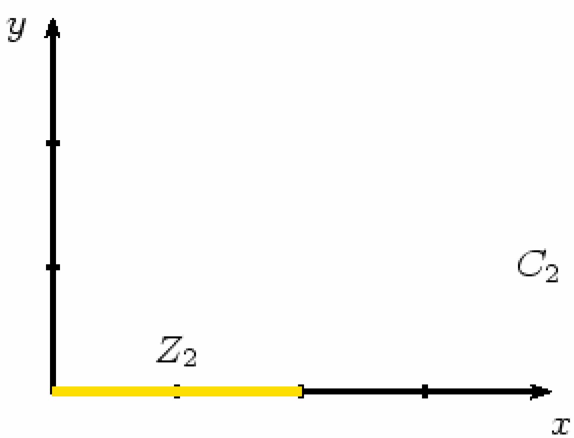
(i)



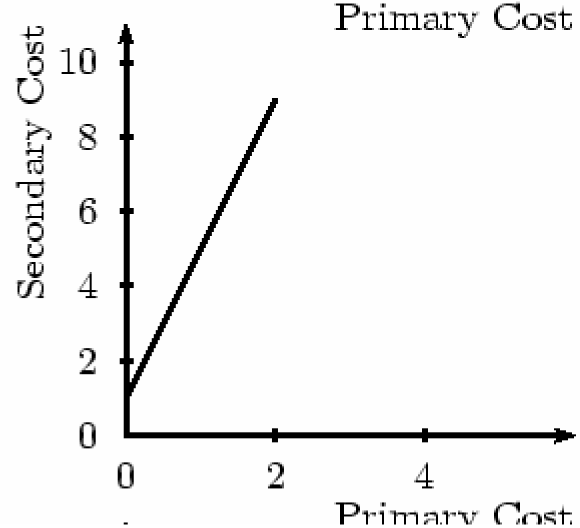
$$C_1 = \{(x, 4x)\}$$



(ii)

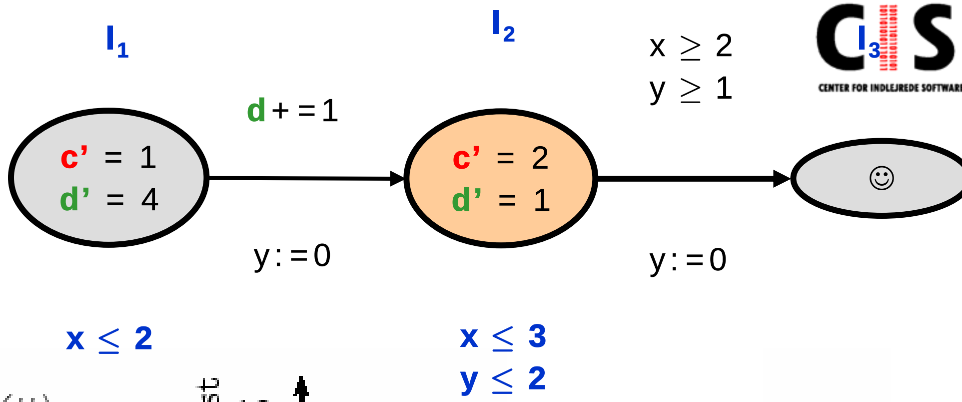


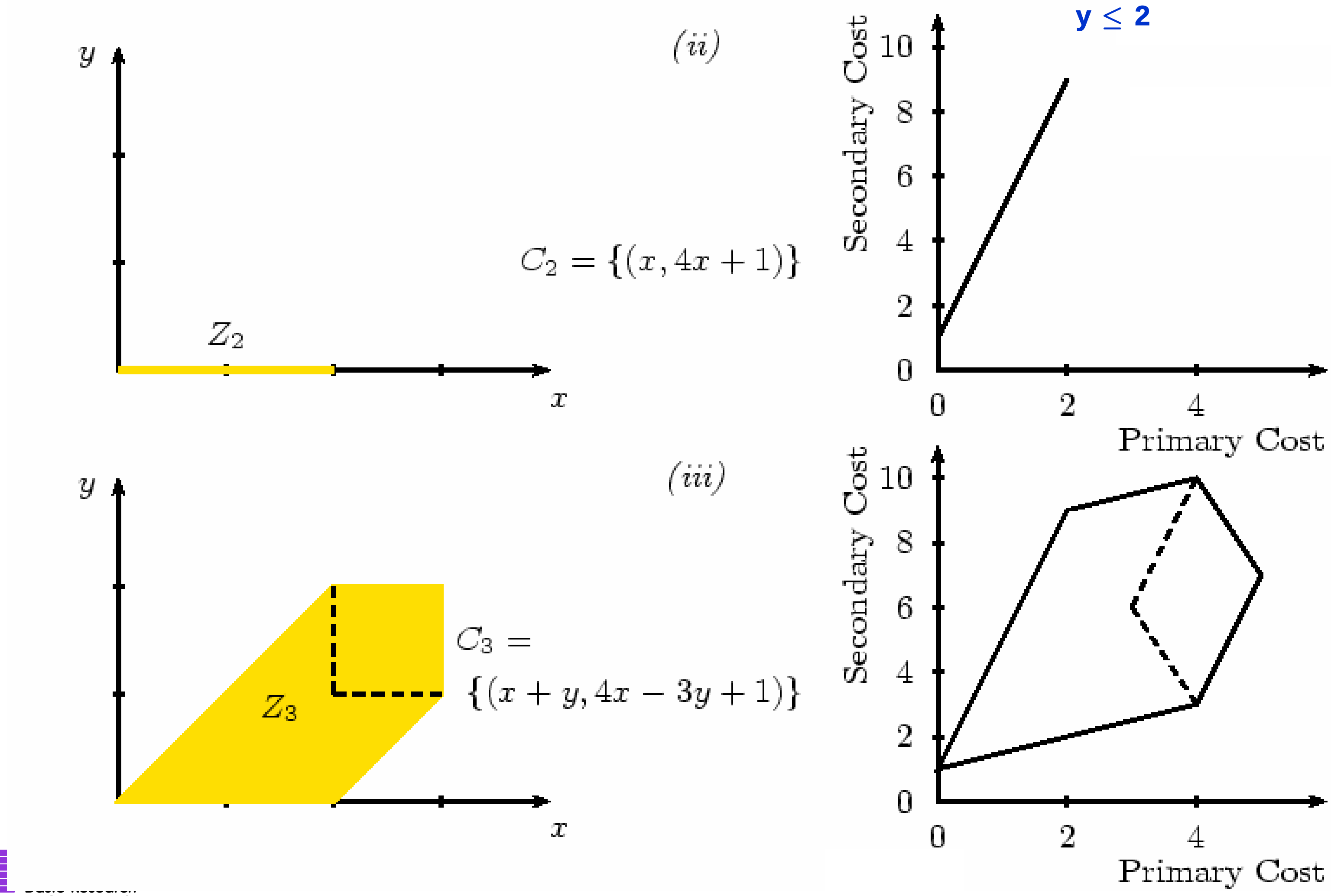
$$C_2 = \{(x, 4x+1)\}$$



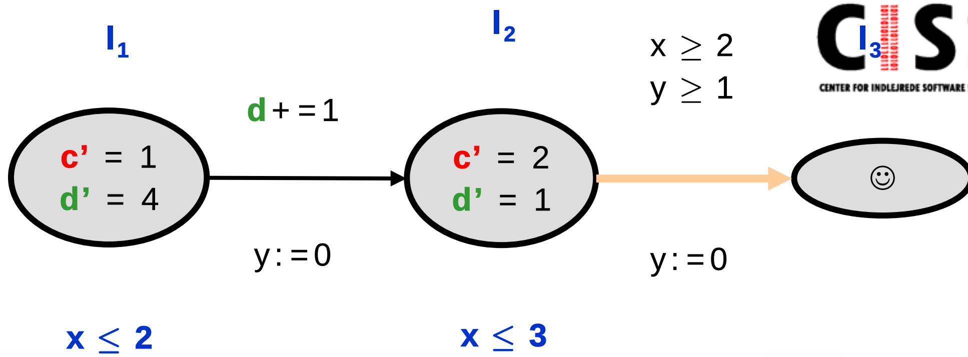
BRICS Basic Research in Computer Science

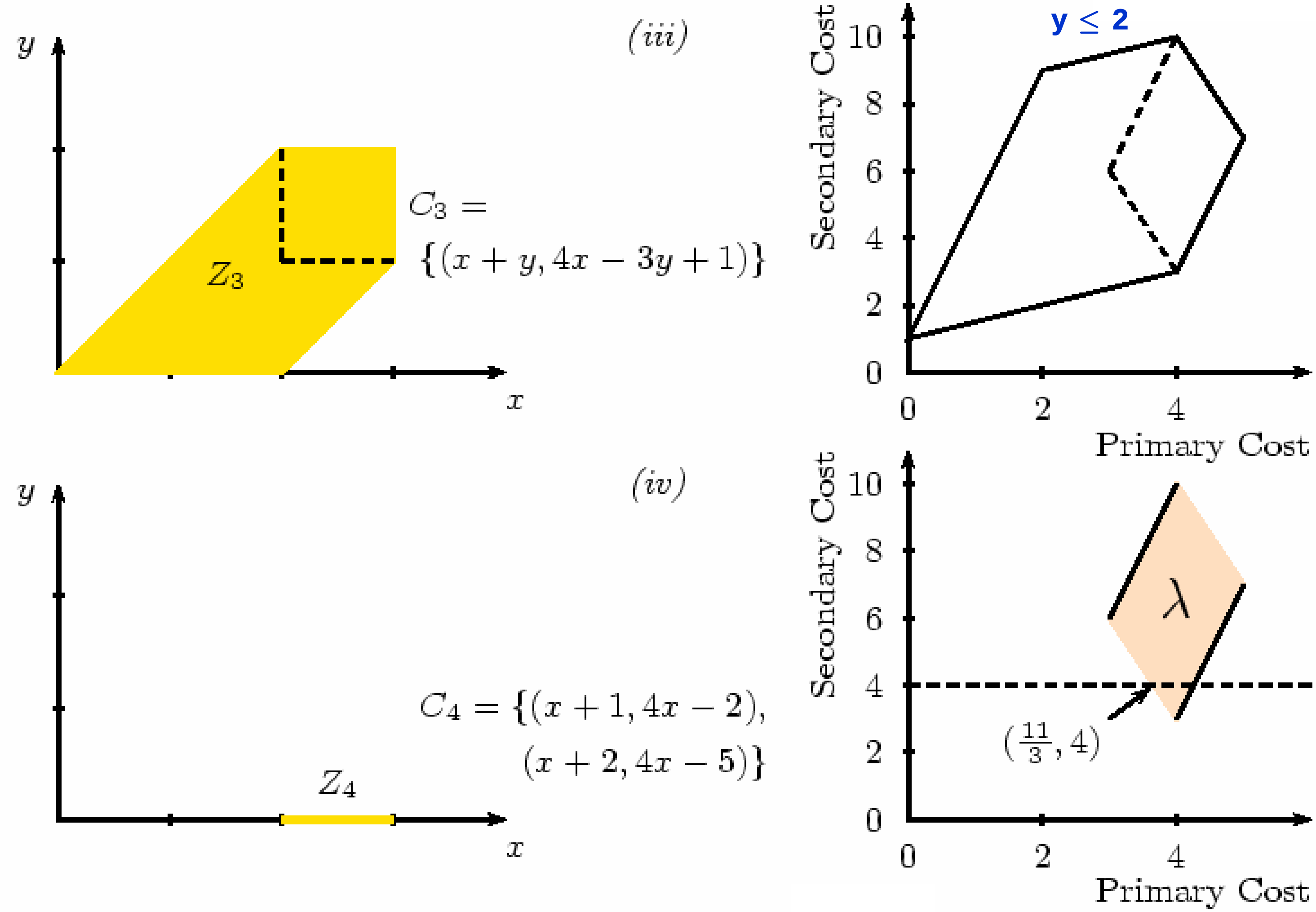
Exploration





Exploration





Termination

Definition

Let (Z, C) and (Z', C') be two dual-priced zones.

Then $(Z, C) \sqsubseteq (Z', C')$ iff

i) $Z' \subseteq Z$, and

ii) for any $(e', d') \in C'$ there exists $(e, d) \in C$

s.t. $e \leq_{Z'} e'$ and $d \leq_{Z'} d'$

Theor

If $(Z,$

$\forall (c'_1,$ … c'_2

THEOREM

Optimal conditional reachability for multi-priced TA is computable.

Theor

$\sqsubseteq$ is a well-quasi ordering.

CISS CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

BRICS Basic Research in Computer Science

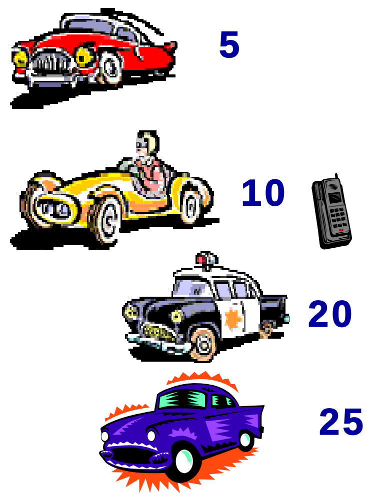
Optimal Infinite Scheduling

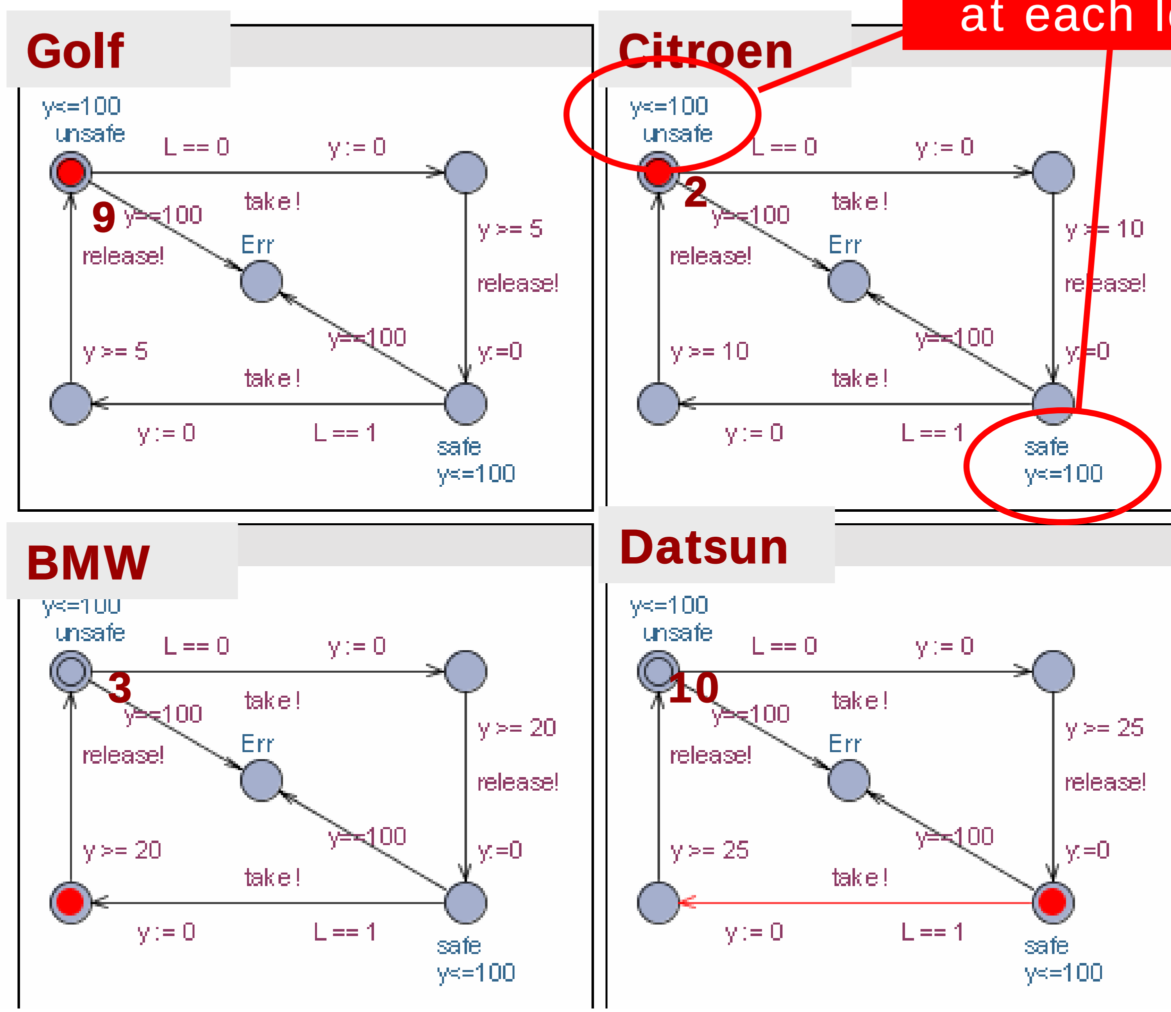
with Ed Brinksma
Patricia Bouyer
Arne Skou

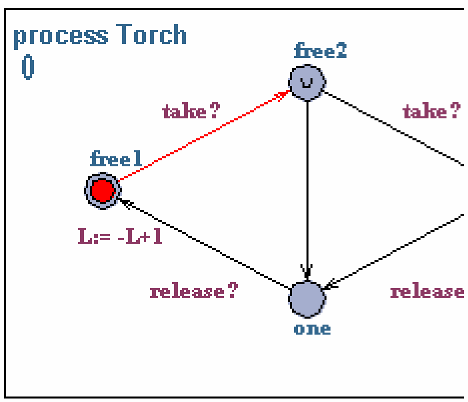
CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

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

maximal 100 min. at each location

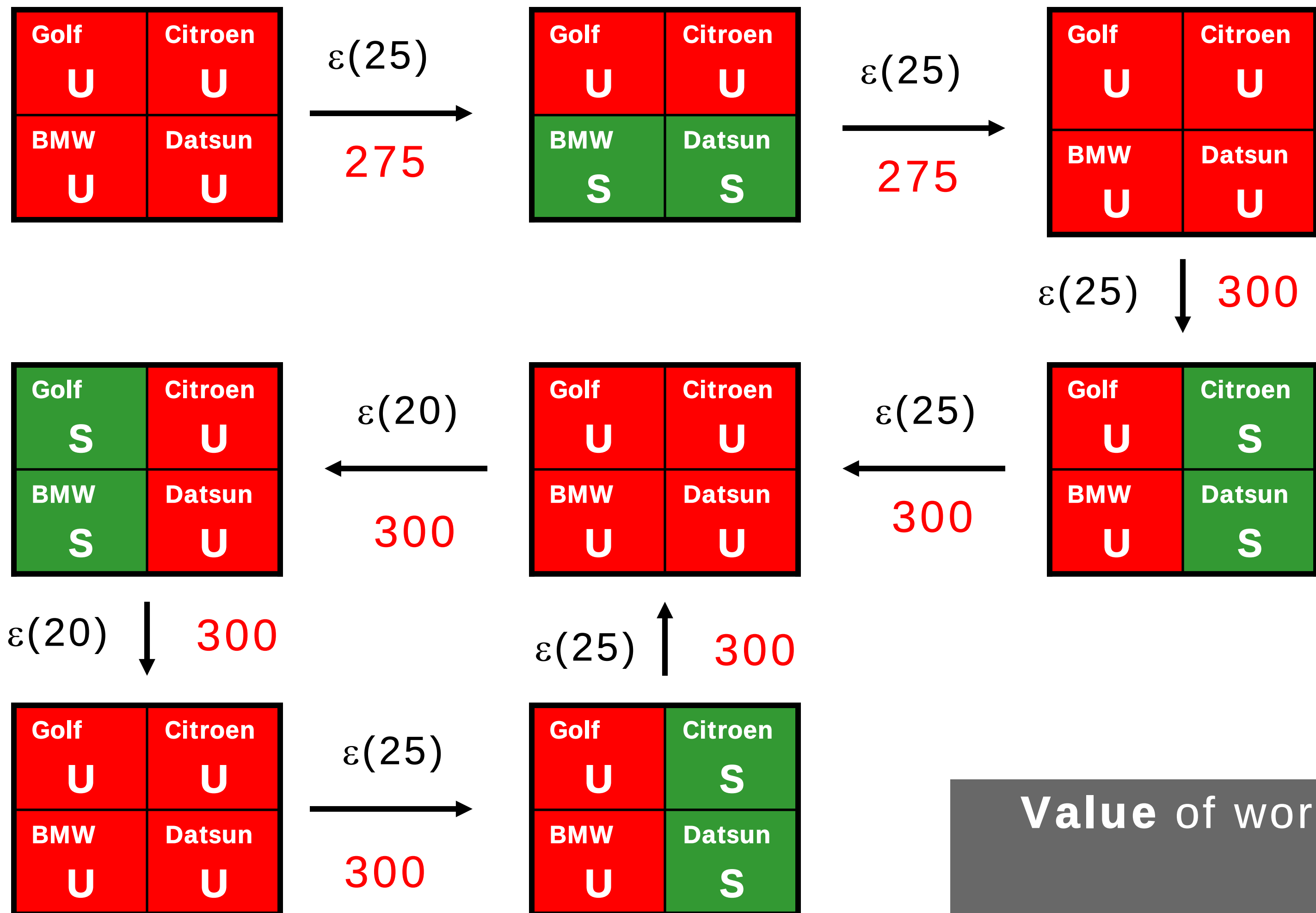






BRICS
Basic Research
in Computer Science

Workplan I



Value of workplan:

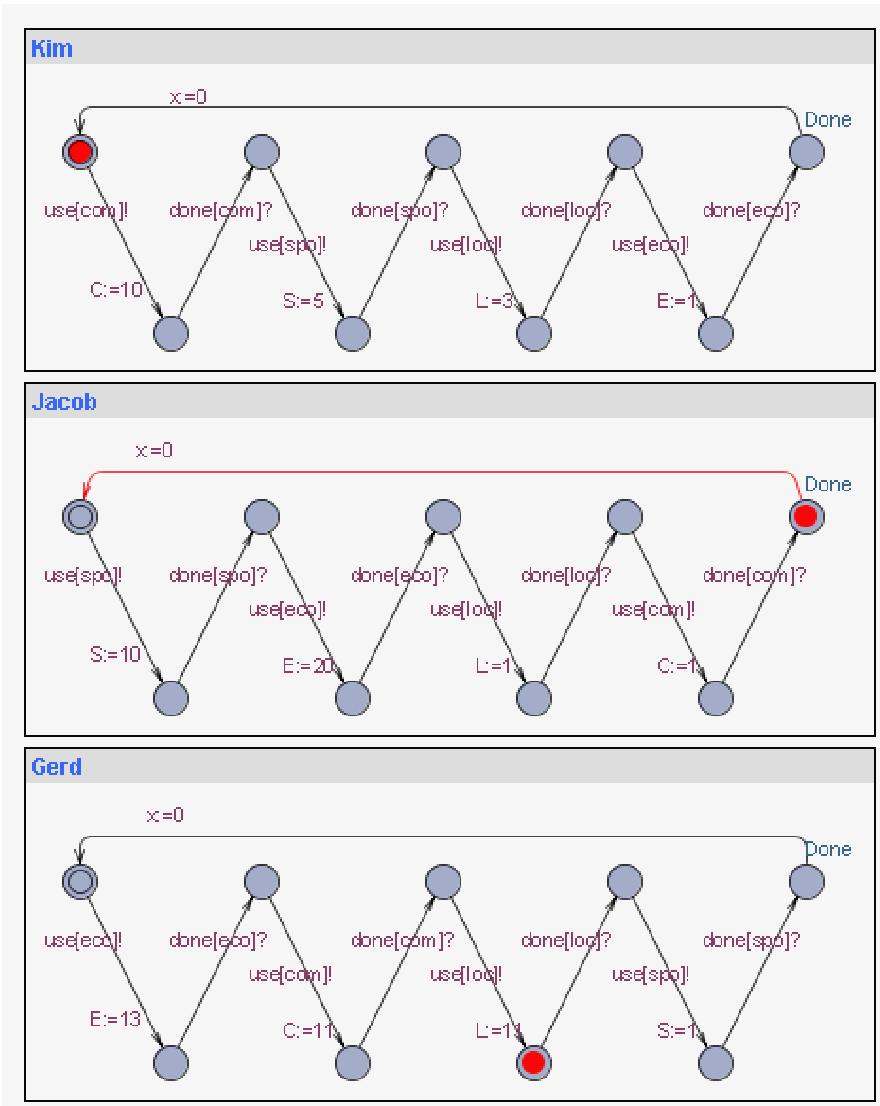
(4 x **300**) / 90 = **13.33**

Workplan II

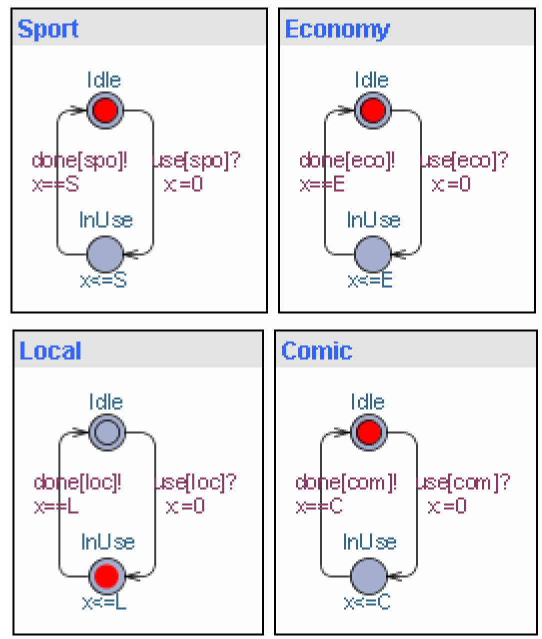
Value of workplan:

560 / 100 = **5.6**

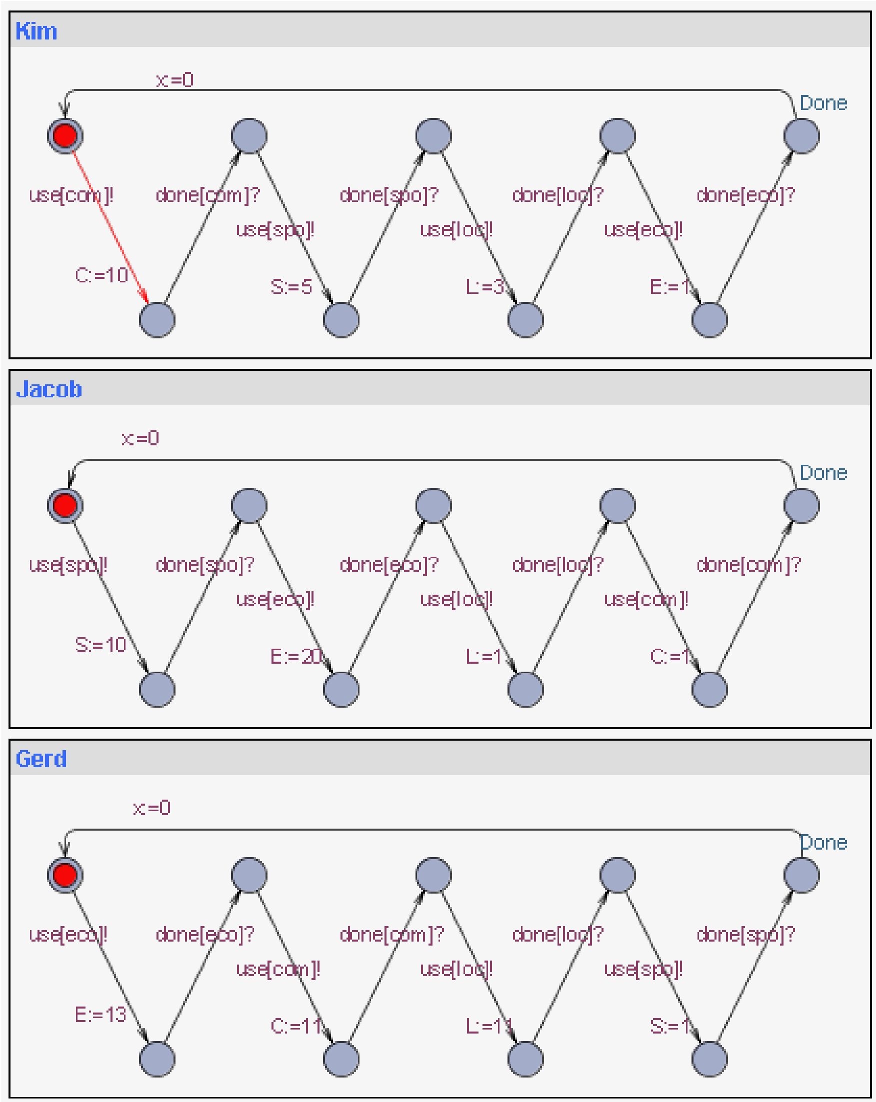
Infinite Scheduling



E[] (Kim.x $\leq$ 100 and
Jacob.x $\leq$ 90 and
Gerd.x $\leq$ 100)

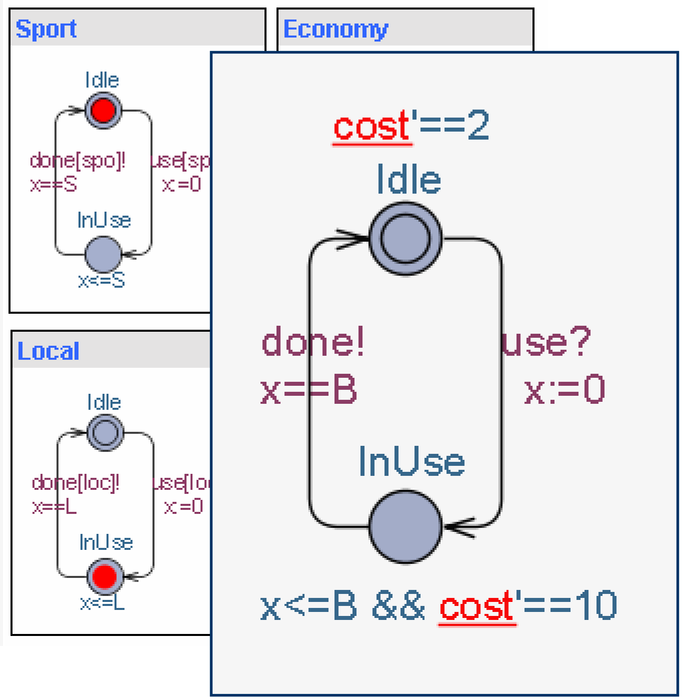


Infinite Optimal Scheduling



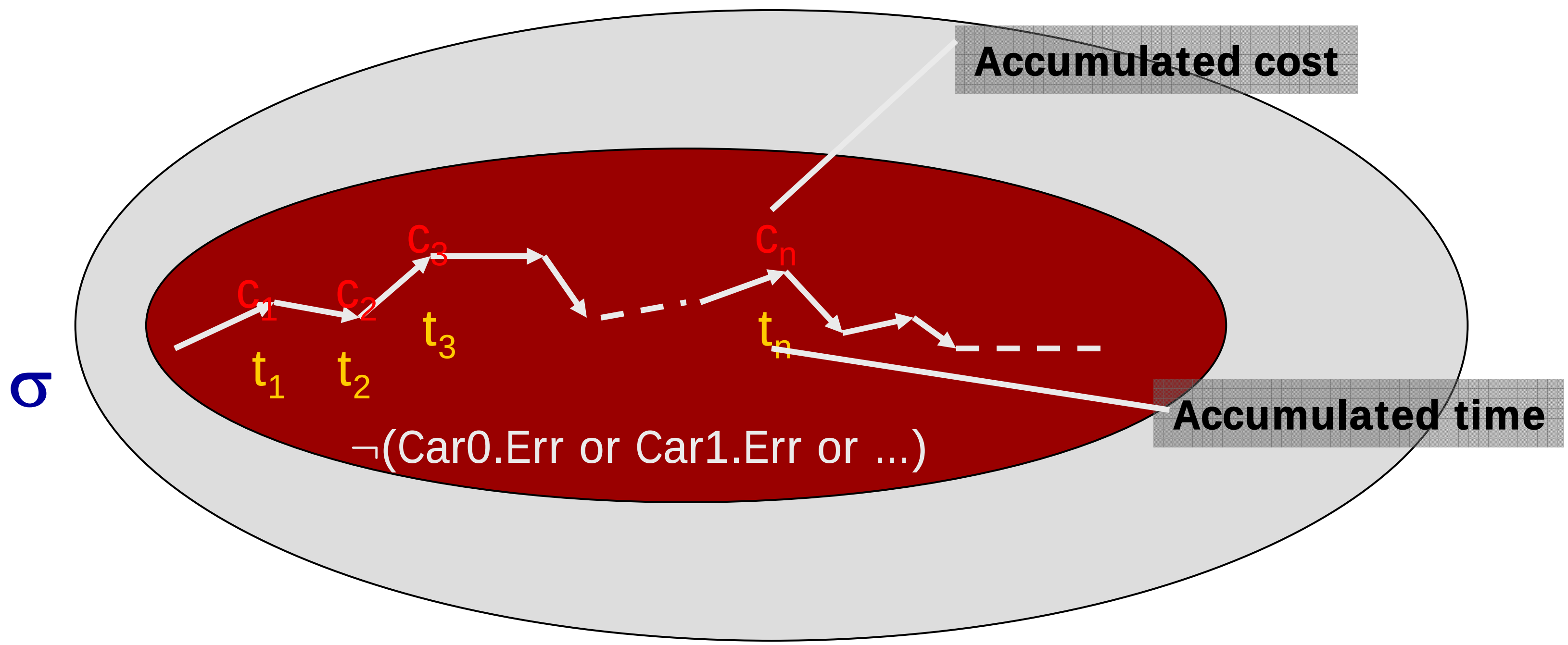
E[] (Kim.x $\leq$ 100 and
Jacob.x $\leq$ 90 and
Gerd.x $\leq$ 100)

+ most minimal limit of cost / time



BRICS
Basic Research
in Computer Science

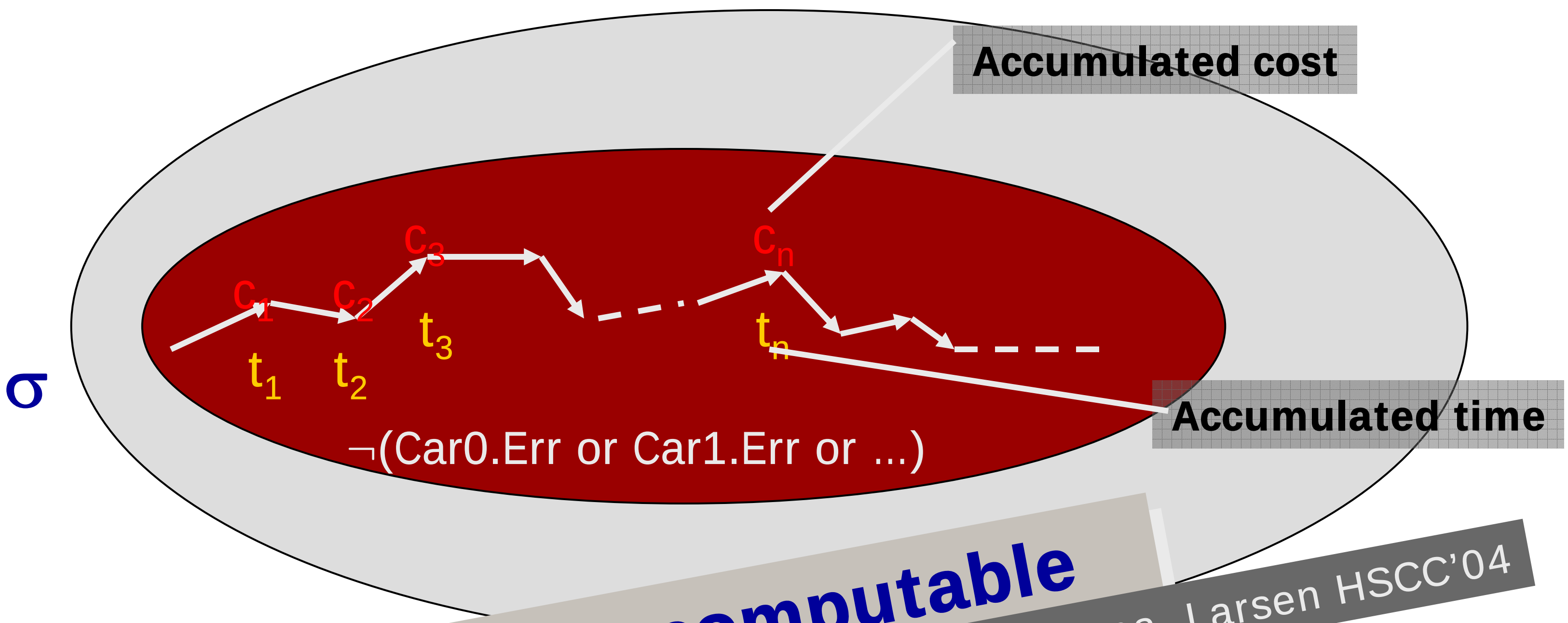
Cost Optimal Scheduling = *Optimal Infinite Path*



Value of path σ: $\quad val(\sigma) = \lim_{n\to\infty} c_n/t_n$

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

Cost Optimal Scheduling = *Optimal Infinite Path*



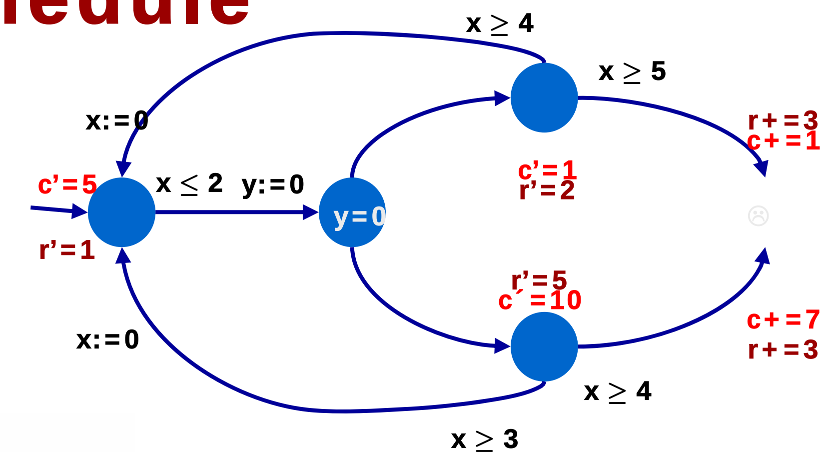
THEOREM: σ^* is computable

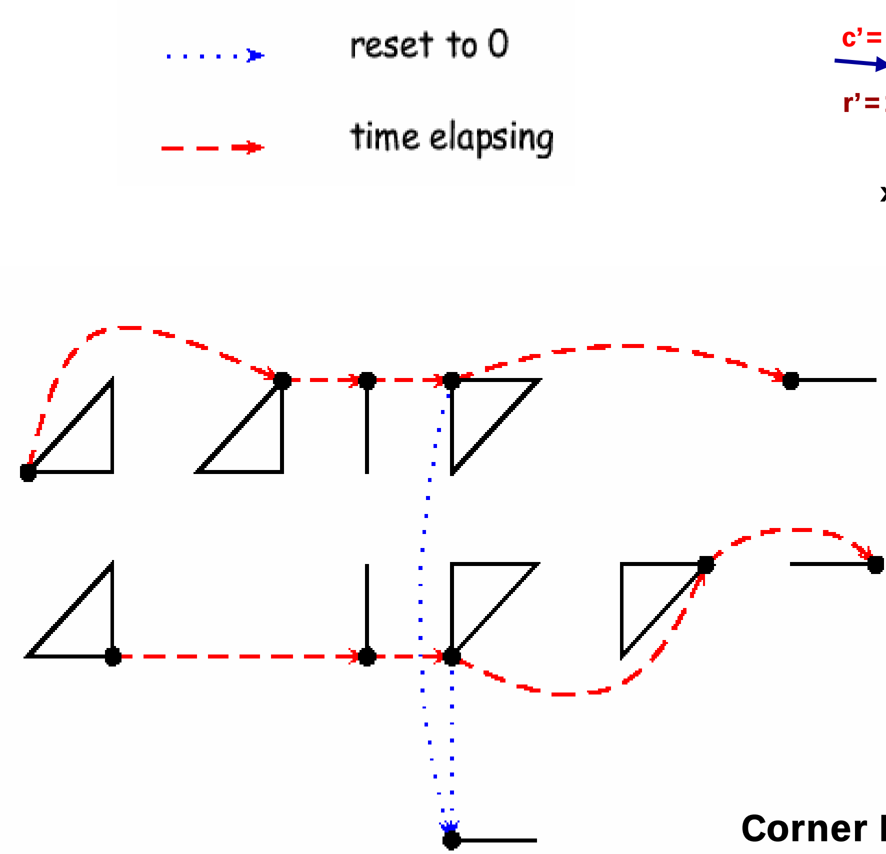
Bouyer, Brinksma, Larsen HSCC'04

… path σ: $val(\sigma) = \lim_{n\to\infty} c_n/t_n$

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

Optimal Infinite Schedule



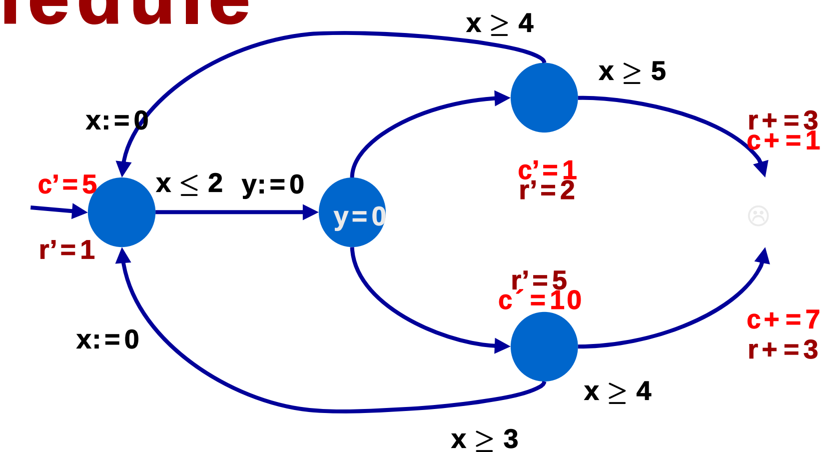


Corner Point Automaton

Aim: prove that $\mu^*(\mathcal{A}_{cp}) = \mu^*(\mathcal{A})$

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Optimal Infinite Schedule



Finite Behaviours

Lemma. Let Z be a zone and f be a function

$$f : (t_1, \ldots, t_n) \mapsto \frac{\sum_{i=1}^{n} c_i t_i + c}{\sum_{i=1}^{n} r_i t_i + r}$$

well-defined on $\overline{Z}$. Then $\inf_Z f$ is obtained on the border of $\overline{Z}$ with integer coordinates.

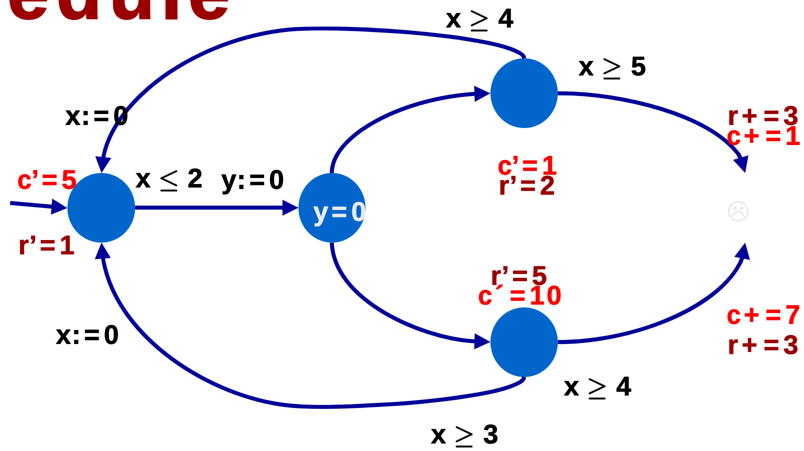
→ for any finite path π in $\mathcal{A}$, there exists a path Π in $\mathcal{A}_{cp}$ such that

$$\text{ratio}(\Pi) \leq \text{ratio}(\pi)$$

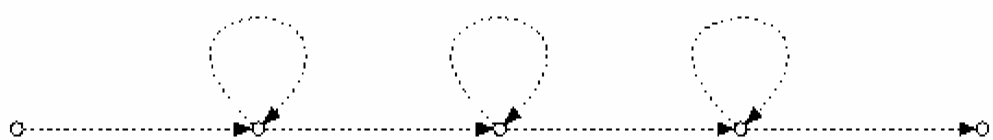
BRICS
Basic Research
in Computer Science

! Optimal behaviours are NOT prefixed closed

Optimal Infinite Schedule



✔ **Infinite behaviours:** decompose each sufficiently long projection into cycles



The linear part will be negligible when the path is long enough.

➜ the optimal cycle of $\mathcal{A}_{cp}$ is better than any infinite path of $\mathcal{A}$

Dynamic Voltage Scaling

BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Performance vs Ressource-Efficiency ?

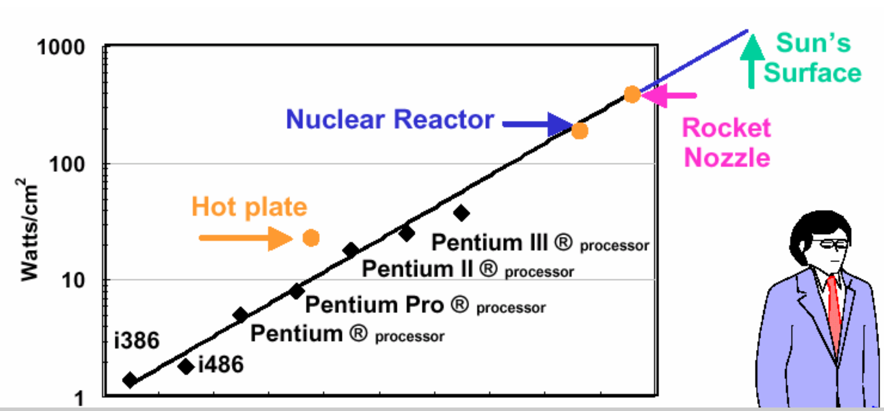
- Consumer constantly demand
  better functionality, flexibility, availability, ...
- .. increase in resources needed:
  - Time
  - Energy
  - Memory
  - Bandwidth
  - ..
- Application of **CUPPAAL** to modeling, analysis and synthesis of resource-efficient schedules for real-time systems.

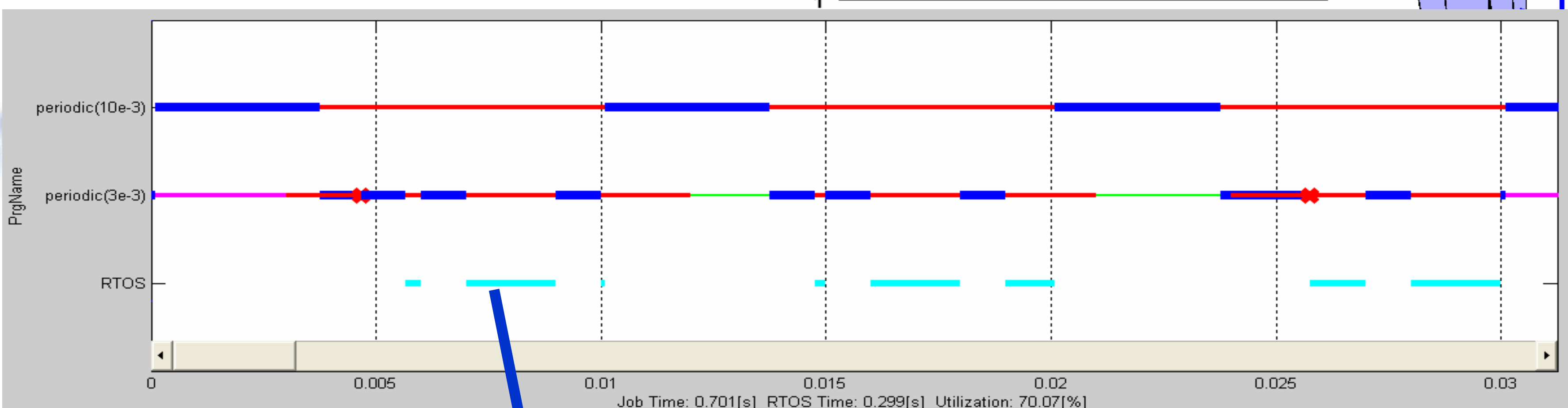
BRICS
Basic Research
in Computer Science

CISS
CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

Power Management

Dynamic Voltage Scaling





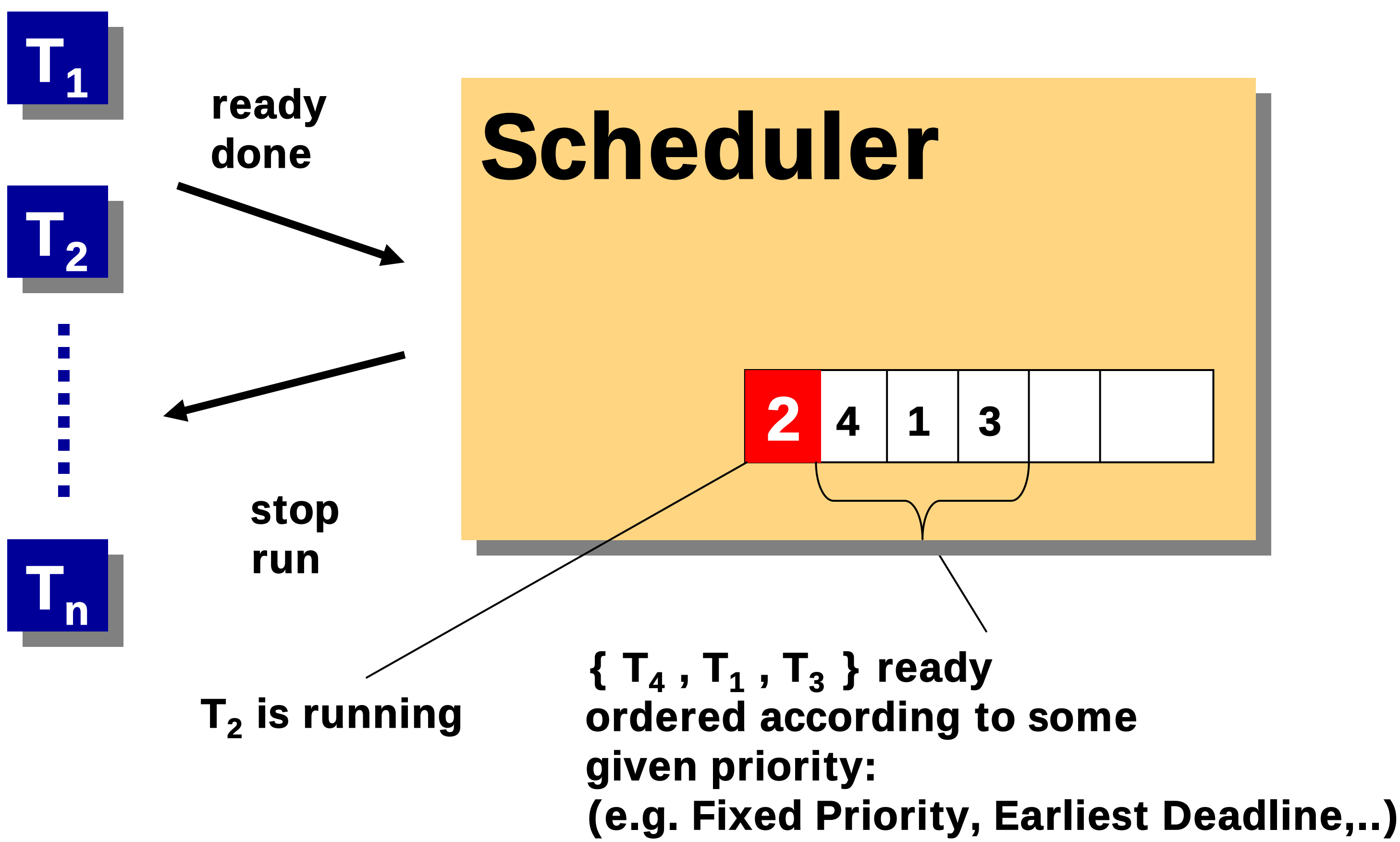
Dynamically lower voltage supply (frequency) to utilize free CPU time

BRICS
Basic Research
in Computer Science

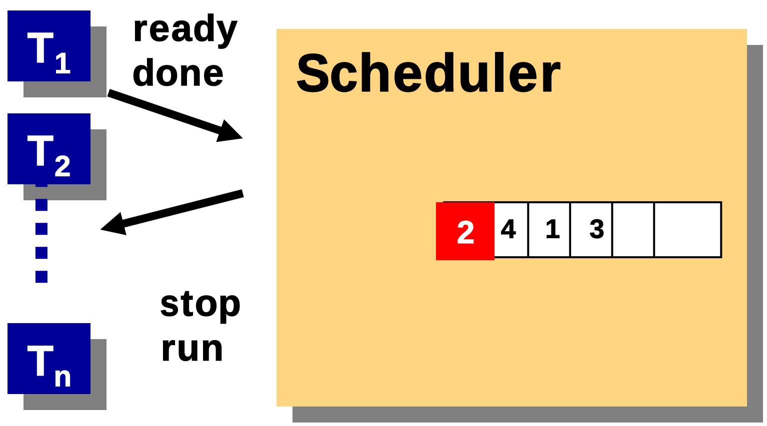
Task Scheduling

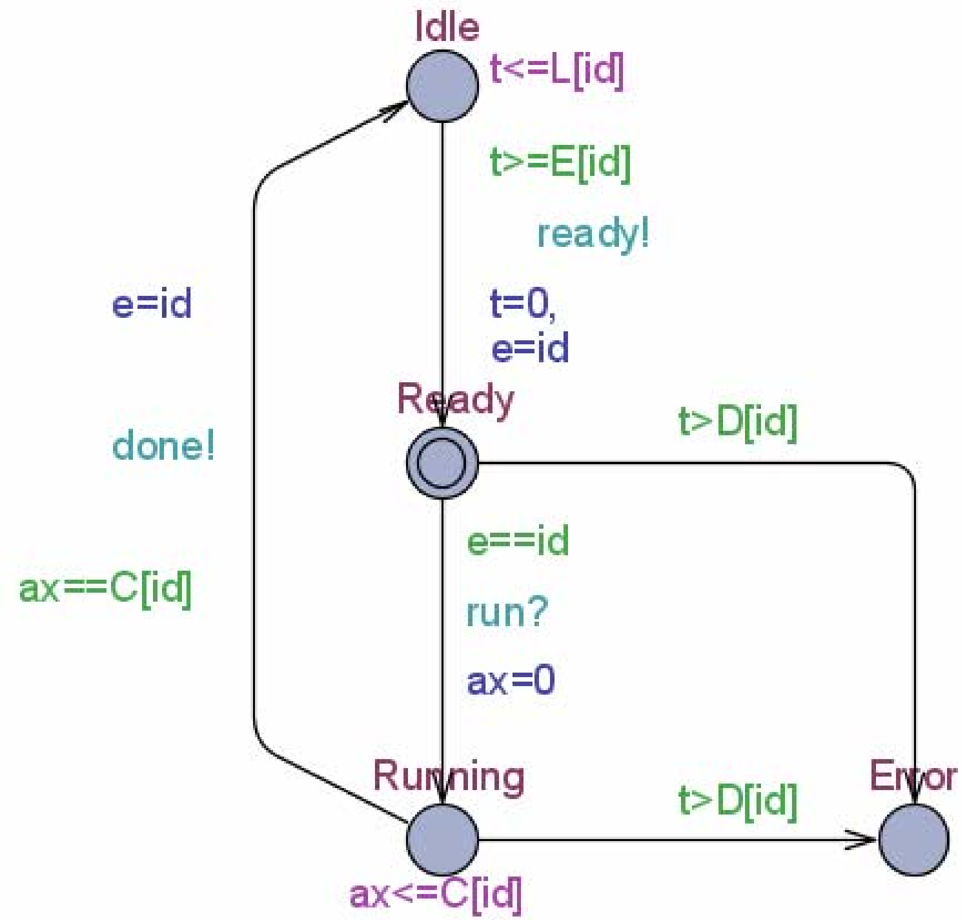
utilization of CPU

P(i), [E(i), L(i)], .. : period or earliest / latest arrival or .. for T_i
C(i): execution time for T_i
D(i): deadline for T_i

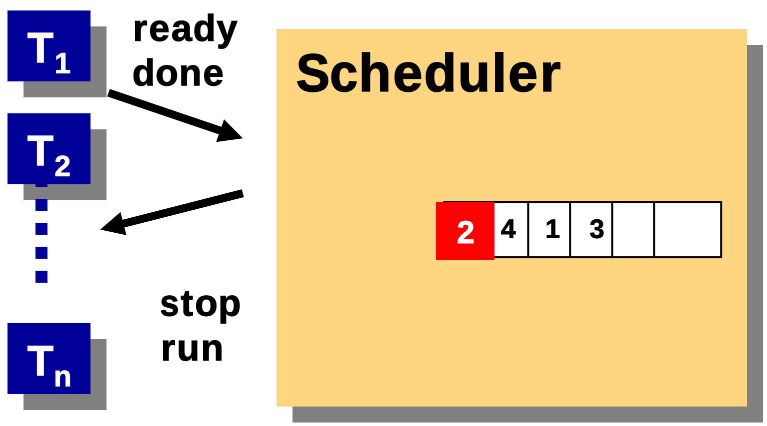


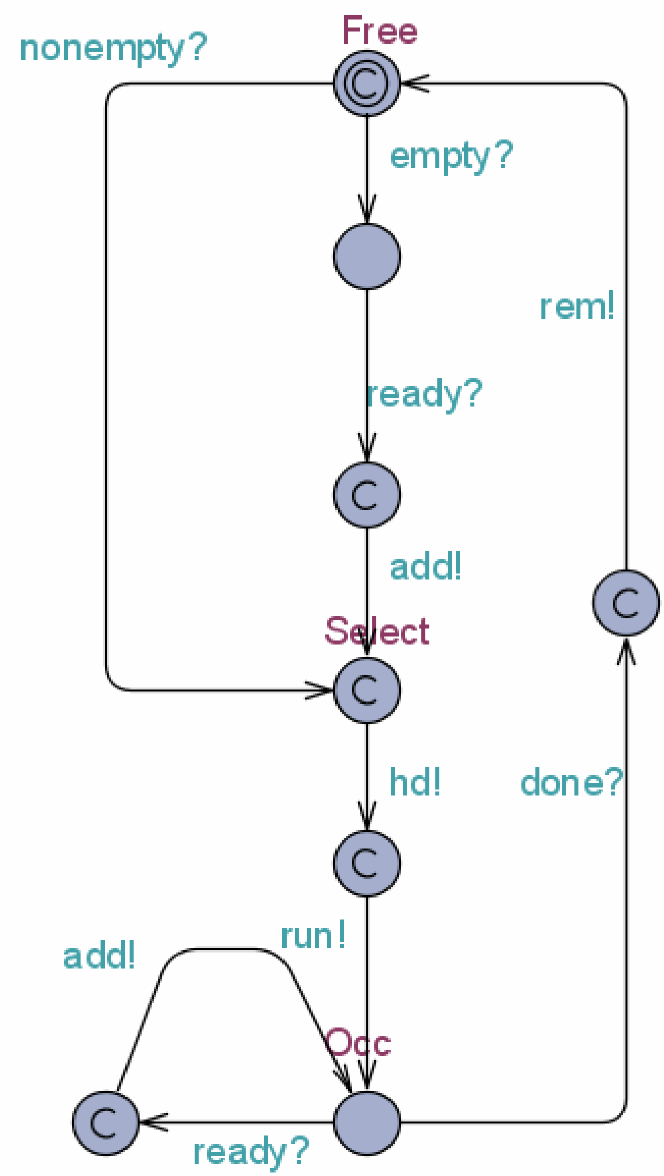
Modeling Task



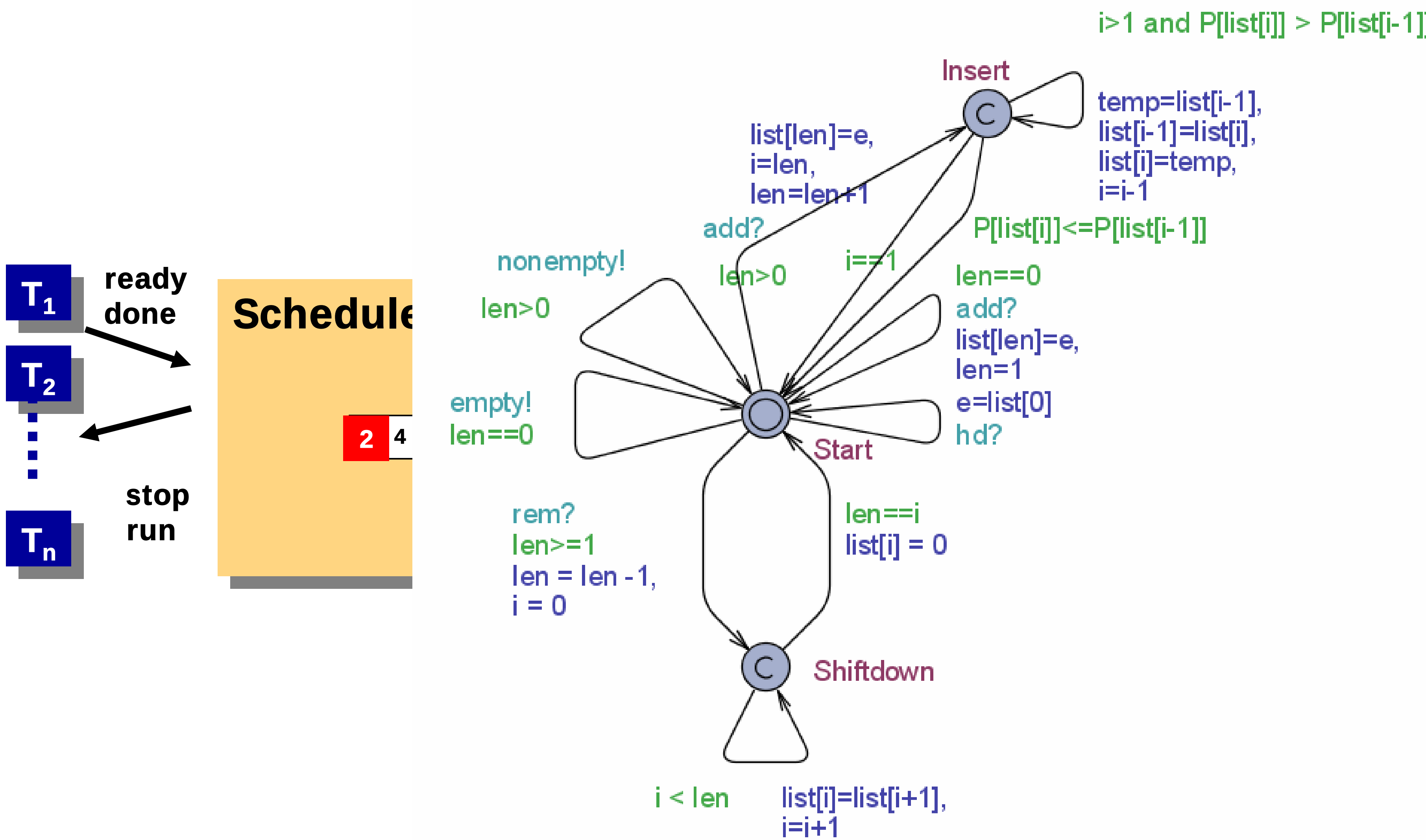


Modeling Scheduler



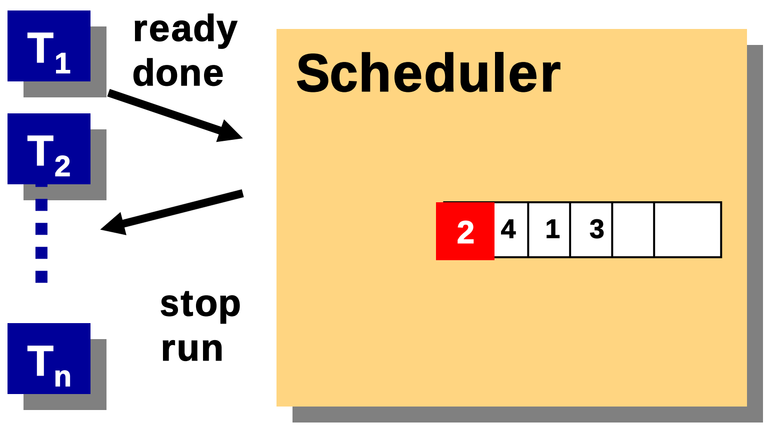


Modeling Queue



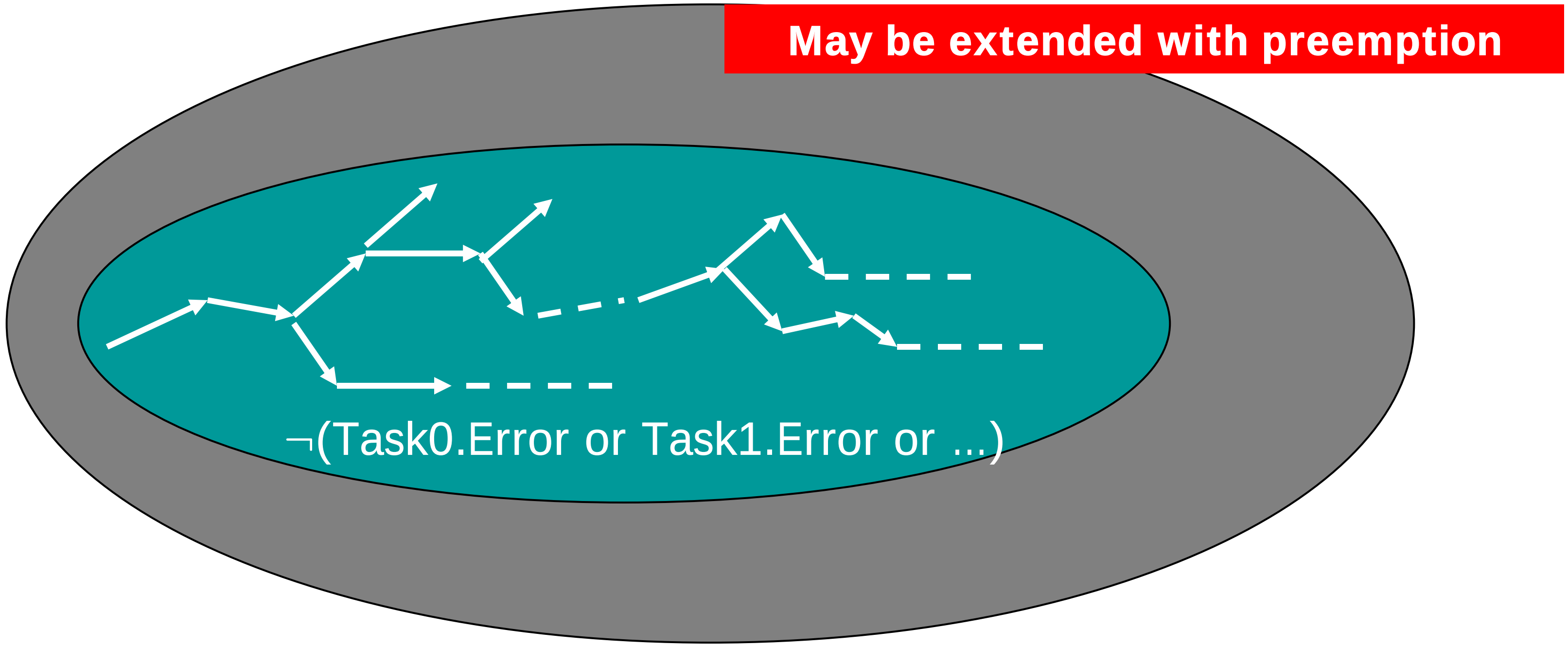
Modeling Queue

In UPPAAL 4.0
User Defined Function




```
void add(id_t element)
{
int i=len;
int temp=0;
if(len==0)
{       list[len]=e;
len=1;
}
    else
{       list[len]=e;
        i=len;
        len=len+1;
        while (i>1 && P[list[i]]> P[list[i-1]])
        {       temp=list[i-1];
                list[i-1]=list[i];
                list[i]=temp;
                i=i-1;
        }
}
}
```

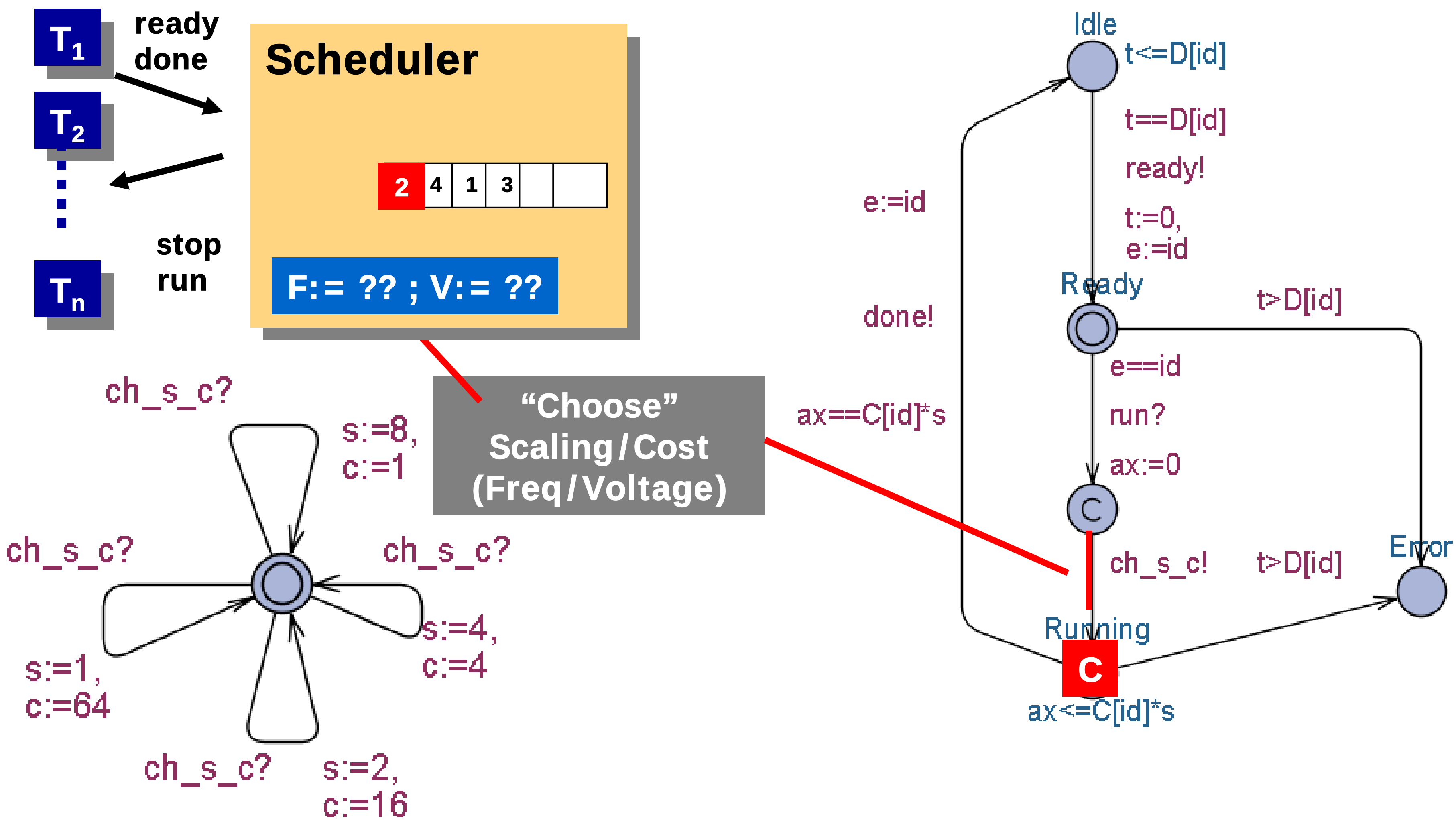
Schedulability = Safety Property



A□ ¬(Task0.Error or Task1.Error or ...)

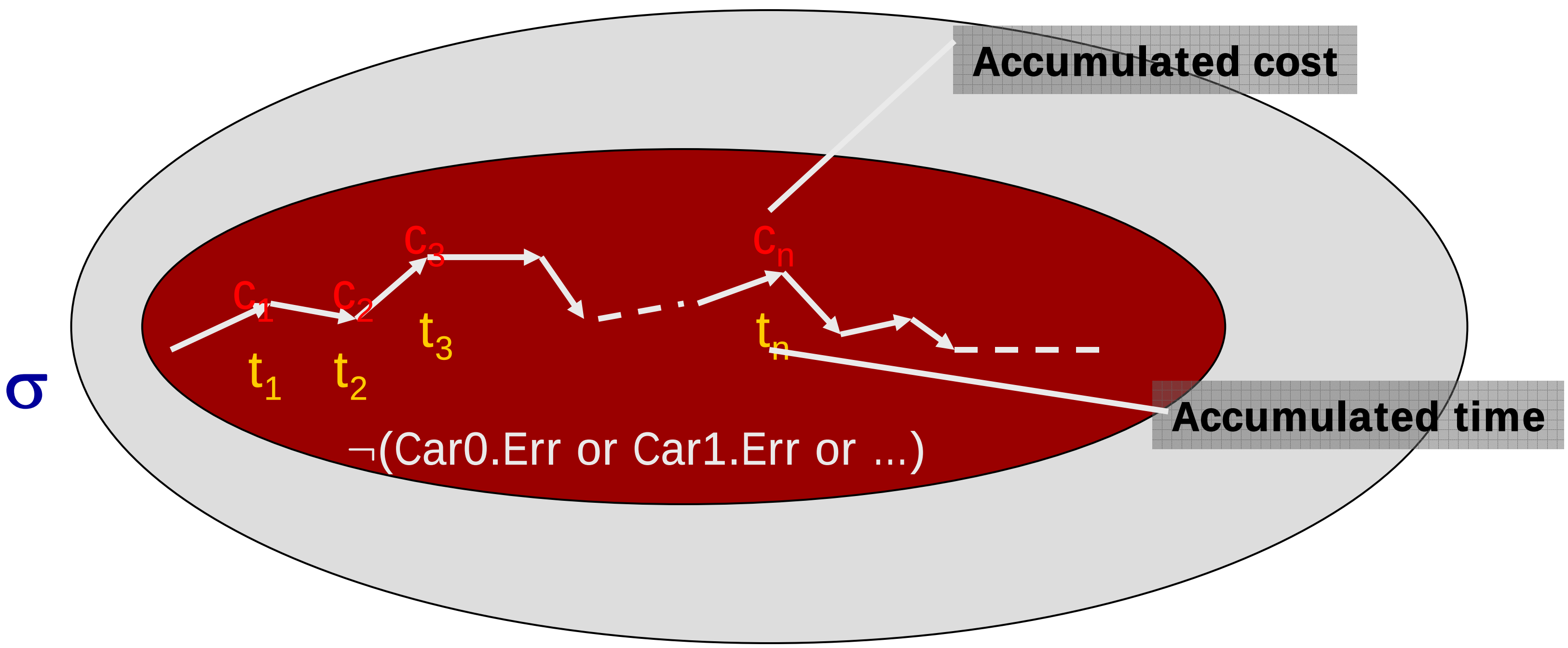
Energy Optimal Scheduling

Using Priced Timed Automata



in Computer Science

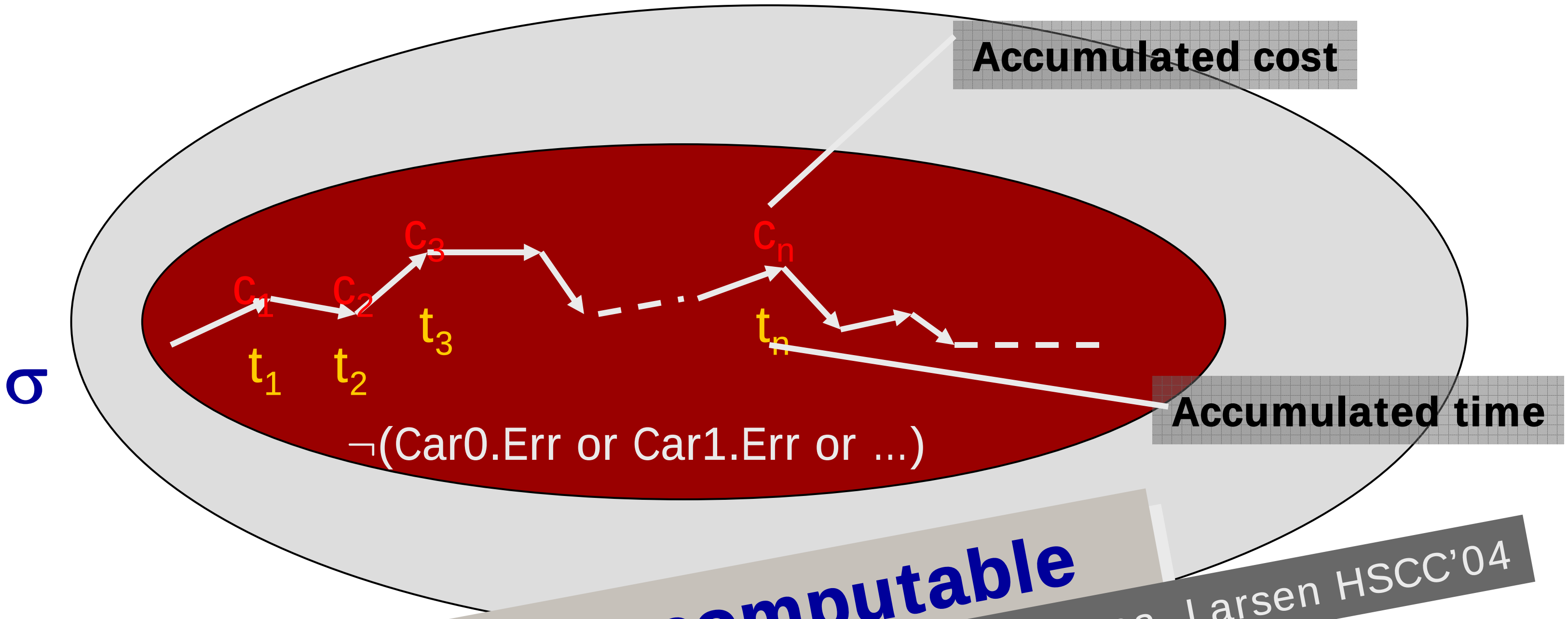
Cost Optimal Scheduling = Optimal Infinite Path



Value of path σ: $\mathrm{val}(\sigma) = \lim_{n\to\infty} c_n/t_n$

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

Cost Optimal Scheduling =
Optimal Infinite Path



THEOREM: σ^* is computable

Bouyer, Brinksma, Larsen HSCC'04

... path σ: $\quad val(\sigma) = \lim_{n\to\infty} c_n/t_n$

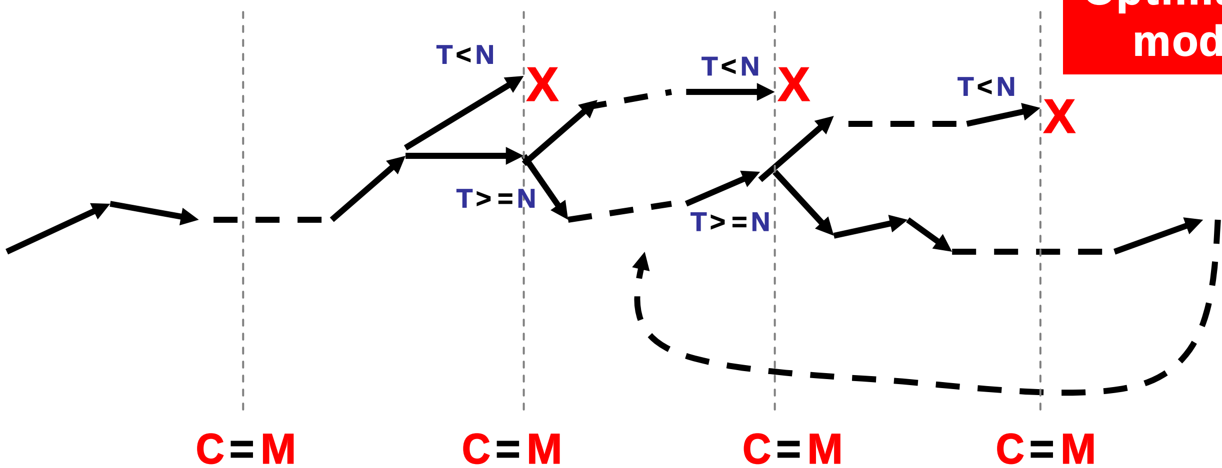
Optimal Schedule σ^*: $\quad val(\sigma^*) = \inf_\sigma val(\sigma)$

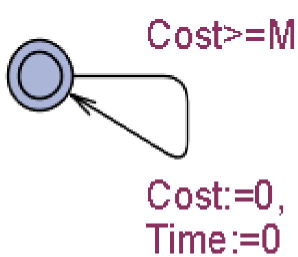
CISS CENTER FOR INDLEJREDE SOFTWARE SYSTEMER

BRICS Basic Research in Computer Science

Approximate Optimal Schedule

Optimal infinite schedule modulo cost-horizon

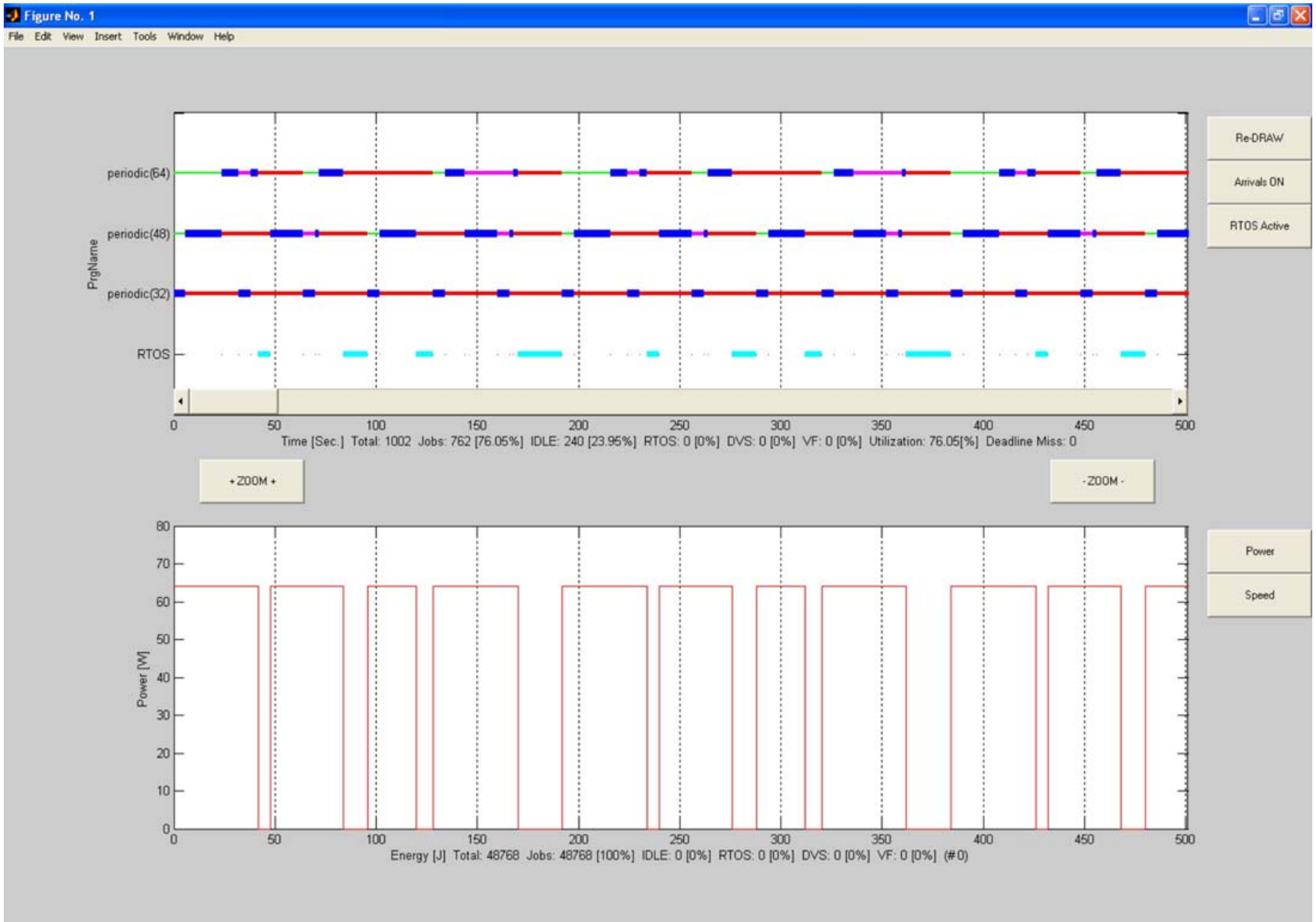




E[] (not (Task0.Error or Task1.Error or Task2.Error)
and
(cost>=M imply time >= N))
=
E[] ϕ(M ,N)

$\sigma \models [\,]\,\phi(M,N)$ imply $val(\sigma) \leq M/N$

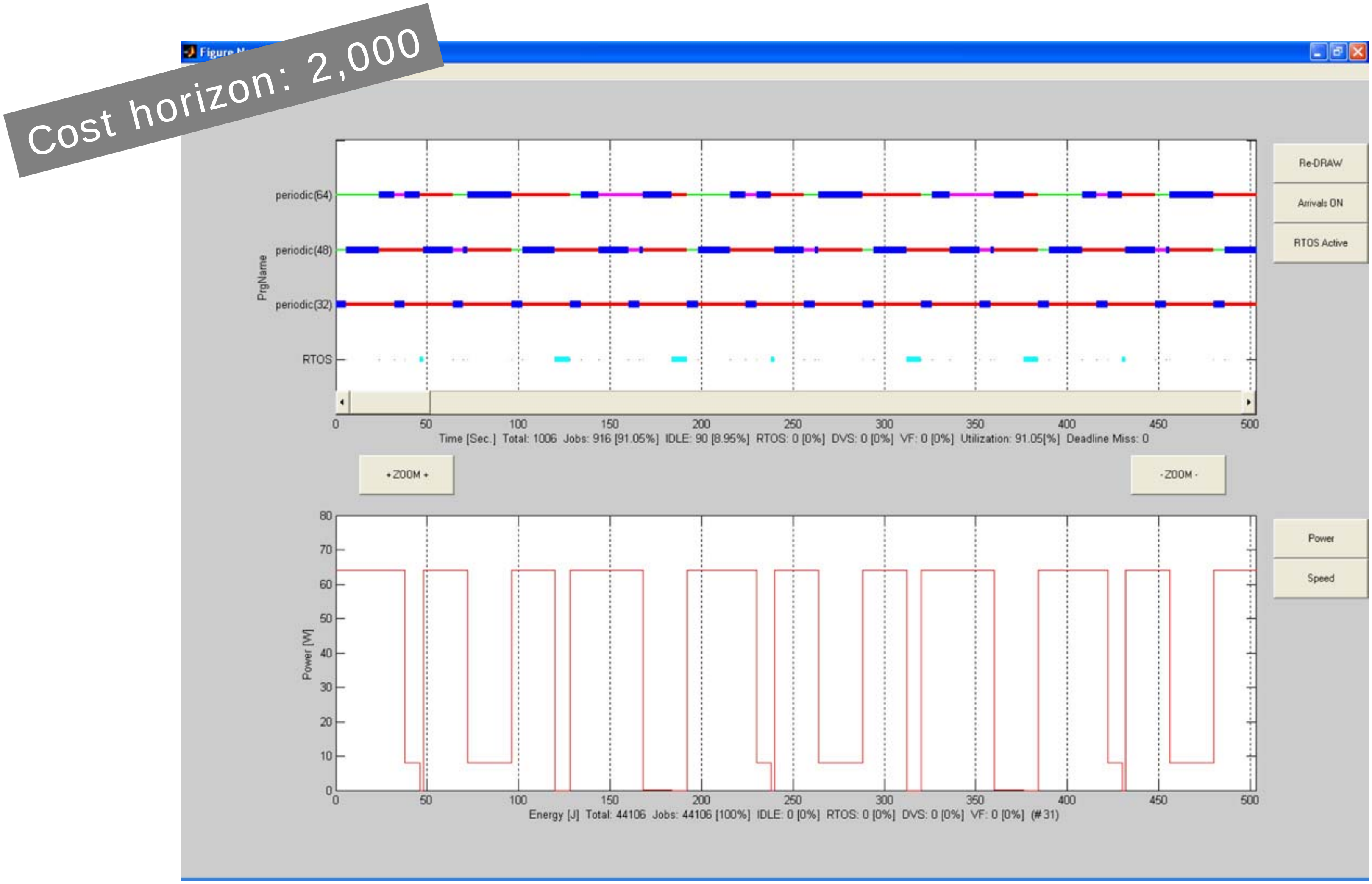
Preliminary Results



$P_1=D_1=32$ $C_1=6$
$P_2=D_2=48$ $C_2=18$
$P_3=D_3=64$ $C_3=12$

EDF w preemption no DVS: avr.: 48

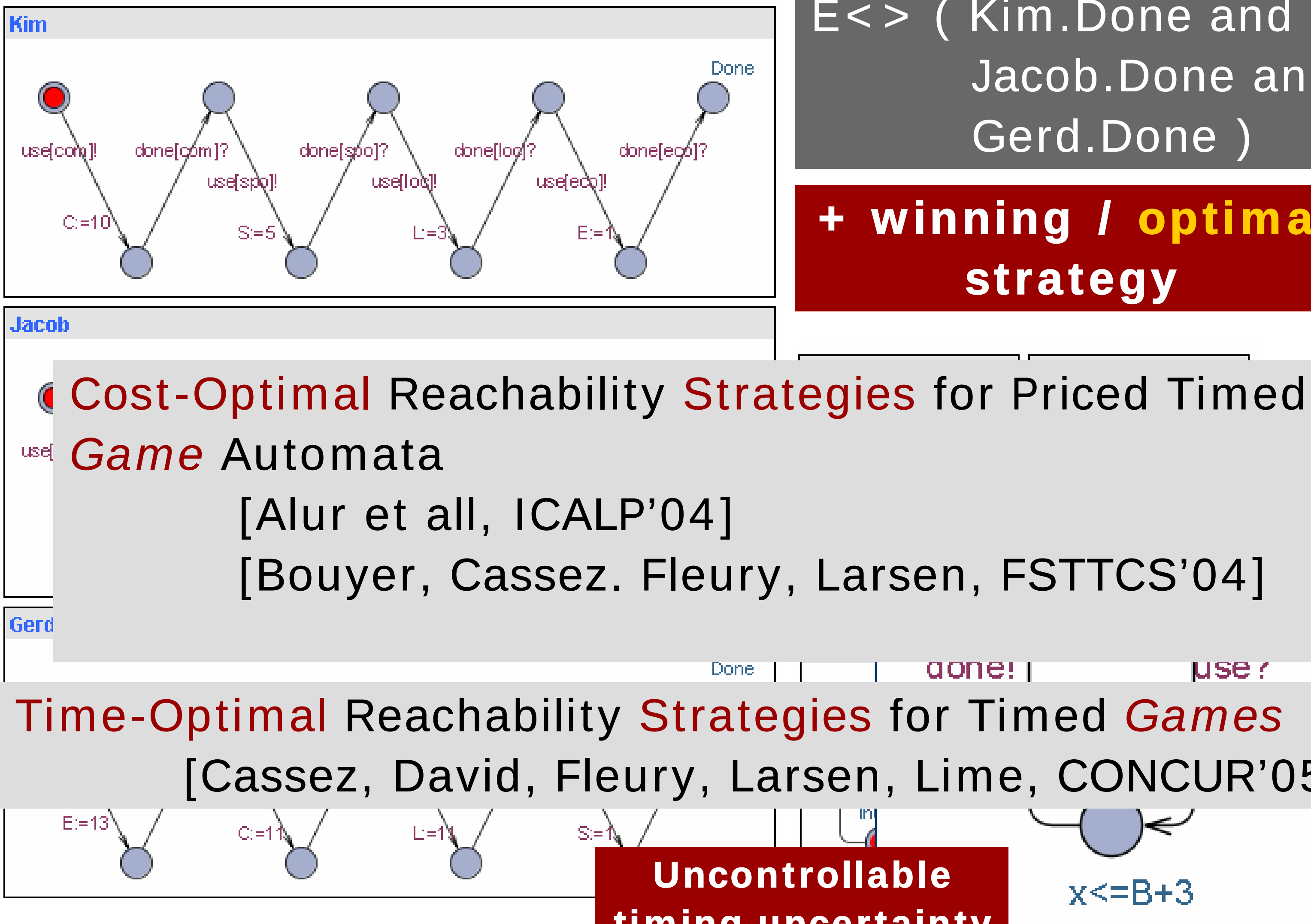
Preliminary Results



EDF w preemption w DVS: avr.: 43.37

$P_1 = D_1 = 32$ $C_1 = 6$
$P_2 = D_2 = 48$ $C_2 = 18$
$P_3 = D_3 = 64$ $C_3 = 12$

Dynamic Scheduling



E<> (Kim.Done and
Jacob.Done and
Gerd.Done)

+ winning / optimal strategy

Cost-Optimal Reachability Strategies for Priced Timed *Game* Automata

[Alur et all, ICALP'04]

[Bouyer, Cassez. Fleury, Larsen, FSTTCS'04]

Time-Optimal Reachability Strategies for Timed *Games*

[Cassez, David, Fleury, Larsen, Lime, CONCUR'05]

Uncontrollable timing uncertainty