Timed Automata
&
Model Checking

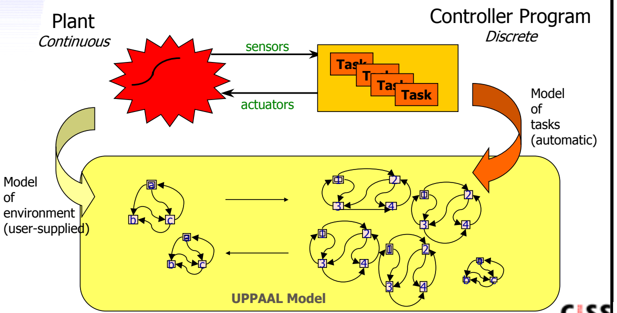
Using UPPAALx : x ∈ {1,2,3,4}

Kim Guldstrand Larsen
BRICS@Aalborg & FMT@Twente

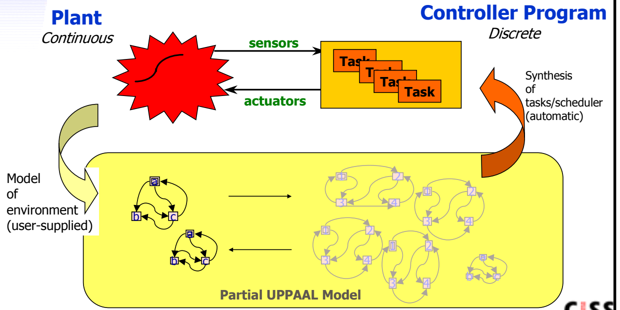
BRICS Basic Research in Computer Science

Formal methods & Tools

Validation & Verification
Construction of UPPAAL models

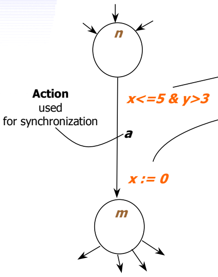


…and Beyond
Synthesis of Control Program



Timed Automata *review*

Alur & Dill 1990



Clocks: *x, y*

Guard
Boolean combination of integer bounds on **clocks** and **clock-differences**.

Reset
Action perfomed on clocks

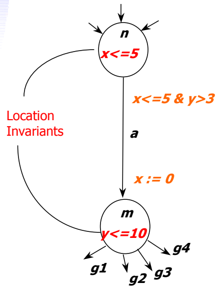
State
(***location*** , x=v , y=u) where v,u are in $\mathbf{R}$

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 *review*
Invariants



Clocks: *x, y*

Transitions

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

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

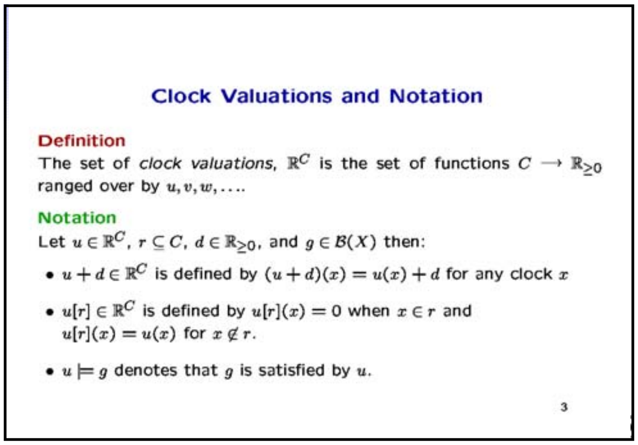
Invariants ensure progress!!

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 $\mathbb{Q}$).

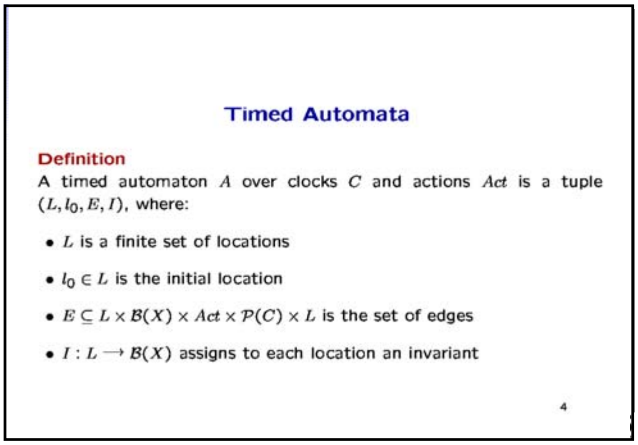


Clock Valuations and Notation

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:

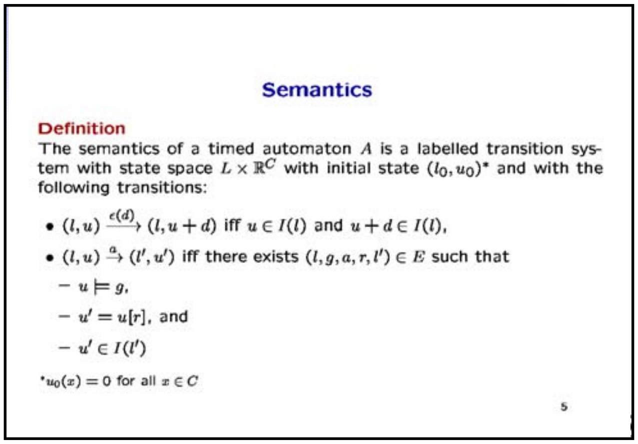
- $u + d \in \mathbb{R}^C$ is defined by $(u+d)(x) = u(x) + d$ for any clock x
- $u[r] \in \mathbb{R}^C$ is defined by $u[r](x) = 0$ when $x \in r$ and $u[r](x) = u(x)$ for $x \notin r$.
- $u \models g$ denotes that g is satisfied by u.



Timed Automata

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

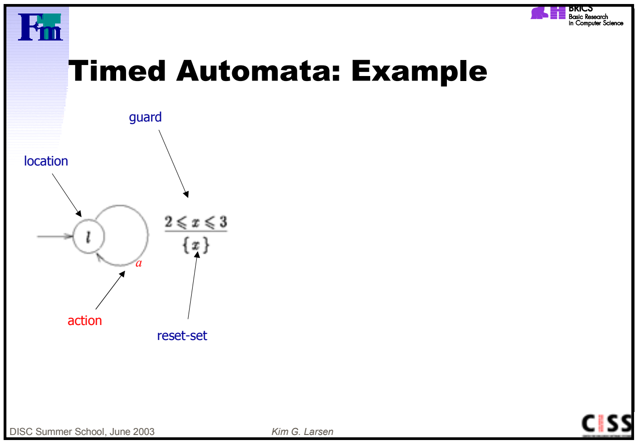


Semantics

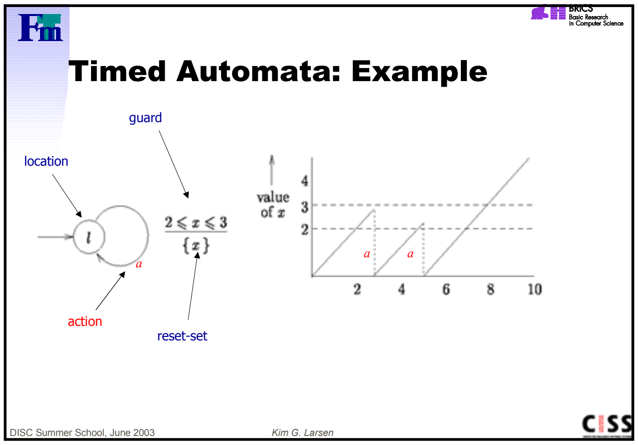
Definition
The semantics of a timed automaton A is a labelled transition system with state space $L \times \mathbb{R}^C$ with initial state (l_0, u_0)* and with the following transitions:

- $(l, u) \xrightarrow{\epsilon(d)} (l, u+d)$ 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
  - $u' \in I(l')$

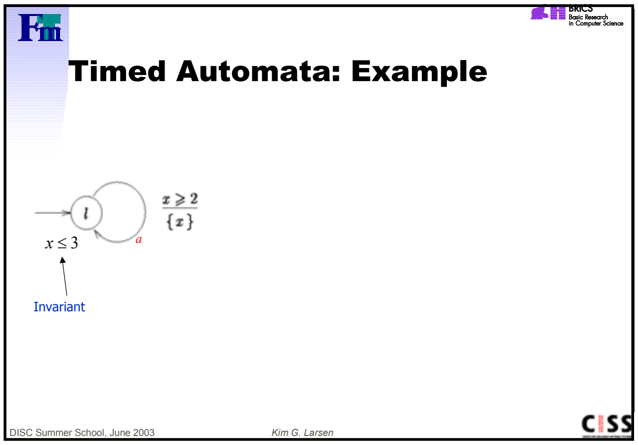
*$u_0(x) = 0$ for all $x \in C$



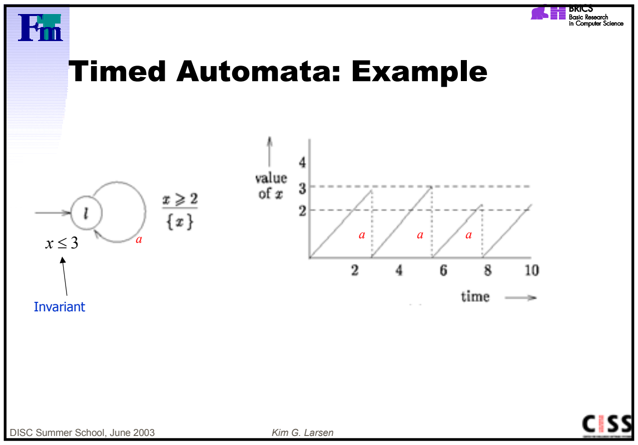
Timed Automata: Example



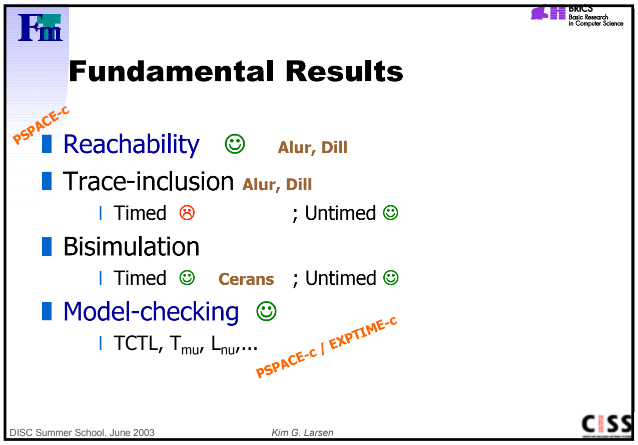
Timed Automata: Example



Timed Automata: Example



Timed Automata: Example

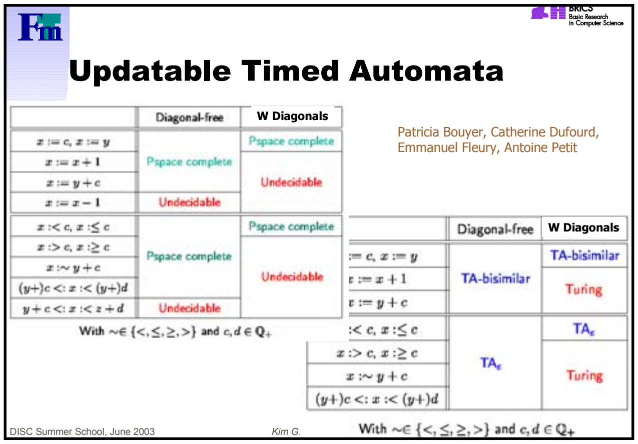


Fundamental Results

PSPACE-c

- Reachability ☺ Alur, Dill
- Trace-inclusion Alur, Dill
  - Timed ☹ ; Untimed ☺
- Bisimulation
  - Timed ☺ Cerans ; Untimed ☺
- Model-checking ☺
  - TCTL, T_{mu}, L_{nu},…

PSPACE-c / EXPTIME-c



Updatable Timed Automata

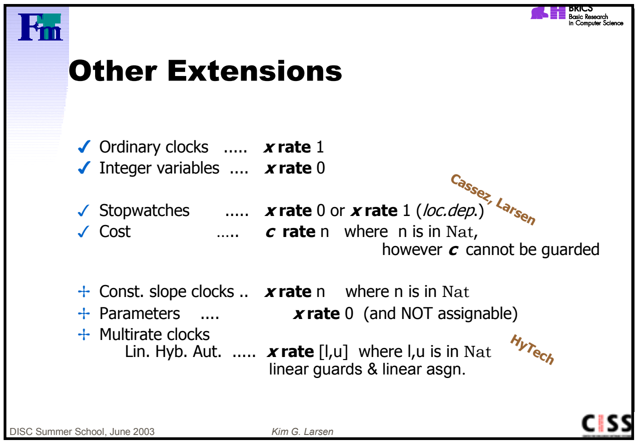
Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, Antoine Petit

	Diagonal-free	W Diagonals
$x := c,\ x := y$	Pspace complete	Pspace complete
$x := x + 1$		Undecidable
$x := y + c$		
$x := x - 1$	Undecidable	
$x :< c,\ x :\le c$	Pspace complete	Pspace complete
$x :> c,\ x :\ge c$		Undecidable
$x :\sim y + c$		
$(y+)c <: x :< (y+)d$		
$y + c <: x :< z + d$	Undecidable	

With $\sim \in \{<, \le, \ge, >\}$ and $c, d \in \mathbb{Q}_+$

	Diagonal-free	W Diagonals
$:= c,\ x := y$	TA-bisimilar	TA-bisimilar
$:= x + 1$		Turing
$:= y + c$		
$:< c,\ x :\le c$	TA_ϵ	TA_ϵ
$x :> c,\ x :\ge c$		Turing
$x :\sim y + c$		
$(y+)c <: x :< (y+)d$		

With $\sim \in \{<, \le, \ge, >\}$ and $c, d \in \mathbb{Q}_+$



Other Extensions

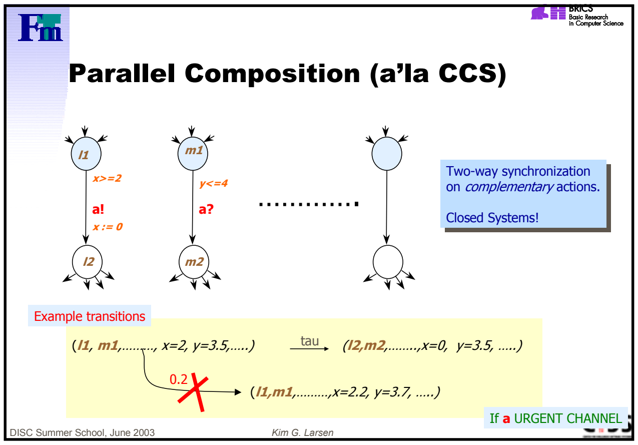
- ✓ Ordinary clocks ….. ***x* rate** 1
- ✓ Integer variables …. ***x* rate** 0

Cassez, Larsen

- ✓ Stopwatches ….. ***x* rate** 0 or ***x* rate** 1 (*loc.dep.*)
- ✓ Cost ….. ***c* rate** n where n is in Nat,
  however ***c*** cannot be guarded

- + Const. slope clocks .. ***x* rate** n where n is in Nat
- + Parameters …. ***x* rate** 0 (and NOT assignable)
- + Multirate clocks
  Lin. Hyb. Aut. ….. ***x* rate** [l,u] where l,u is in Nat
  linear guards & linear asgn.

HyTech



Parallel Composition (a'la CCS)

Two-way synchronization on *complementary* actions.

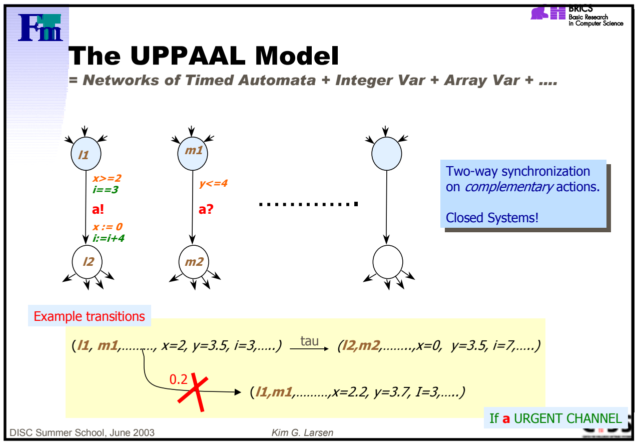
Closed Systems!

Example transitions

(*l1*, *m1*,………, *x=2, y=3.5,…..*) $\xrightarrow{\text{tau}}$ (*l2*,*m2*,……..,*x=0, y=3.5, …..*)

(*l1*, *m1*,………, *x=2, y=3.5,…..*) $\xrightarrow{0.2}$ (*l1*,*m1*,………,*x=2.2, y=3.7, …..*)

If **a** URGENT CHANNEL



The UPPAAL Model

= Networks of Timed Automata + Integer Var + Array Var + ….

Two-way synchronization on *complementary* actions.

Closed Systems!

Example transitions

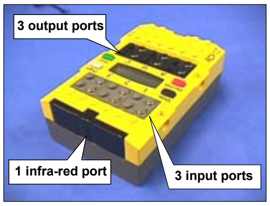
(*l1*, *m1*,………, *x=2, y=3.5, i=3,…..*) $\xrightarrow{\text{tau}}$ (*l2*,*m2*,……..,*x=0, y=3.5, i=7,…..*)

(*l1*, *m1*,………, *x=2, y=3.5, i=3,…..*) $\xrightarrow{0.2}$ (*l1*,*m1*,………,*x=2.2, y=3.7, I=3,…..*)

If **a** URGENT CHANNEL

LEGO Mindstorms/RCX

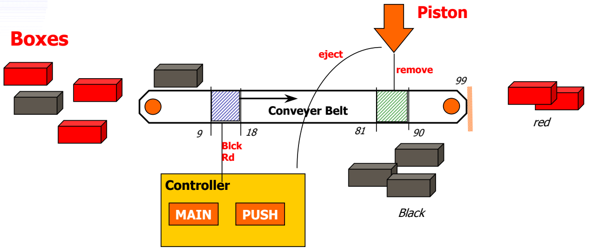
- Sensors: temperature, light, rotation, pressure.
- Actuators: motors, lamps,
- Virtual machine:
  - 10 tasks, 4 timers, 16 integers.
- Several Programming Languages:
  - NotQuiteC, Mindstorm, Robotics, legOS, etc.



First UPPAAL model

Sorting of Lego Boxes

Ken Tindell



Exercise: Design **Controller** so that only black boxes are being pushed out

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);
  }
}
```

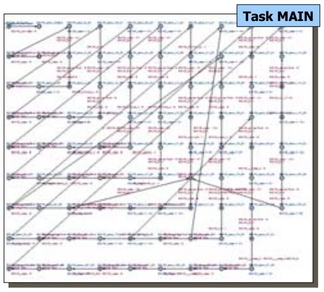
```
task PUSH{
  while(true){
    wait(Timer(1)>DELAY && active==1);
    active=0;
    Rev(OUT_C,1);
    Sleep(8);
    Fwd(OUT_C,1);
    Sleep(12);
    Off(OUT_C);
  }
}
```

UPPAAL Demo

IDA foredrag 20.4.99

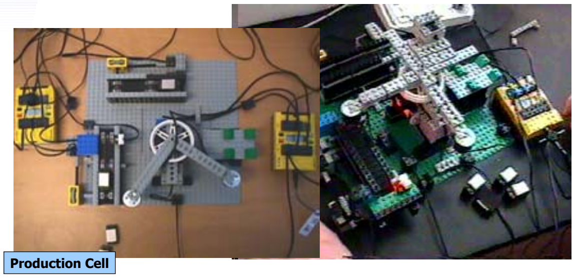
From RCX to UPPAAL

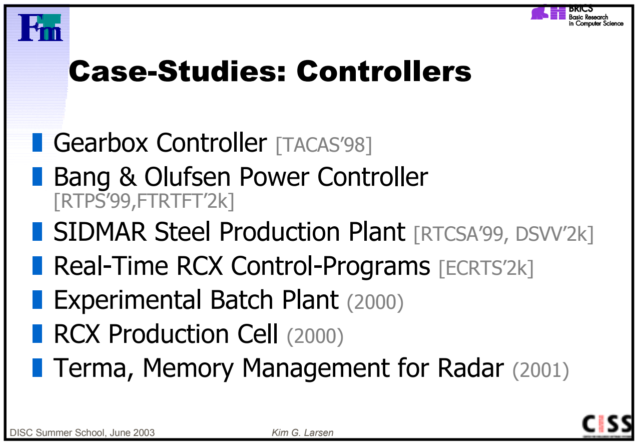
- Model includes Round-Robin Scheduler.
- Compilation of RCX tasks into TA models.
- Presented at ECRTS 2000


Task MAIN

The Production Cell

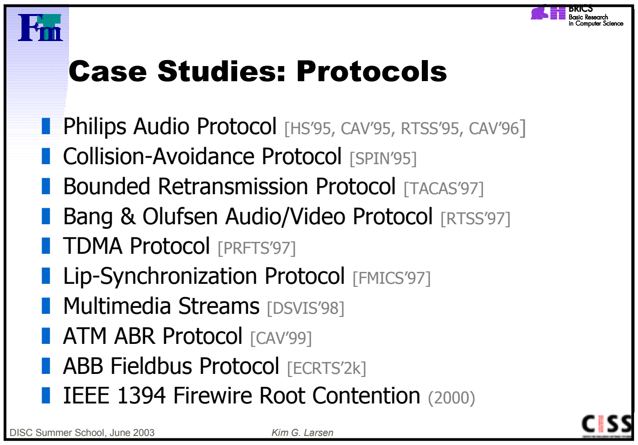
Course at DTU, Copenhagen


Production Cell



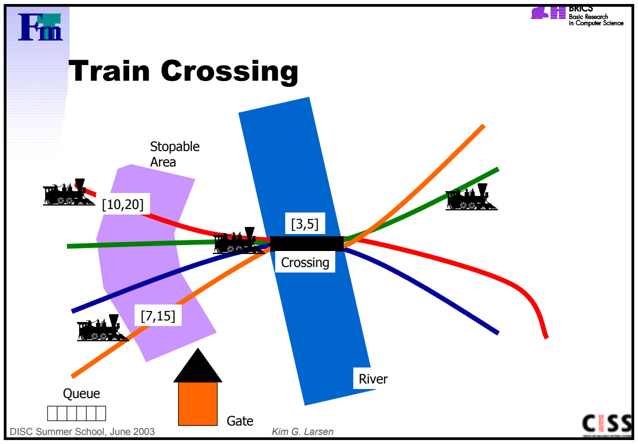
Case-Studies: Controllers

- 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, Memory Management for Radar (2001)

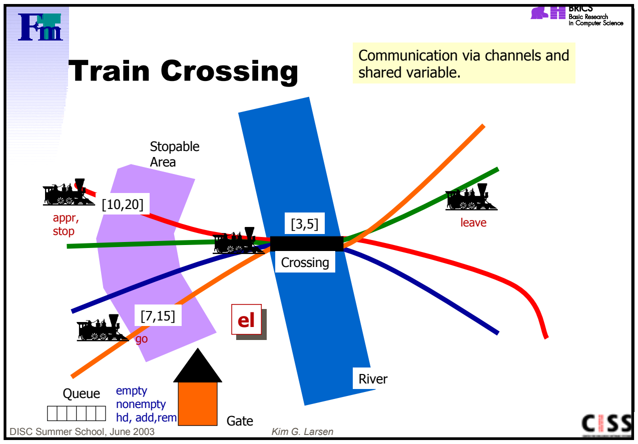


Case Studies: Protocols

- 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]
- ABB Fieldbus Protocol [ECRTS'2k]
- IEEE 1394 Firewire Root Contention (2000)

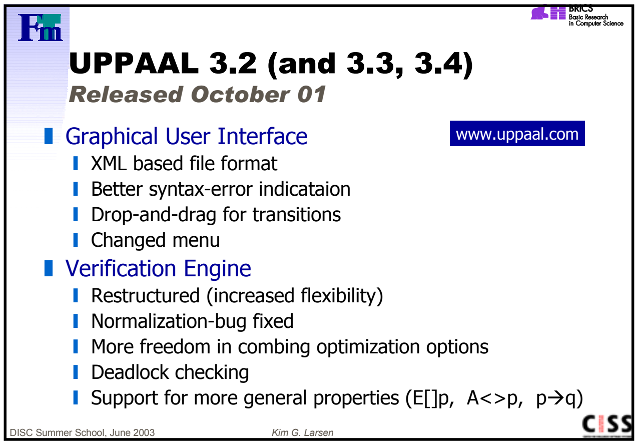


Train Crossing



Train Crossing

Communication via channels and shared variable.

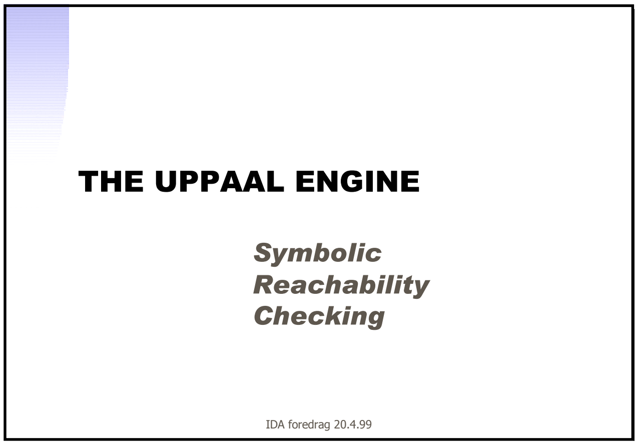


UPPAAL 3.2 (and 3.3, 3.4)

Released October 01

www.uppaal.com

- Graphical User Interface
  - XML based file format
  - Better syntax-error indicataion
  - Drop-and-drag for transitions
  - Changed menu
- Verification Engine
  - Restructured (increased flexibility)
  - Normalization-bug fixed
  - More freedom in combing optimization options
  - Deadlock checking
  - Support for more general properties (E[]p, A<>p, p→q)



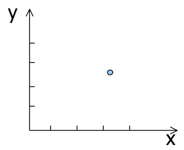
THE UPPAAL ENGINE

Symbolic Reachability Checking

Zones

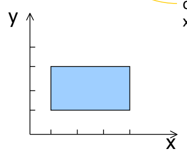
From infinite to finite

State
(n, x=3.2, y=2.5)

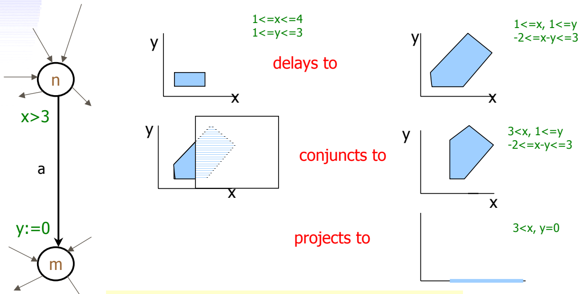


Symbolic state (set)
(n, 1≤x≤4,1≤y≤3)

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



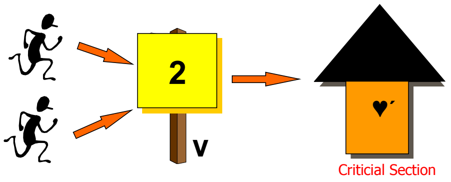
Symbolic Transitions



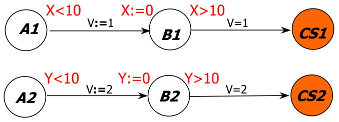
Thus (n,1<=x<=4,1<=y<=3) =a => (m,3<x, y=0)

Fischer's Protocol

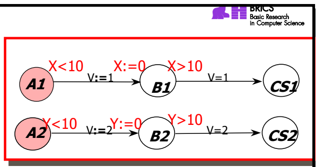
analysis using zones



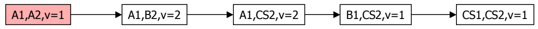
Init
V=1



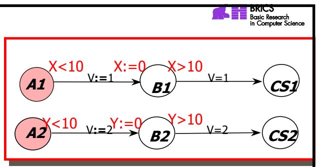
Fischers cont.



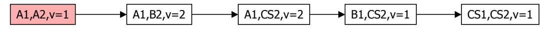
Untimed case


A1,A2,v=1 → A1,B2,v=2 → A1,CS2,v=2 → B1,CS2,v=1 → CS1,CS2,v=1

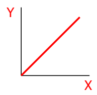
Fischers cont.



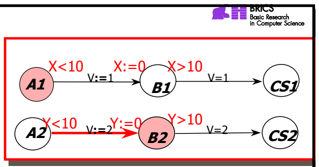
Untimed case


A1,A2,v=1 → A1,B2,v=2 → A1,CS2,v=2 → B1,CS2,v=1 → CS1,CS2,v=1

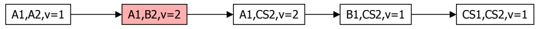
Taking time into account



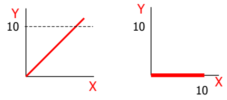
Fischers cont.

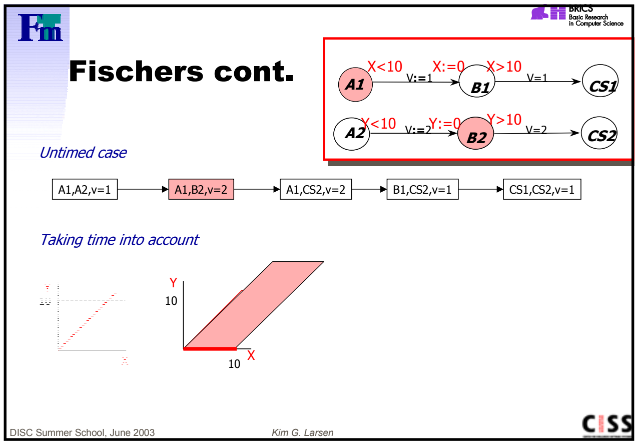


Untimed case


A1,A2,v=1 → A1,B2,v=2 → A1,CS2,v=2 → B1,CS2,v=1 → CS1,CS2,v=1

Taking time into account



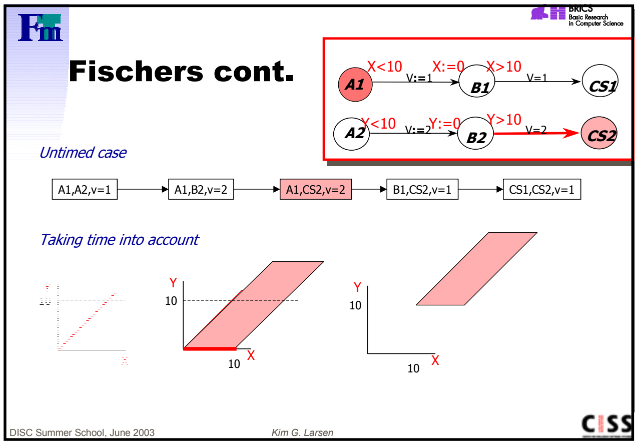


Fischers cont.

Untimed case

A1,A2,v=1 → A1,B2,v=2 → A1,CS2,v=2 → B1,CS2,v=1 → CS1,CS2,v=1

Taking time into account

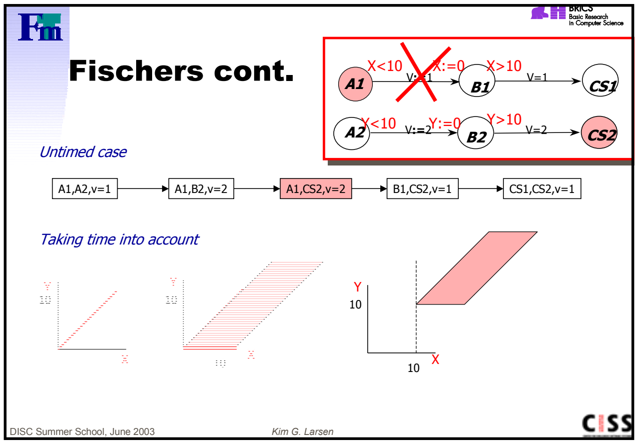


Fischers cont.

Untimed case

A1,A2,v=1 → A1,B2,v=2 → A1,CS2,v=2 → B1,CS2,v=1 → CS1,CS2,v=1

Taking time into account

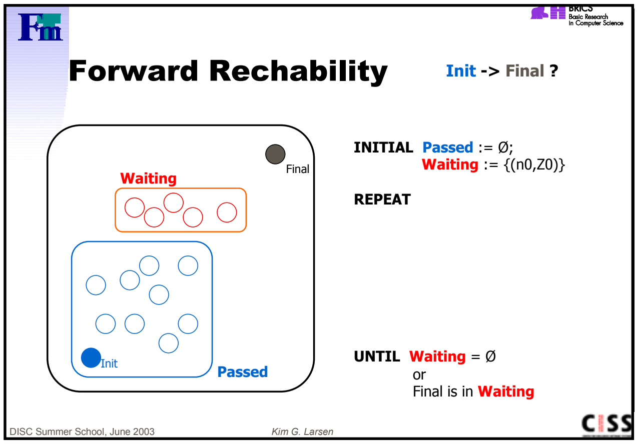


Fischers cont.

Untimed case

A1,A2,v=1 → A1,B2,v=2 → A1,CS2,v=2 → B1,CS2,v=1 → CS1,CS2,v=1

Taking time into account



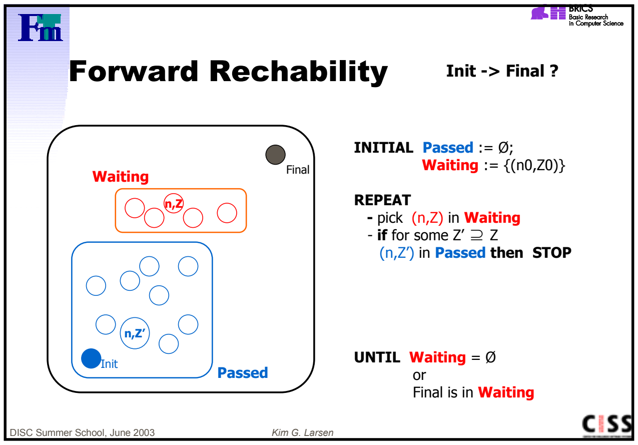
Forward Rechability

Init -> Final ?

```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT

UNTIL  Waiting = Ø
       or
       Final is in Waiting
```



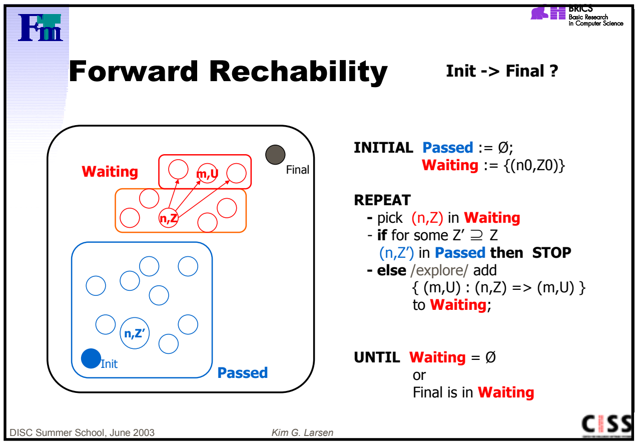
Forward Rechability

Init -> Final ?

```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT
  - pick  (n,Z) in Waiting
  - if for some Z' ⊇ Z
    (n,Z') in Passed then  STOP

UNTIL  Waiting = Ø
       or
       Final is in Waiting
```



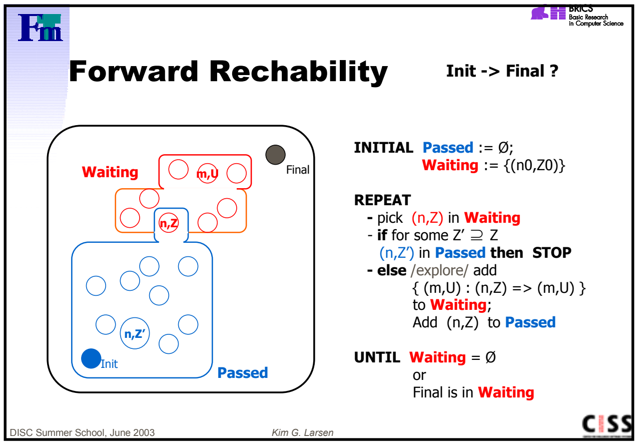
Forward Rechability

Init -> Final ?

```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT
  - pick  (n,Z) in Waiting
  - if for some Z' ⊇ Z
    (n,Z') in Passed then  STOP
  - else /explore/ add
        { (m,U) : (n,Z) => (m,U) }
        to Waiting;

UNTIL  Waiting = Ø
       or
       Final is in Waiting
```



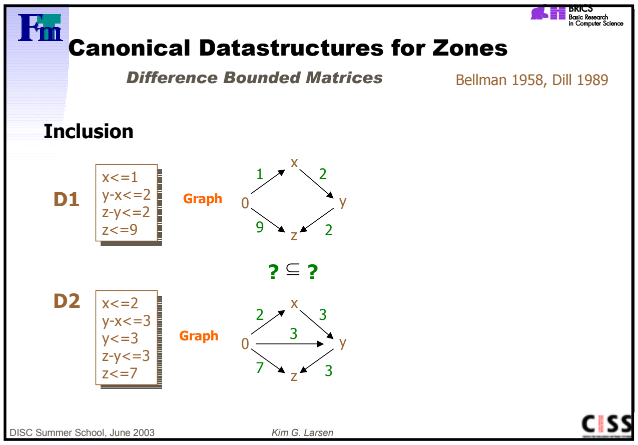
Forward Rechability

Init -> Final ?

```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT
  - pick (n,Z) in Waiting
  - if for some Z' ⊇ Z
      (n,Z') in Passed then STOP
  - else /explore/ add
         { (m,U) : (n,Z) => (m,U) }
         to Waiting;
         Add (n,Z) to Passed

UNTIL  Waiting = Ø
       or
       Final is in Waiting
```



Canonical Datastructures for Zones

Difference Bounded Matrices

Bellman 1958, Dill 1989

Inclusion

D1

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

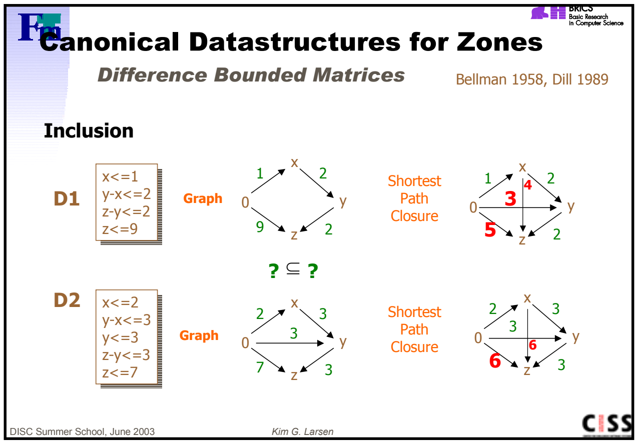
Graph

? ⊆ ?

D2

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

Graph



Canonical Datastructures for Zones

Difference Bounded Matrices

Bellman 1958, Dill 1989

Inclusion

D1

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

Graph

Shortest Path Closure

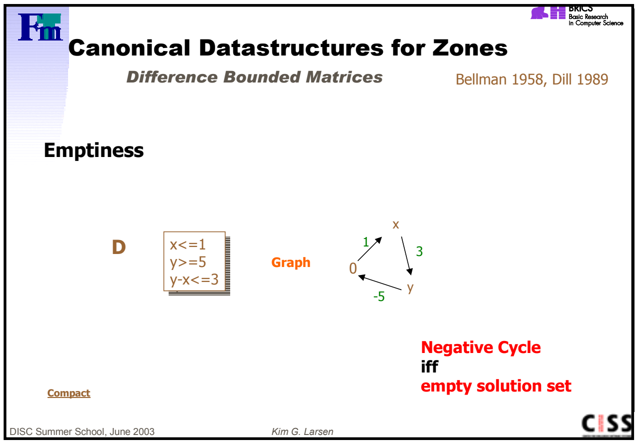
? ⊆ ?

D2

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

Graph

Shortest Path Closure



Canonical Datastructures for Zones

Difference Bounded Matrices

Bellman 1958, Dill 1989

Emptiness

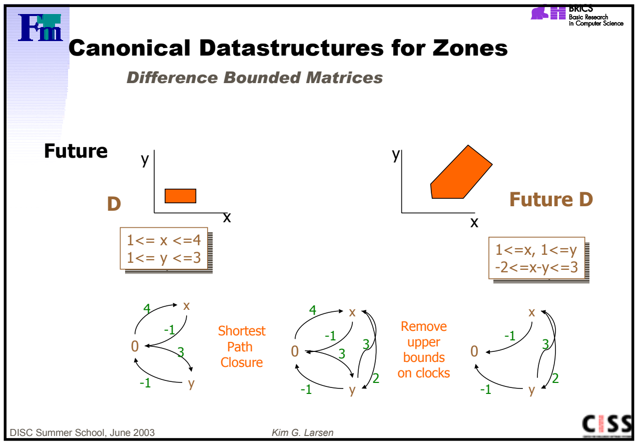
D

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

Graph

Negative Cycle
iff
empty solution set

Compact



Canonical Datastructures for Zones

Difference Bounded Matrices

Future

D

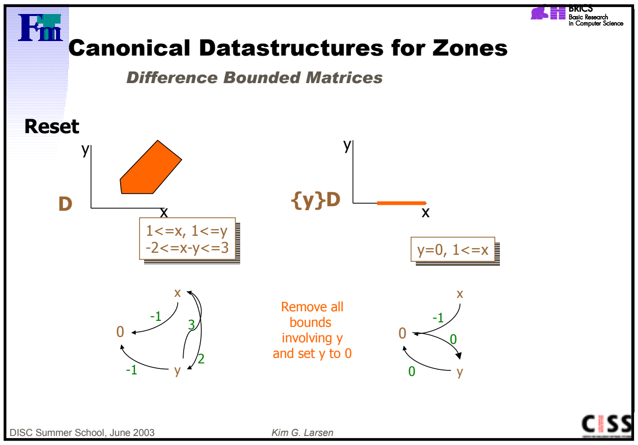
1<= x <=4
1<= y <=3

Future D

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

Shortest Path Closure

Remove upper bounds on clocks



Canonical Datastructures for Zones

Difference Bounded Matrices

Reset

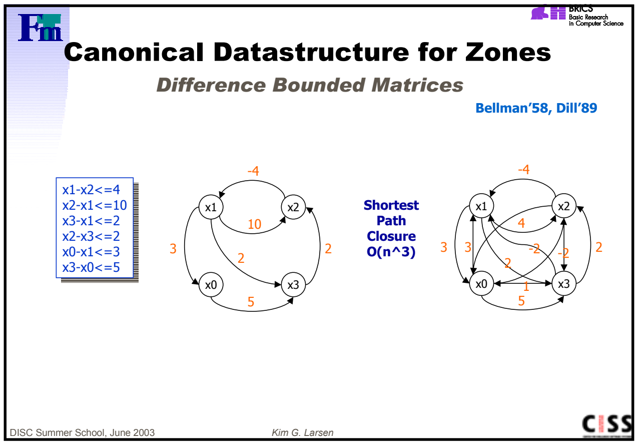
D

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

{y}D

y=0, 1<=x

Remove all bounds involving y and set y to 0



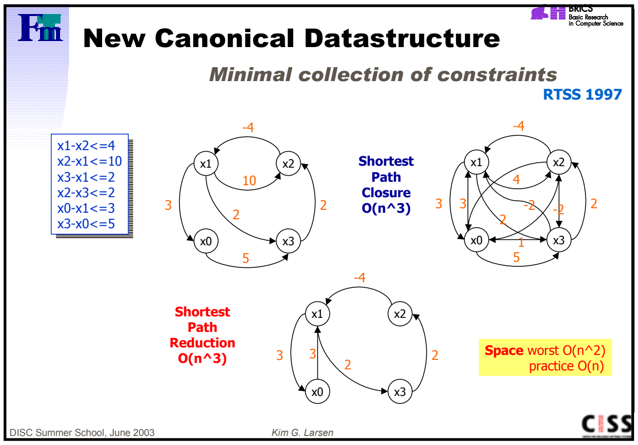
Canonical Datastructure for Zones

Difference Bounded Matrices

Bellman'58, Dill'89

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

Shortest Path Closure O(n^3)



New Canonical Datastructure

Minimal collection of constraints

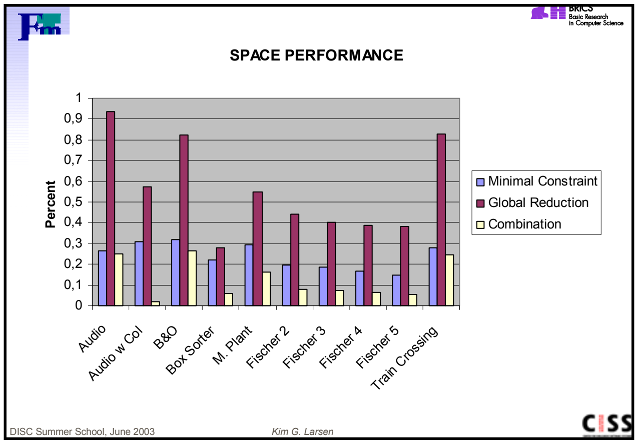
RTSS 1997

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

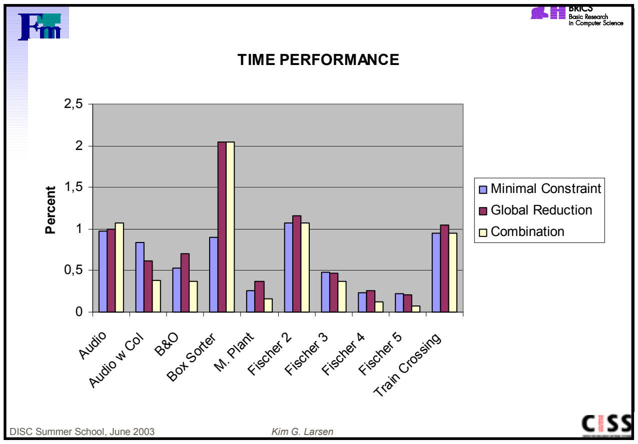
Shortest Path Closure O(n^3)

Shortest Path Reduction O(n^3)

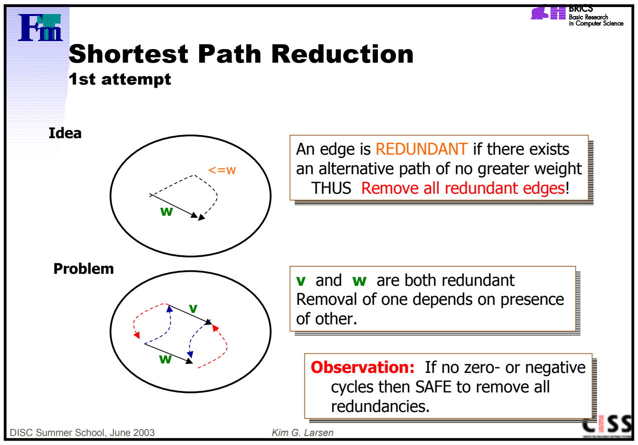
Space worst O(n^2)
practice O(n)



SPACE PERFORMANCE



TIME PERFORMANCE



Shortest Path Reduction

1st attempt

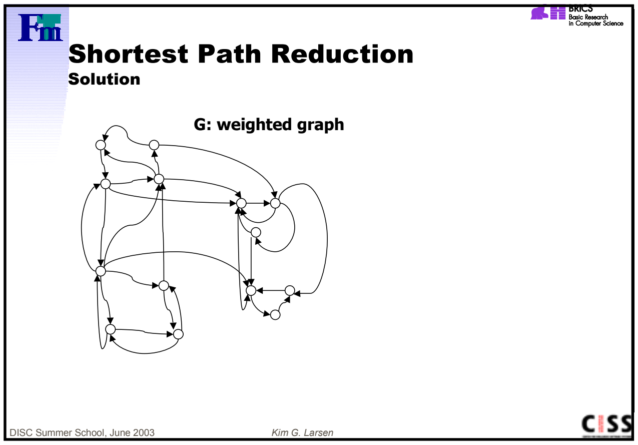
Idea

An edge is REDUNDANT if there exists an alternative path of no greater weight THUS Remove all redundant edges!

Problem

v and **w** are both redundant
Removal of one depends on presence of other.

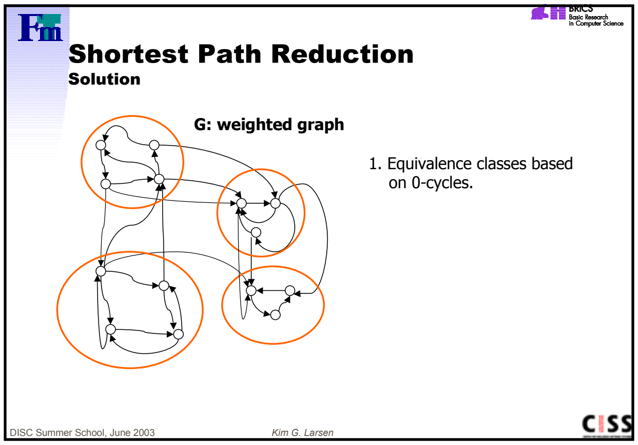
Observation: If no zero- or negative cycles then SAFE to remove all redundancies.



Shortest Path Reduction

Solution

G: weighted graph

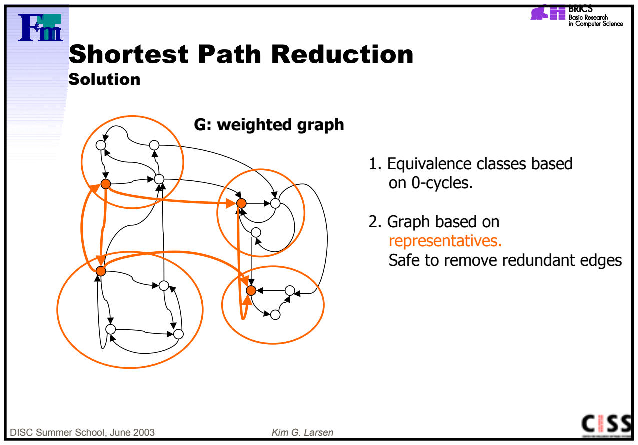


Shortest Path Reduction

Solution

G: weighted graph

1. Equivalence classes based on 0-cycles.

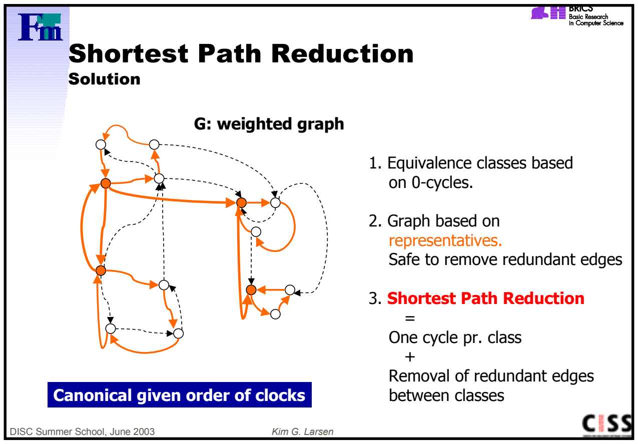


Shortest Path Reduction

Solution

G: weighted graph

1. Equivalence classes based on 0-cycles.
2. Graph based on representatives. Safe to remove redundant edges



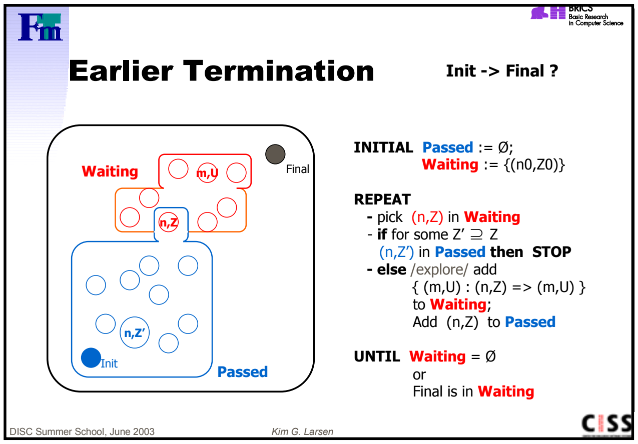
Shortest Path Reduction

Solution

G: weighted graph

1. Equivalence classes based on 0-cycles.
2. Graph based on representatives. Safe to remove redundant edges
3. **Shortest Path Reduction**
   =
   One cycle pr. class
   +
   Removal of redundant edges between classes

Canonical given order of clocks



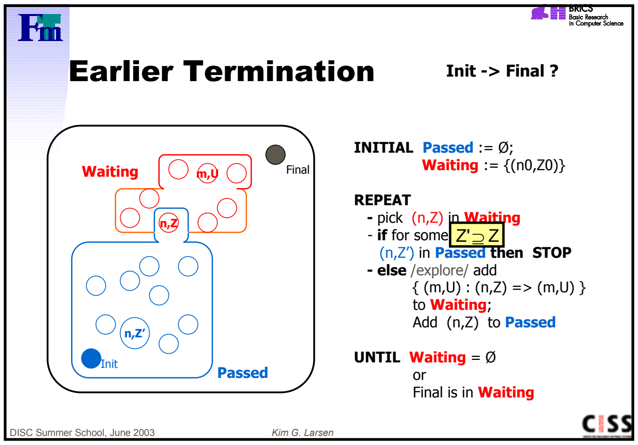
Earlier Termination

Init -> Final ?

```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT
  - pick  (n,Z) in Waiting
  - if for some Z' ⊒ Z
      (n,Z') in Passed then  STOP
  - else /explore/ add
        { (m,U) : (n,Z) => (m,U) }
        to Waiting;
        Add  (n,Z)  to Passed

UNTIL  Waiting = Ø
        or
        Final is in Waiting
```



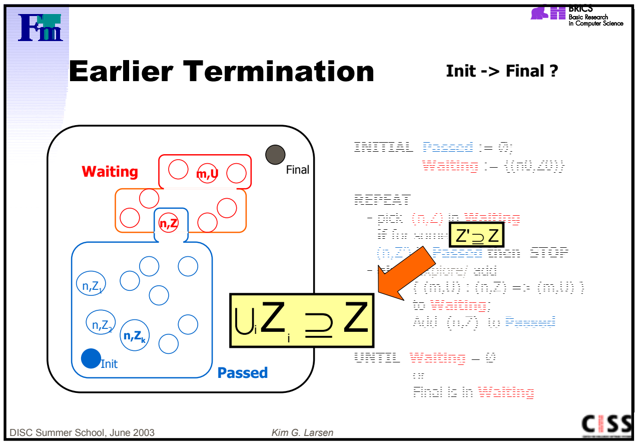
Earlier Termination

Init -> Final ?

```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT
  - pick  (n,Z) in Waiting
  - if for some Z'⊒Z
      (n,Z') in Passed then  STOP
  - else /explore/ add
        { (m,U) : (n,Z) => (m,U) }
        to Waiting;
        Add  (n,Z)  to Passed

UNTIL  Waiting = Ø
        or
        Final is in Waiting
```



Earlier Termination

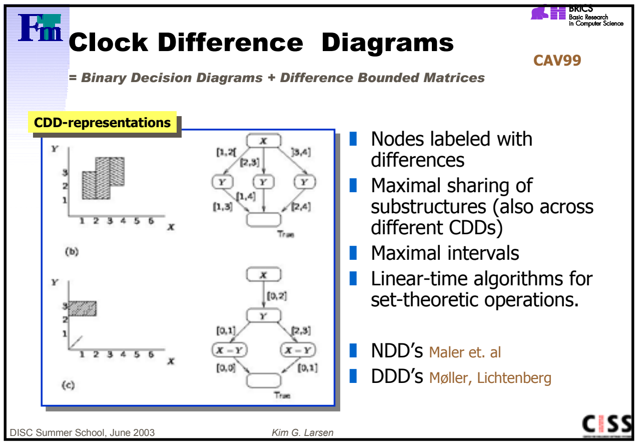
Init -> Final ?

$\bigcup_i Z_i \supseteq Z$

```
INITIAL  Passed := Ø;
         Waiting := {(n0,Z0)}

REPEAT
  - pick  (n,Z) in Waiting
  - if for some Z'⊒Z
      (n,Z') in Passed then  STOP
  - else /explore/ add
        { (m,U) : (n,Z) => (m,U) }
        to Waiting;
        Add  (n,Z)  to Passed

UNTIL  Waiting = Ø
        or
        Final is in Waiting
```



BRICS Basic Research in Computer Science

Clock Difference Diagrams

CAV99

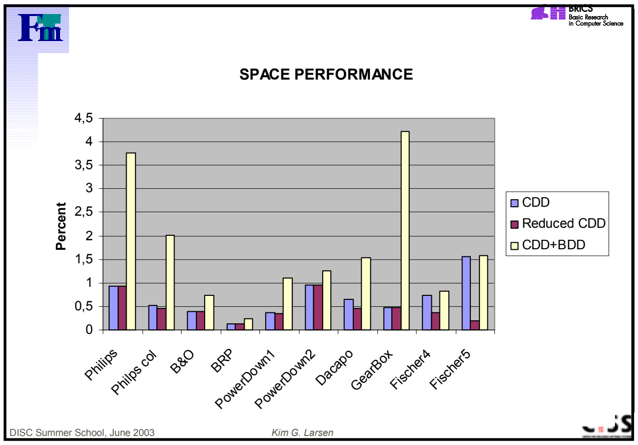
= Binary Decision Diagrams + Difference Bounded Matrices

CDD-representations

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

- NDD's Maler et. al
- DDD's Møller, Lichtenberg

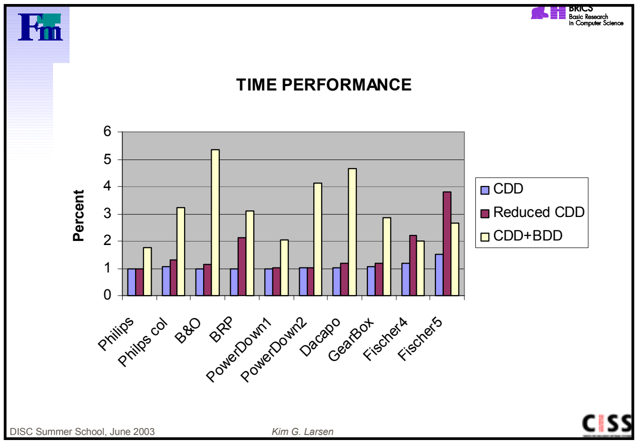
DISC Summer School, June 2003 *Kim G. Larsen*



BRICS Basic Research in Computer Science

SPACE PERFORMANCE

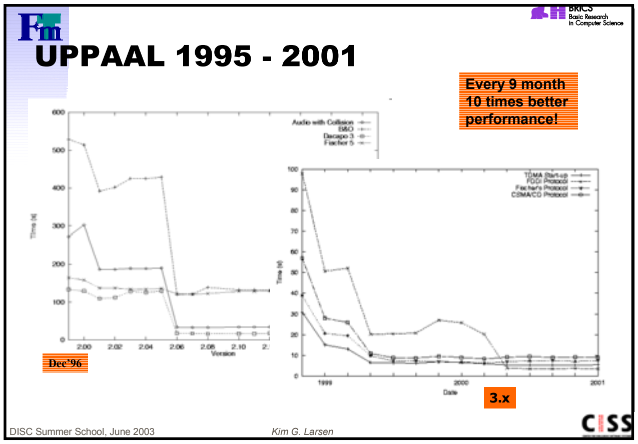
DISC Summer School, June 2003 *Kim G. Larsen*



BRICS Basic Research in Computer Science

TIME PERFORMANCE

DISC Summer School, June 2003 *Kim G. Larsen*



BRICS Basic Research in Computer Science

UPPAAL 1995 - 2001

Every 9 month 10 times better performance!

DISC Summer School, June 2003 *Kim G. Larsen*