CUPPAAL

optimal

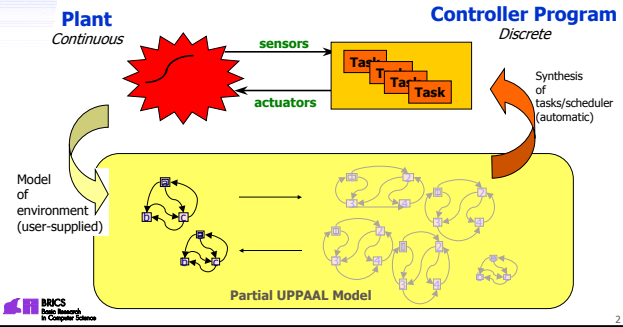
Scheduling & Synthesis of Control Programs

w Gerd Behrman, Ed Brinksma, Ansgar Fehnker,
Thomas Hune, Paul Pettersson,
Judi Romijn, Frits Vaandrager, Henning Dierks

..., HSCC'01, TACAS'01, CAV'01, AIPS'02,.... AMETIST

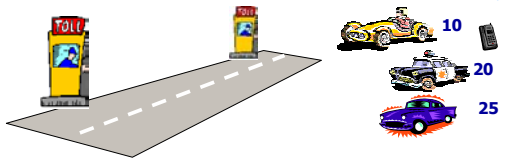
...and Beyond

Synthesis of Control Program



Real Time Scheduling

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



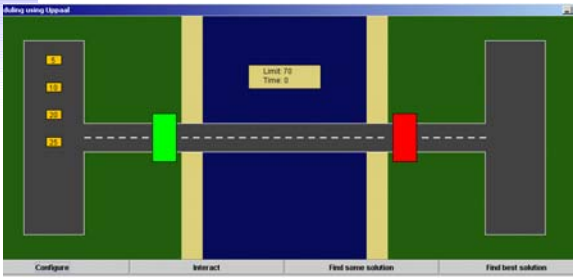
UNSAFE

Crossing Times

5
10
20
25

SAFE

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



OR use UPPAAL

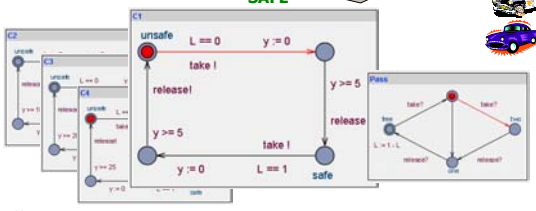
Real Time Scheduling

Solve Scheduling Problem using UPPAAL

UNSAFE

5
10
20
25

SAFE



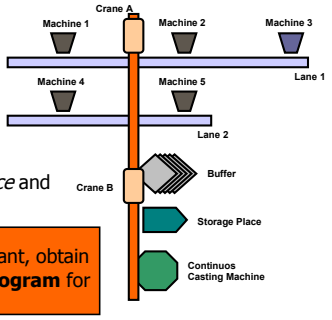
Rush Hour

ObJECTIVE: Get your CAR out

Steel Production Plant

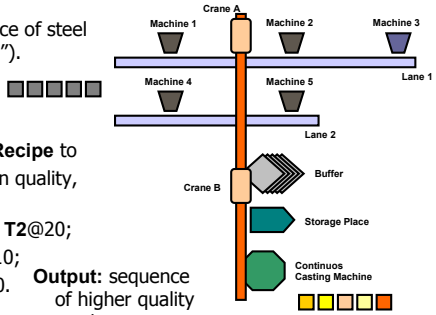
- A. Fehnker
- Hune, Larsen, Pettersson
- Case study of Esprit-LTR project 26270 VHS
- Physical plant of SIDMAR located in Gent, Belgium.
- Part between *blast furnace* and *hot rolling mill*.

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



Steel Production Plant

Input: sequence of steel loads ("pigs").

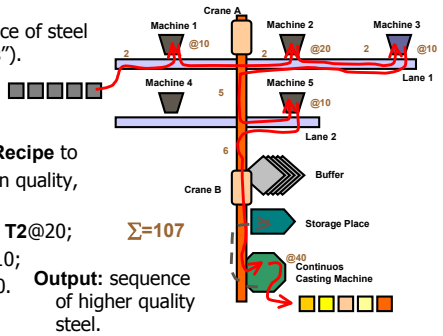


Load follows **Recipe** to become certain quality, e.g:
start; **T1**@10; **T2**@20; **T3**@10; **T2**@10; end within 120.

Output: sequence of higher quality steel.

Steel Production Plant

Input: sequence of steel loads ("pigs").



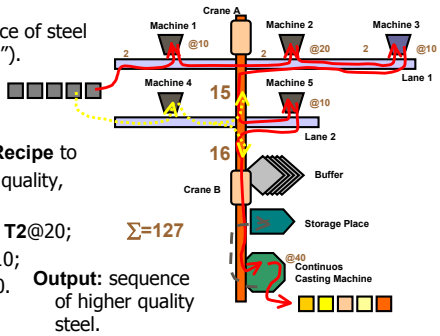
Load follows **Recipe** to become certain quality, e.g:
start; **T1**@10; **T2**@20; **T3**@10; **T2**@10; end within 120.

Σ=107

Output: sequence of higher quality steel.

Steel Production Plant

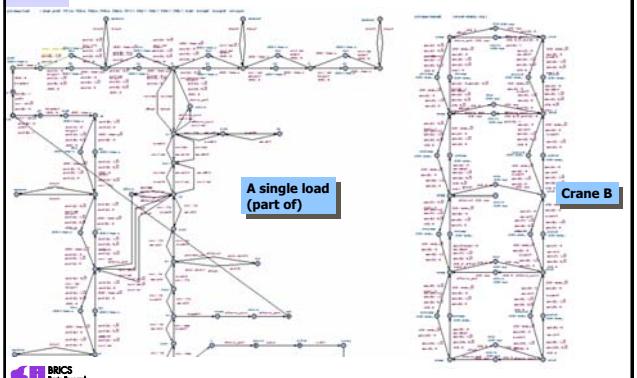
Input: sequence of steel loads ("pigs").



Load follows **Recipe** to obtain certain quality, e.g:
start; **T1**@10; **T2**@20; **T3**@10; **T2**@10; end within 120.

Σ=127

Output: sequence of higher quality steel.



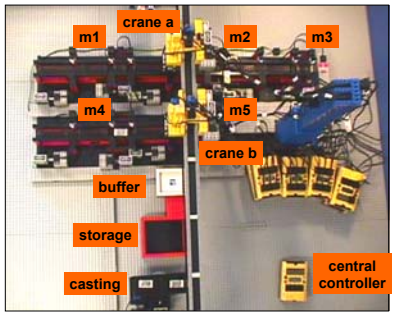
Experiment

n	All Guides						Some Guides						No Guides					
	BFS		DFS		BSH		BFS		DFS		BSH		BFS		DFS		BSH	
	s	MB	s	MB	s	MB	s	MB	s	MB	s	MB	s	MB	s	MB	s	MB
1	0,1	0,9	0,1	0,9	0,1	0,9	0,1	0,9	0,1	0,9	0,1	0,9	3,2	6,1	0,8	2,2	3.9	3.3
2	18,4	36,4	0,1	1	0,1	1,1	-	-	4,4	7,8	7,8	1,2	-	-	19,5	36,1	-	-
3	-	-	3,2	6,5	3,4	1,4	-	-	72,4	92,1	901	3,4	-	-	-	-	-	-
4	-	-	4	8,2	4,6	1,8	-	-	-	-	-	-	-	-	-	-	-	-
5	-	-	5	10,2	5,5	2,2	-	-	-	-	-	-	-	-	-	-	-	-
10	-	-	13,3	25,3	16,1	9,3	-	-	-	-	-	-	-	-	-	-	-	-
15	-	-	31,6	51,2	48,1	22,2	-	-	-	-	-	-	-	-	-	-	-	-
20	-	-	61,8	89,6	332	46,1	-	-	-	-	-	-	-	-	-	-	-	-
25	-	-	104	144	87,2	83,3	-	-	-	-	-	-	-	-	-	-	-	-
30	-	-	166	216	124,2	136	-	-	-	-	-	-	-	-	-	-	-	-
35			209	250	-	-	-	-	-	-	-	-	-	-	-	-	-	-

- **BFS** = breadth-first search, **DFS** = depth-first search, **BSH** = bit-state hashing,
- **"-"** = requires >2h (on 450MHz Pentium III), >256 MB, or suitable hash-table size was not found.
- **System size**: 2n+5 automata and 3n+3 clocks, if n=35: 75 automata and 108 clocks.
- **Schedule generated for n=60 on Sun Ultra with 2x300MHz with 1024MB in 2257s .**

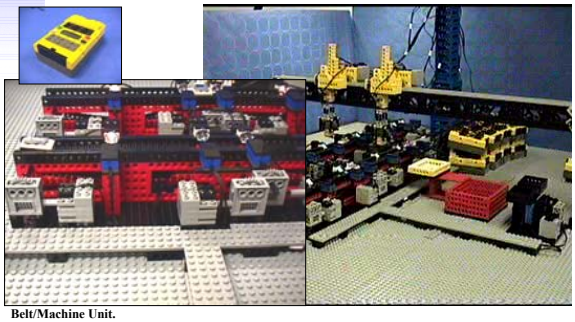
LEGO Plant Model

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



Synthesis

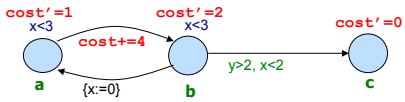
LEGO Plant Model



Belt/Machine Unit.

Linearly Priced Timed Automata:

Optimal Scheduling



- **Timed Automata + Costs on transitions and locations.**
  - Cost of performing transition: `Transition cost`.
  - Cost of performing delay **d**: (**d** x `Location cost`).

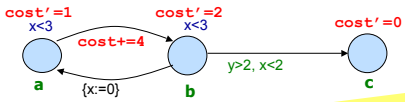
- Trace:

$$(a,x=y=0) \xrightarrow{4} (b,x=y=0) \xrightarrow[2.5 \times 2]{\varepsilon(2.5)} (b,x=y=2.5) \xrightarrow{0} (a,x=0,y=2.5)$$

- Cost of Execution Trace: Sum of costs: **4 + 5 + 0 = 9**

Linearly Priced Timed Automata:

Optimal Scheduling



Problem :

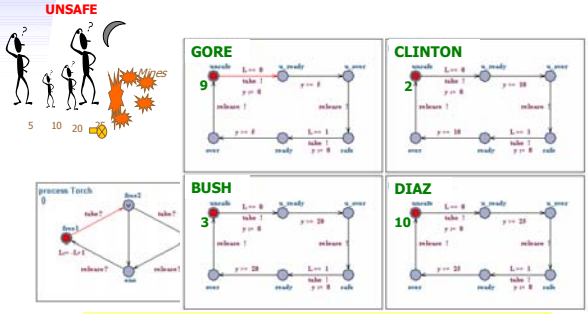
Find the minimum cost of reaching location **c**

- Tra

$$\xrightarrow[2.5 \times 2]{} (b,x=y=2.5) \xrightarrow{0} (a,x=0,y=2.5)$$

- Cost of Execution Trace: Sum of costs: **4 + 5 + 0 = 9**

EXAMPLE: Optimal rescue plan for important persons (Presidents and Actors)



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

Experiments *MC Order*

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

Jobshop Scheduling

[TACAS'2001]

	Sport	Economy	Local News	Comic Stip
Kim	2. 5 min	4. 1 min	3. 3 min	1. 10 min
Jüri	1. 10 min	2. 20 min	3. 1 min	4. 1 min
Jan	4. 1 min	1. 13 min	3. 11 min	2. 11 min
Wang	1. 1 min	2. 1 min	3. 1 min	4. 1 min

Problem: compute the minimal **MAKESPAN**

NP-hard
- Simulated annealing
- Shiffted bottleneck
- Branch-and-Bound
- Gentic Algorithms

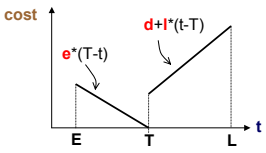
Experiments

B-&-B algorithm running for 60 sec.

		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	-	-	-	-	-	-	2084	-	769	-	626	-	597
		la04	-	-	-	-	-	-	2212	-	783	-	639	-	590
	j=15	la05	-	-	-	-	593	9791	1955	-	696	-	593	284	593
		la06	-	-	-	-	-	-	3656	-	1076	-	926	480	926
		la07	-	-	-	-	-	-	3410	-	1113	-	890	-	890
		la08	-	-	-	-	-	-	3520	-	1009	-	863	400	863
		la09	-	-	-	-	-	-	3984	-	1154	-	951	425	951
	j=20	la10	-	-	-	-	-	-	3681	-	1063	-	958	454	958
		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
	j=10	la15	-	-	-	-	-	-	5264	-	1459	-	1289	-	1207
		la16	-	-	-	-	-	-	4849	-	1298	-	1022	-	945
m=10		la17	-	-	-	-	-	-	4299	-	938	-	786	-	784
		la18	-	-	-	-	-	-	4763	-	1034	-	922	-	848
		la19	-	-	-	-	-	-	4566	-	1140	-	904	-	842
		la20	-	-	-	-	-	-	5056	-	1378	-	964	-	902
	j=15	la21	-	-	-	-	-	-	7008	-	1326	-	1149	-	(1040,1053)
		la22	-	-	-	-	-	-	6920	-	1413	-	1047	-	927
		la23	-	-	-	-	-	-	7676	-	1357	-	1075	-	1032
		la24	-	-	-	-	-	-	7237	-	1346	-	1061	-	935
		la25	-	-	-	-	-	-	7141	-	1290	-	1070	-	977

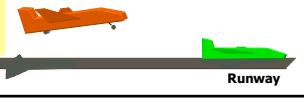
Lawrence Job Shop Problems

Example: Aircraft Landing

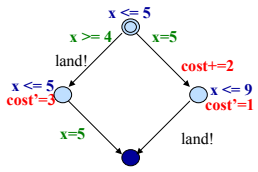


E earliest landing time
T target time
L latest time
e cost rate for being early
l cost rate for being late
d fixed cost for being late

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

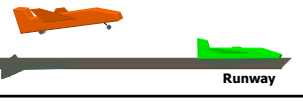


Example: Aircraft Landing



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



Aircraft Landing

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		

Symbolic Semantics

Zones

Definition

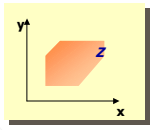
A *zone* Z over a set of clocks C is a finite conjunction of simple constraints of the forms:

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

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

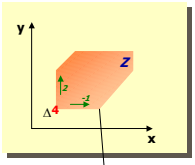


Reset: $\{x\}Z = \{u[0/x] \mid u \models Z\}$

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:

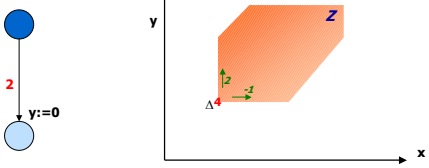
- Z is a zone
- $c \in \mathbb{N}$ describes the cost of Δ_Z
- $r : C \to \mathbb{Z}$ gives a *rate* for any clock $x \in C$.

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

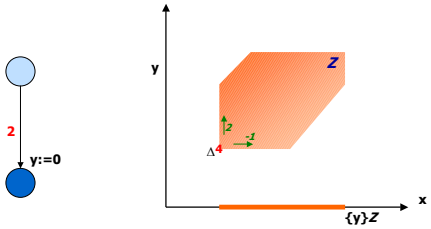
$$\mathrm{Cost}(u, P) = c + \sum_{x \in C} r(x) \cdot (u(x) - \Delta_Z(x))$$

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

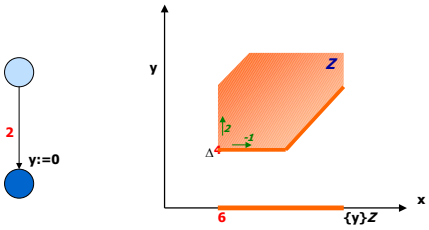
Reset



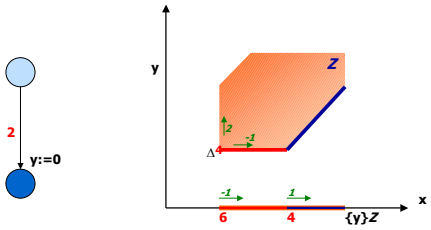
Reset



Reset



Reset



A split of $\{y\}Z$

Facets

The solution

Relative Facets

Definition

Whenever $x \geq y - l$ (resp. $x \geq l$) is a constraint of Z, then $Z \wedge (x = y - l)$ (resp. $Z \wedge x = l$) is a *lower facet* of Z *relative* to x. We denote by $LF_x(Z)$ the collection of lower facets of Z relative to x.

Similarly the collection of upper facets $UF_x(Z)$ relative to x may be defined.

Theorem

Any "point" in a projection of a zone will be in the projection of one of the lower (upper) facets of the zone:

$$\{y\}Z = \bigcup_{F \in LF_y(Z)} \{y\}F \qquad \{y\}Z = \bigcup_{F \in UF_y(Z)} \{y\}F$$

Reset

Definition

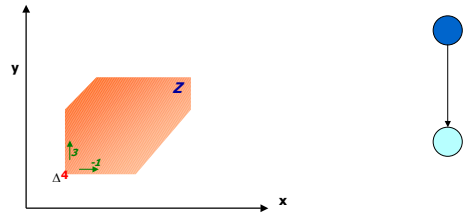
Let $P = (F, c, r)$ where F is a facet relative to y in the sense that $y = x - l$ or $y = l$ is a constraint of F. Now $\{y\}P = (F', c', r')$ where:

$$\begin{aligned} F' &= \{y\}F \\ c' &= c \\ r'(z) &= \begin{cases} r(z) & z \neq x \\ r(y) + r(x) & z = x \end{cases} \end{aligned}$$

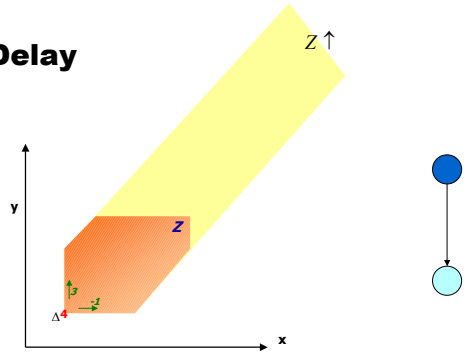
Let $P = (Z, c, r)$ be a priced zone with $r(y) \geq 0$. Then:

$$\{y\}P = \left\{ \{y\}Q \mid Q \in LF_y(P) \right\}$$

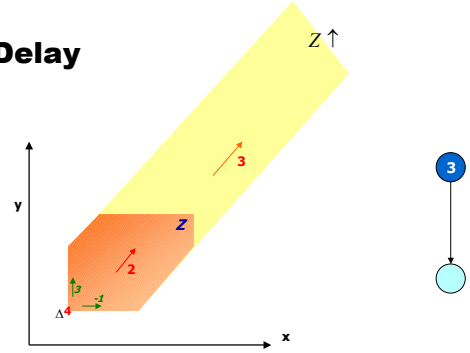
Delay



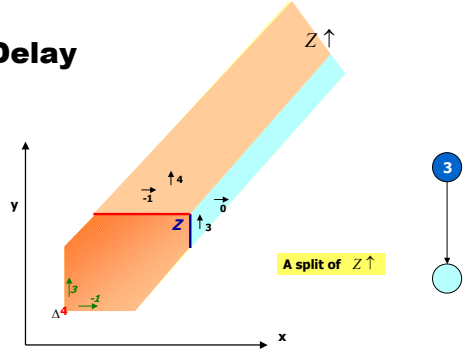
Delay



Delay



Delay



Facets

The solution

Facets

Definition

Whenever $x \geq l$ is a constraint in Z then $Z \wedge (x = l)$ is a *lower facet* of Z.

We denote by $LF(Z)$ the collection of all lower facets of Z.

Theorem

Any state reachable by delay from "points" in Z are reachable from one of the lower facets of Z, i.e.:

$$Z^{\uparrow} = \bigcup_{F \in LF(Z)} F^{\uparrow}$$

Theorem

Any state reachable by delay from "points" in Z are either within Z or reachable from one of the upper facets of Z, i.e.:

$$Z^{\uparrow} = Z \cup \bigcup_{F \in UF(Z)} F^{\uparrow}$$

Delay

Definition

Let $P = (F, c, r)$ where F is a lower (upper) facet and let $p \in \mathbb{R}_{\geq 0}$ (the price rate of a location). Then $Delay(p, P) = (F', c', r')$ where:

$$F' = F^{\uparrow}$$
$$c' = c$$
$$r'(z) = \begin{cases} r(z) & F \text{ not a lower facet wrt } z \\ p - \sum_{x \neq z} r(x) & F \text{ is a lower facet wrt } z \end{cases}$$

NOTE: By adding a clock which is always reset when taking a transition we would only be required to apply $Delay$ on facets! There is a general construction however.

Branch & Bound Algorithm

```
Cost := ∞
Passed := ∅
Waiting := {(l0, R0)}
while Waiting ≠ ∅ do
  select (l, R) from Waiting
  if l = lg and mincost(R) < Cost then
    Cost := mincost(R)
  if for all (l, R') in Passed: R' ⋠ R then
    add (l, R) to Passed
    for all (l', R') such that (l, R) → (l', R'): add (l', R') to Waiting
return Cost
```


Branch & Bound Algorithm

```
Cost := ∞
Passed := ∅
Waiting := {(l0, R0)}
while Waiting ≠ ∅ do
  select (l, R) from Waiting
  if l = lg and mincost(R) < Cost then
    Cost := mincost(R)
  if for all (l, R') in Passed: R' ⋠ R then
    add (l, R) to Passed
    for all (l', R') such that (l, R) → (l', R'): add (l', R') to Waiting
return Cost
```


Branch & Bound Algorithm

```
Cost := ∞
Passed := ∅
Waiting := {(l0, R0)}
while Waiting ≠ ∅ do
  select (l, R) from Waiting
  if l = lg and mincost(R) < Cost then
    Cost := mincost(R)
  if for all (l, R') in Passed: R' ⋠ R then
    add (l, R) to Passed
    for all (l', R') such that (l, R) → (l', R'): add (l', R') to Waiting
return Cost
```


Branch & Bound Algorithm

```
Cost := ∞
Passed := ∅
Waiting := {(l0, R0)}
while Waiting ≠ ∅ do
  select (l, R) from Waiting
  if l = lg and mincost(R) < Cost then
    Cost := mincost(R)
  if for all (l, R') in Passed: R' ⋠ R then
    add (l, R) to Passed
    for all (l', R') such that (l, R) → (l', R'): add (l', R') to Waiting
return Cost
```

$R' \leq R$

R'* is bigger & cheaper than *R

≤ is a well-quasi ordering which guarantees termination!

Branch & Bound Algorithm

```
Cost := ∞
Passed := ∅
Waiting := {(l0, R0)}
while Waiting ≠ ∅ do
  select (l, R) from Waiting
  if l = lg and mincost(R) < Cost then
    Cost := mincost(R)
  if for all (l, R') in Passed: R' ⋠ R then
    add (l, R) to Passed
    for all (l', R') such that (l, R) → (l', R'): add (l', R') to Waiting
return Cost
```


Related Work

- Alur, Courcourbetis, Henzinger (1993)
  Accumulated delays in Realtime Systems
- Alur, Torre, Pappas (HSCC'01)
  Optimal Paths in Weighted Timed Automata
- Behrmann, Fehnker, et all (HSCC'01)
  Minimum-Cost Reachability for Priced Timed Automata

Related Work (cont)

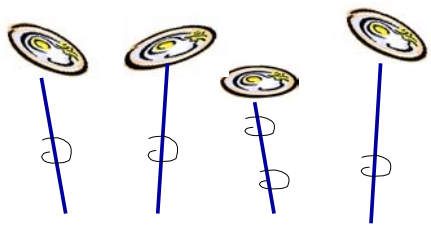
- Asarin & Maler (1999)
  Time optimal control using backwards fixed point computation
- Niebert, Tripakis & Yovine (2000)
  Minimum-time reachability using forward reachability
- Behrmann, Fehnker et all (TACAS'2001)
  Minimum-time reachability using Branch-and-Bound
- Abdeddaim, Maler (CAV'01)
  Job-Shop Scheduling using Timed Automata
- General Trend (AAAI'01):
  Integrating Scheduling/Planning and Model Checking

Optimal Infinite Scheduling

work in progress

Ed Brinksma **Kim G Larsen**
Patricia Bouyer
Emmanuel Fleury

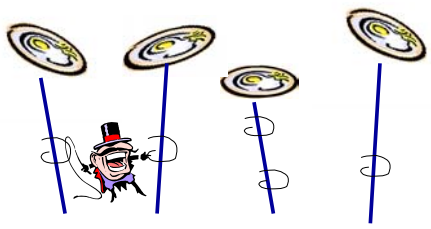
The Plate Juggling Problem



Problem: avoid having the plates falling down

The Plate Juggling Problem

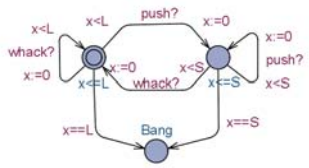
thanks to Oded



Problem: avoid having the plates falling down

The Plate Juggling Problem

using Timed Automata



A Plate

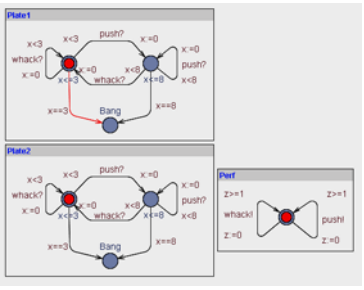
The Joggler

Infinite Scheduling

E□ ¬(Plate1.Bang or Plate2.Bang or ...)

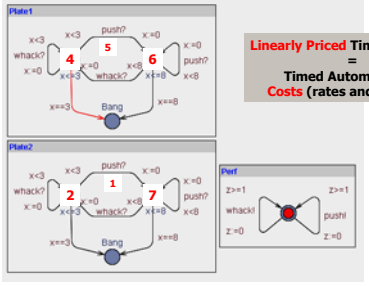
Optimal Infinite Scheduling

with respect to what ??



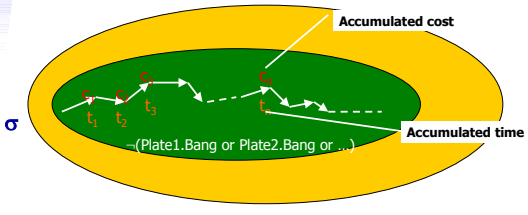
Optimal Infinite Scheduling

with respect to what ??



Linearly Priced Timed Automata
=
Timed Automata with
Costs (rates and impulses)

Optimal Infinite Scheduling

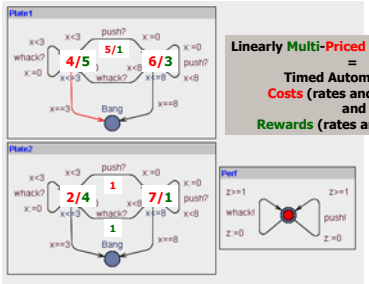


Value of path σ: $val(\sigma) = \lim_{n\to\infty} c_n/t_n$

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

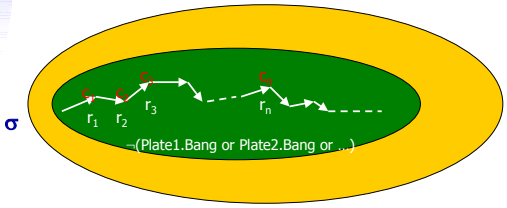
Optimal Infinite Scheduling

with respect to what ??



Linearly Multi-Priced Timed Automata
=
Timed Automata with
Costs (rates and impulses)
and
Rewards (rates and impulses)

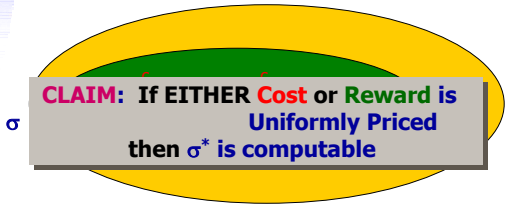
Optimal Infinite Scheduling



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

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

Optimal Infinite Scheduling



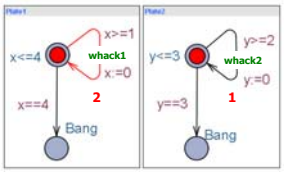
Value of path σ: $val(\sigma) = \lim_{n\to\infty} c_n/r_n$

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

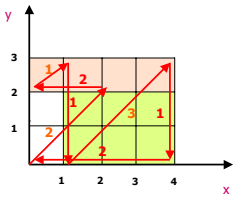
Appr Optimal Schedules

using UPPAAL

cost: impulse
reward: time



Simplified Juggling Problem

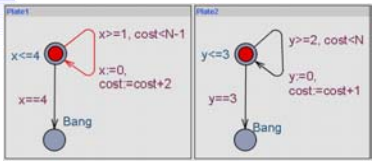


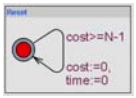
Infinite Schedule:
σ = (ε(2);whack1;epsilon(1);
whack2;ε(3);whack2;whack1)*

val(σ)= (2+1+1+2)/(2+1+3) = 1

σ optimal ??

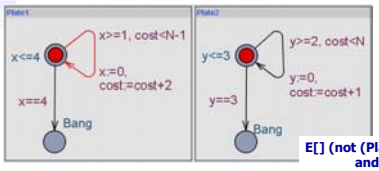
Appr Optimal Schedules

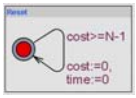




int[0,N] cost;
clock time;

Appr Optimal Schedules

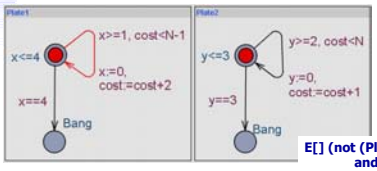


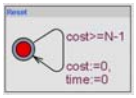


int[0,N] cost;
clock time;

E[] (not (Plate1.Bang or Plate2.Bang)
and
(cost>=N-1 imply time >= M))
=
E[] ϕ(N,M)

Appr Optimal Schedules



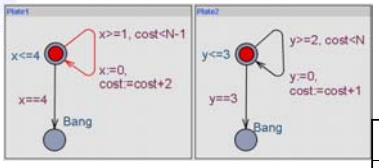


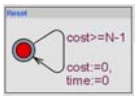
int[0,N] cost;
clock time;

E[] (not (Plate1.Bang or Plate2.Bang)
and
(cost>=N-1 imply time >= M))
=
E[] ϕ(N,M)

σ ⊨ [] ϕ(N,M) imply val(σ)≤ N/M

Appr Optimal Schedules





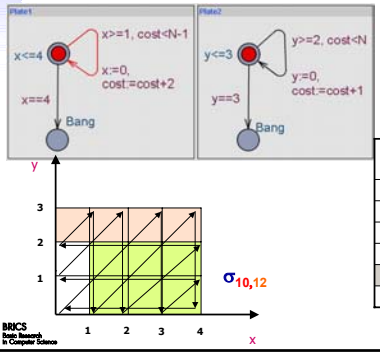
int[0,N] cost;
clock time;

E[] (not (Plate1.Bang or Plate2.Bang)
and
(cost>=N-1 imply time >= M))
=
E[] ϕ(N,M)

(N,M)	∃σ. σ⊨[] ϕ(N,M)
(3,3)	YES
(3,4)	NO
(7,8)	YES
(7,9)	NO
(10,12)	YES
(10,13)	NO

BRICS Basic Research in Computer Science

Appr Optimal Schedules



int[0,N] cost;
clock time;

(N,M)	∃σ. σ⊨[] ϕ(N,M)
(3,3)	YES
(3,4)	NO
(7,8)	YES
(7,9)	NO
(10,12)	YES
(10,13)	NO

BRICS Basic Research in Computer Science

To be Continued ....