UPPAAL 4.0.8

Engine & Formalism

Kim G. Larsen

Outline

- UPPAAL Models
  & Specifications
- UPPAAL Engine
  - Zones, CDDs
- UPPAAL Options

- LAB Exercises

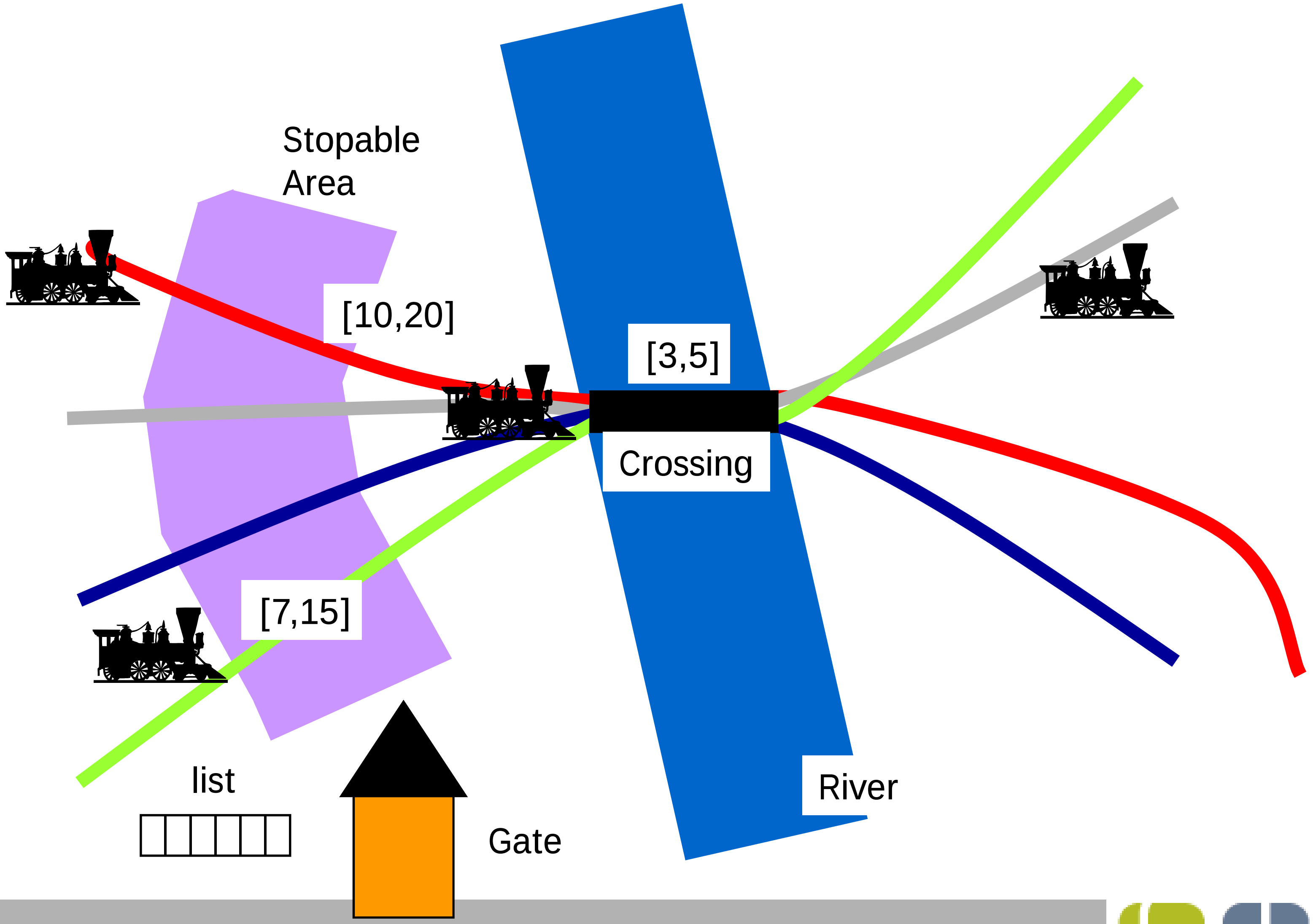
Copyright 1995-2006 by Uppsala University and Aalborg University. All rights reserved.
More information at http://www.uppaal.com

UPPAAL 4.0.2 (rev. 2491), August 2006.

UPPAAL

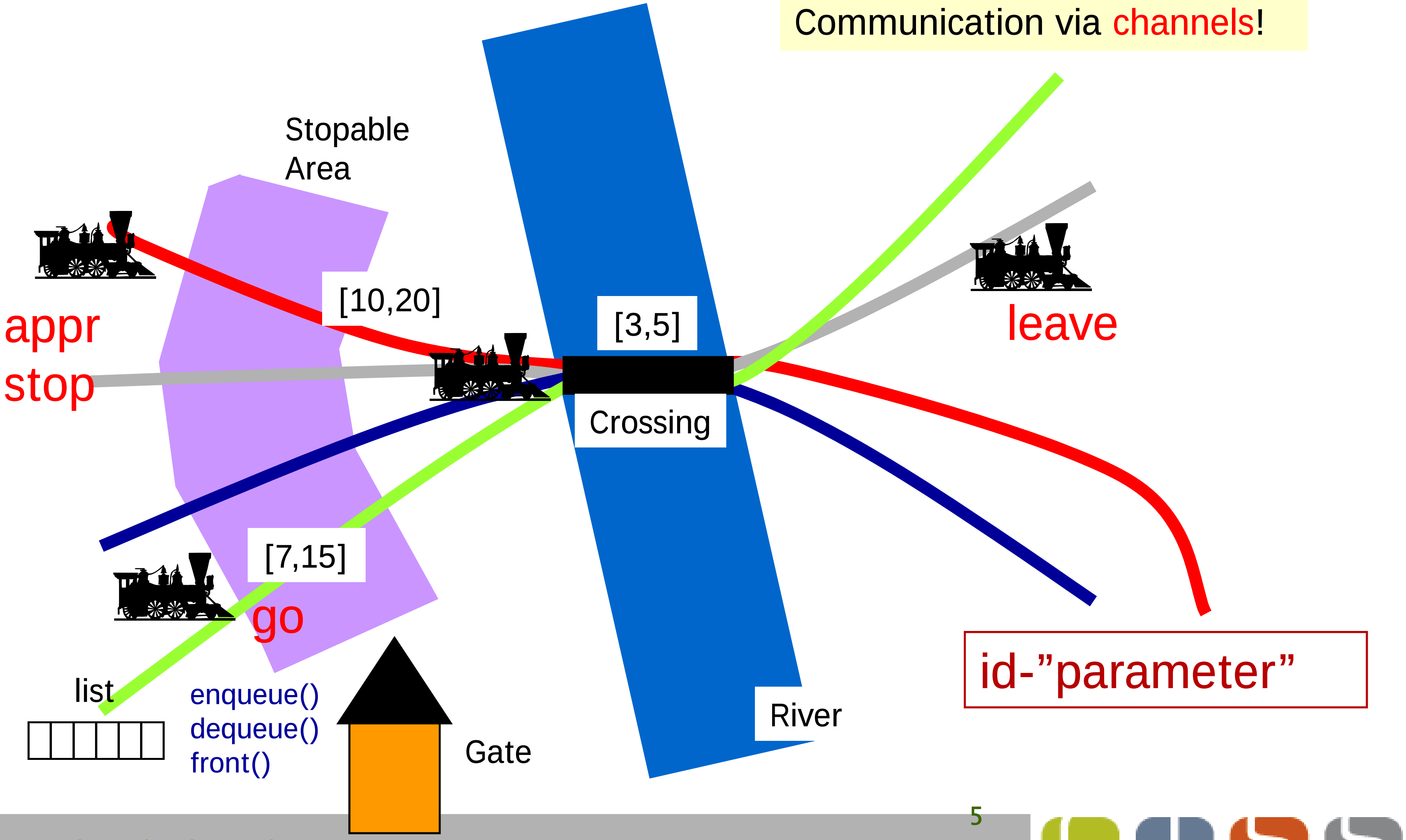
Modeling & Specification

Train Crossing

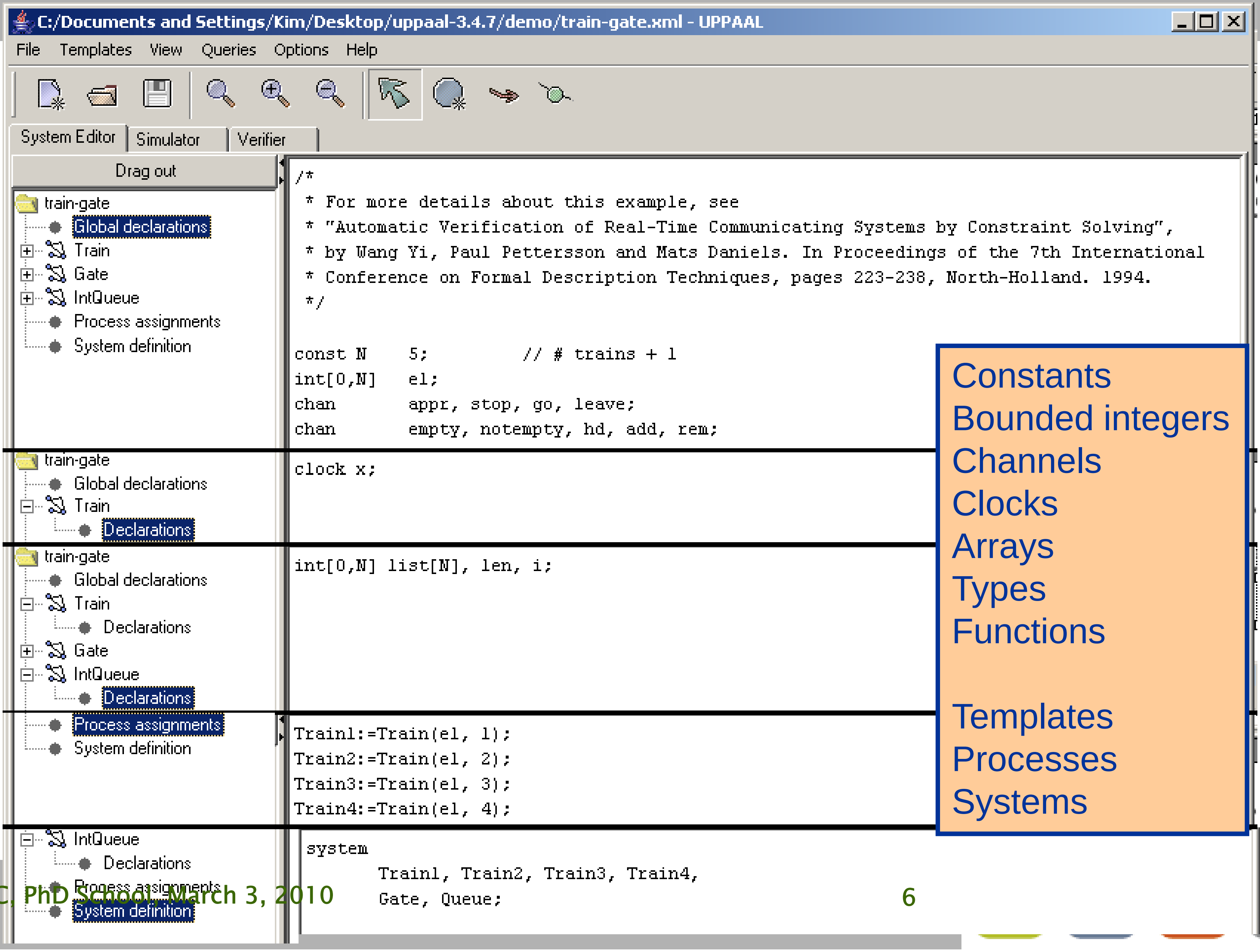


Train Crossing

Communication via channels!



Declarations



```
/*
 * For more details about this example, see
 * "Automatic Verification of Real-Time Communicating Systems by Constraint Solving",
 * by Wang Yi, Paul Pettersson and Mats Daniels. In Proceedings of the 7th International
 * Conference on Formal Description Techniques, pages 223-238, North-Holland. 1994.
 */

const N     5;          // # trains + 1
int[0,N]    el;
chan        appr, stop, go, leave;
chan        empty, notempty, hd, add, rem;
```

```
clock x;
```

```
int[0,N] list[N], len, i;
```

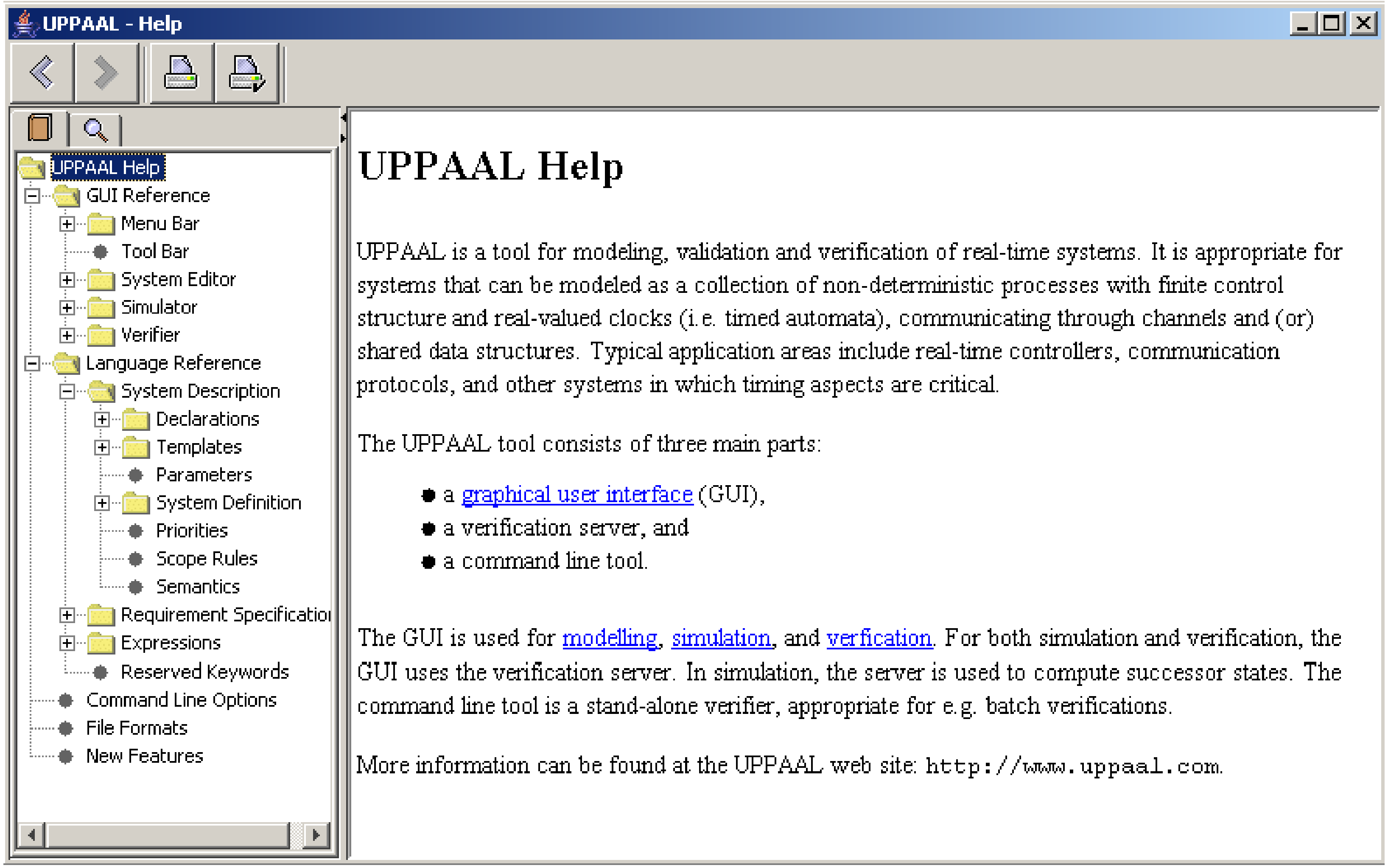
```
Train1:=Train(el, 1);
Train2:=Train(el, 2);
Train3:=Train(el, 3);
Train4:=Train(el, 4);
```

```
system
       Train1, Train2, Train3, Train4,
       Gate, Queue;
```

Constants
Bounded integers
Channels
Clocks
Arrays
Types
Functions

Templates
Processes
Systems

UPPAAL Help



UPPAAL Help

UPPAAL is a tool for modeling, validation and verification of real-time systems. It is appropriate for systems that can be modeled as a collection of non-deterministic processes with finite control structure and real-valued clocks (i.e. timed automata), communicating through channels and (or) shared data structures. Typical application areas include real-time controllers, communication protocols, and other systems in which timing aspects are critical.

The UPPAAL tool consists of three main parts:

- a graphical user interface (GUI),
- a verification server, and
- a command line tool.

The GUI is used for modelling, simulation, and verfication. For both simulation and verification, the GUI uses the verification server. In simulation, the server is used to compute successor states. The command line tool is a stand-alone verifier, appropriate for e.g. batch verifications.

More information can be found at the UPPAAL web site: `http://www.uppaal.com`.

Logical Specifications

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

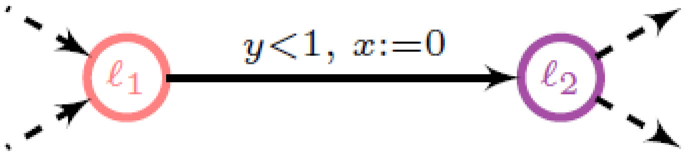
The expressions P and Q must be type safe, side effect free, and evaluate to a boolean.

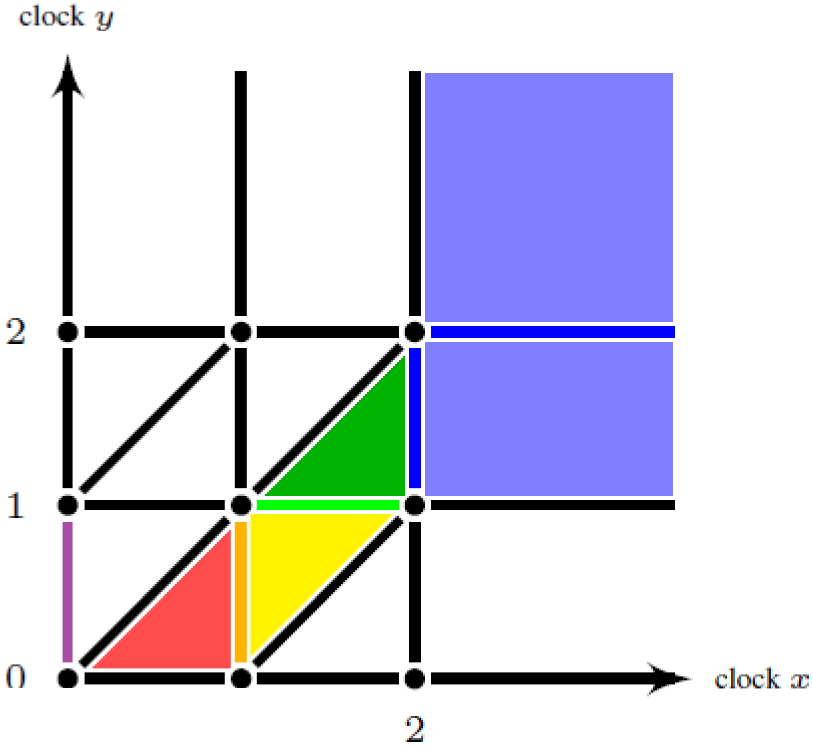
Only references to integer variables, constants, clocks, are allowed (and arrays of these).

UPPAAL

ENGINE

Regions – From Infinite to Finite

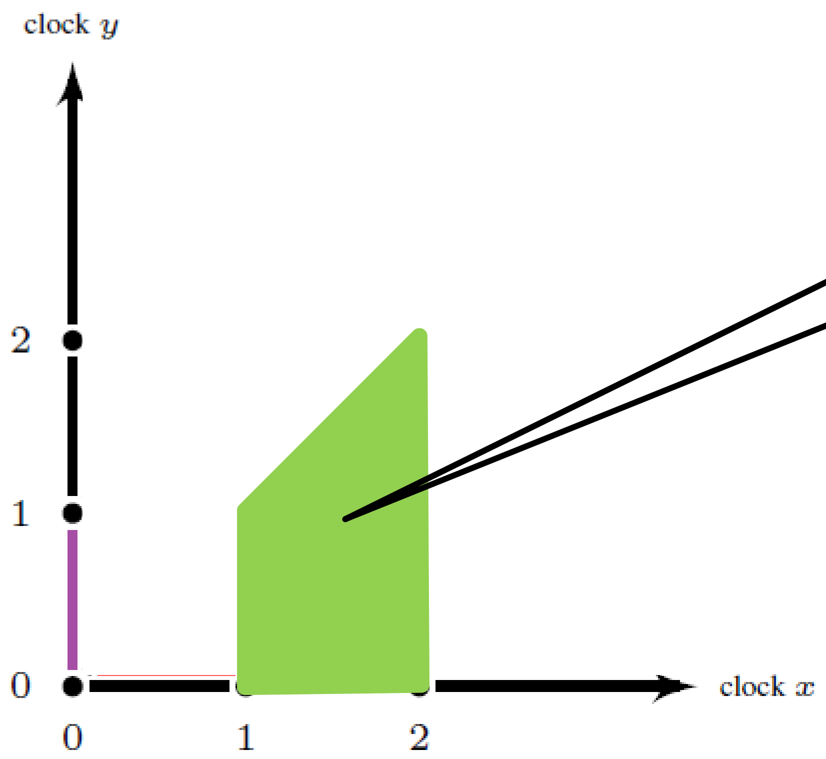




Theorem

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

Zones – From Finite to Efficiency



A zone **Z**:

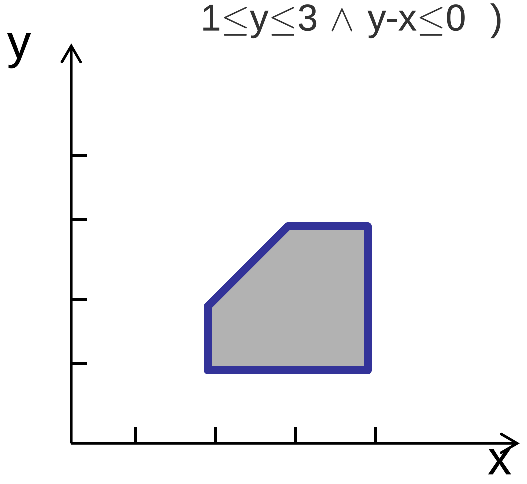
$1 \leq x \leq 2 \quad \wedge$

$0 \leq y \leq 2 \quad \wedge$

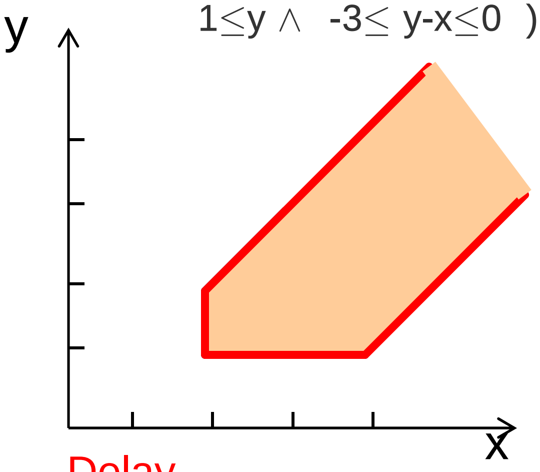
$x - y \geq 0$

Zones – Operations

$(n,\ 2\leq x\leq 4 \wedge$
$1\leq y\leq 3 \wedge y\text{-}x\leq 0\ )$

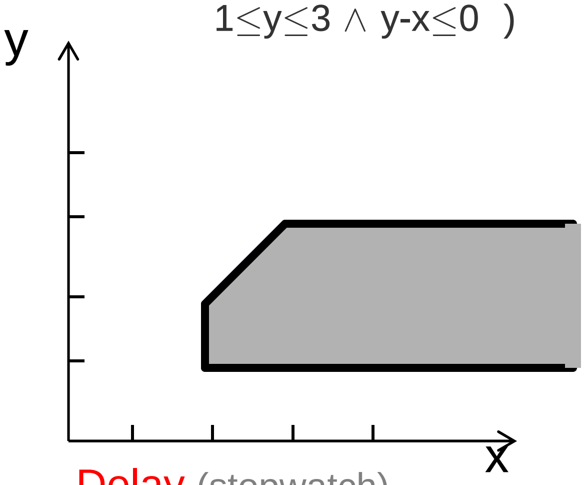


$(n,\ 2\leq x \wedge$
$1\leq y \wedge\ -3\leq y\text{-}x\leq 0\ )$



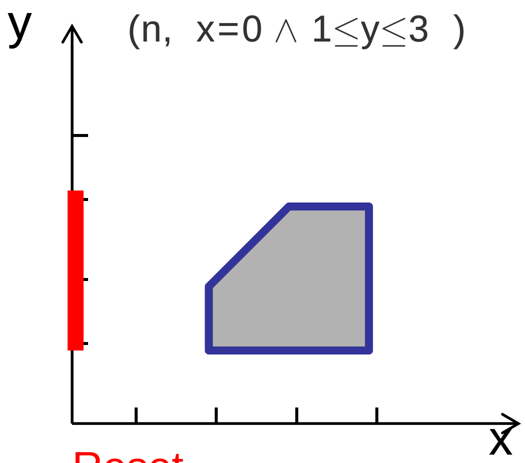
Delay

$(n,\ 2\leq x \wedge$
$1\leq y\leq 3 \wedge y\text{-}x\leq 0\ )$



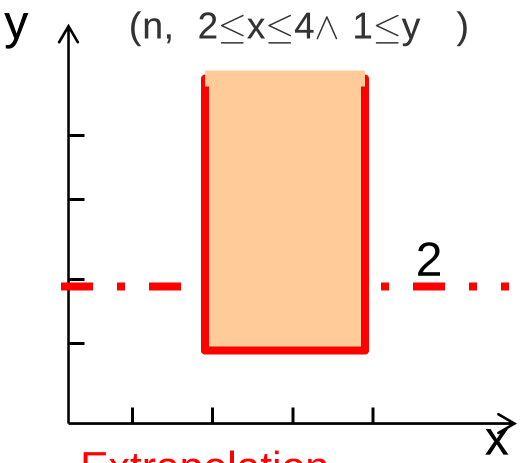
Delay (stopwatch)

$(n,\ x=0 \wedge 1\leq y\leq 3\ )$

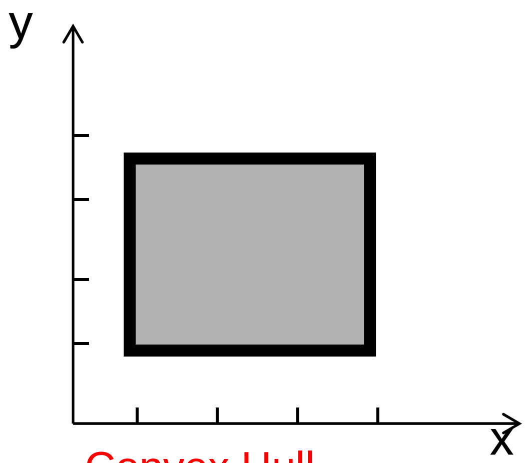


Reset

$(n,\ 2\leq x\leq 4\wedge 1\leq y\ )$

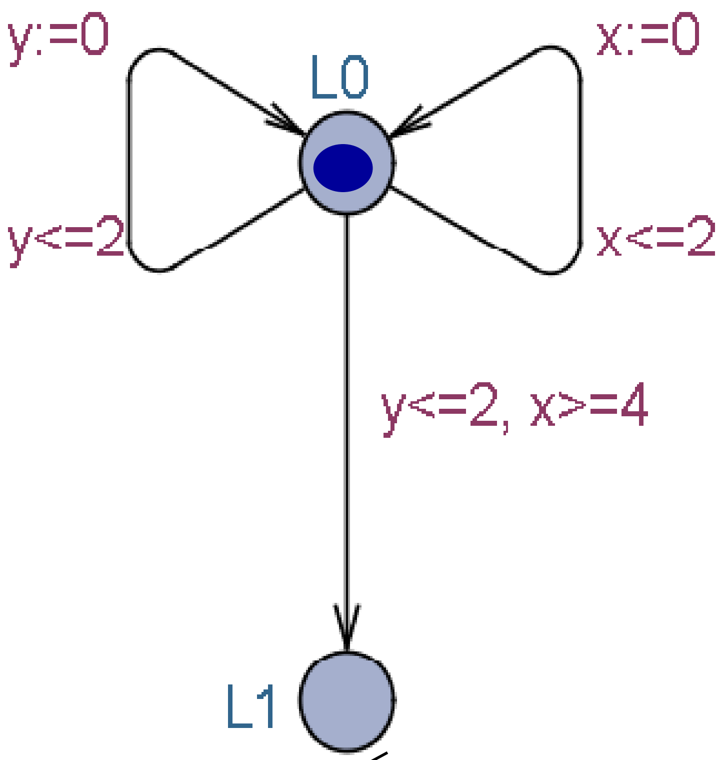


Extrapolation

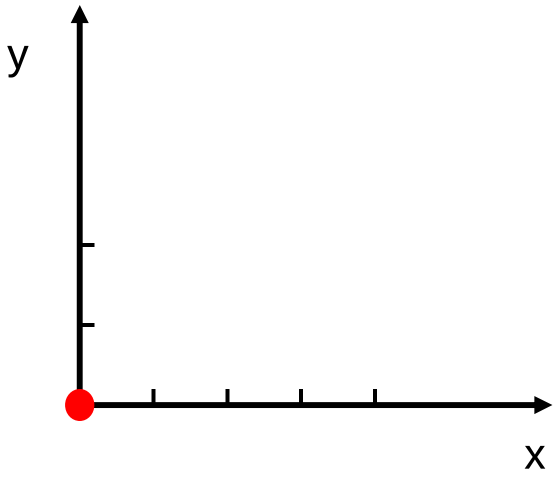


Convex Hull

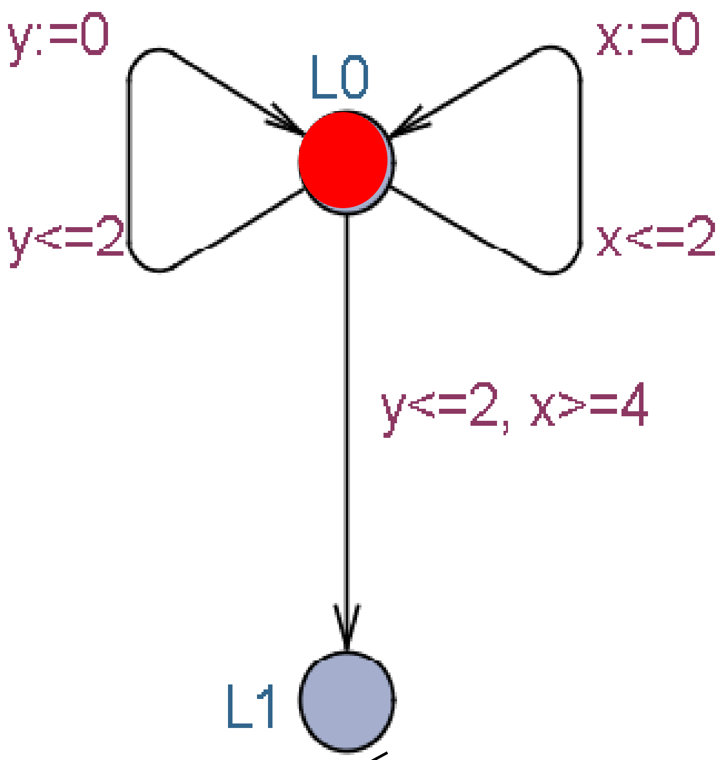
Symbolic Exploration



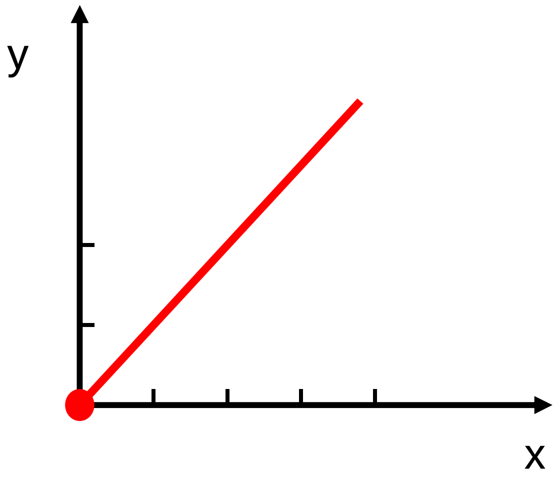
Reachable?



Symbolic Exploration

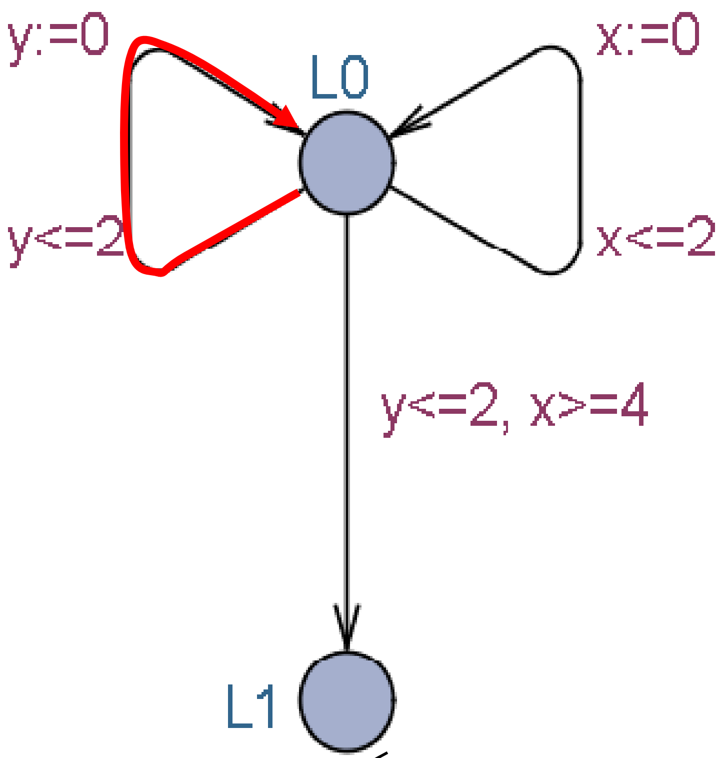


Reachable?

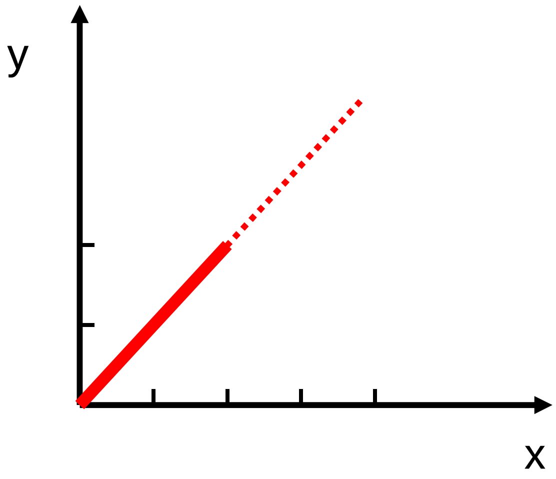


Delay

Symbolic Exploration

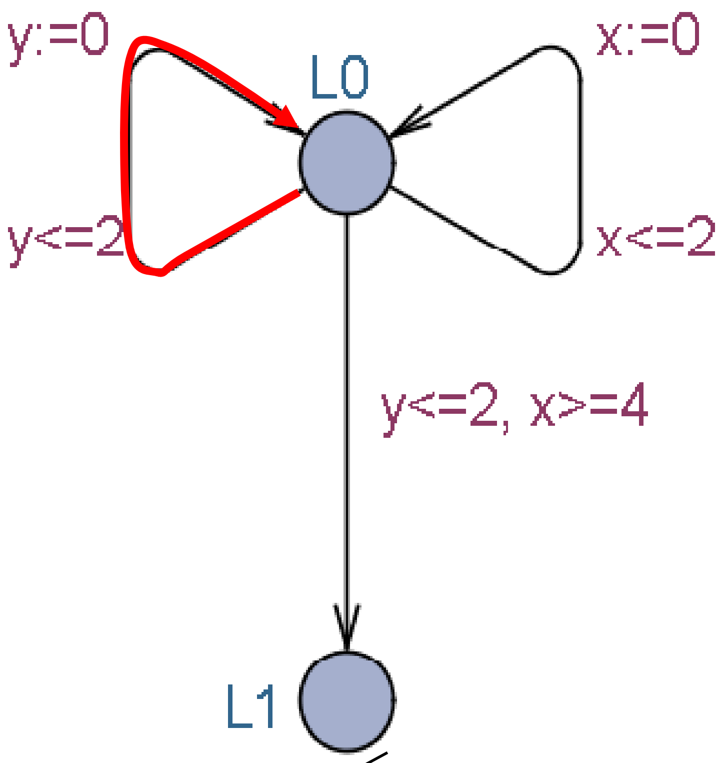


Reachable?

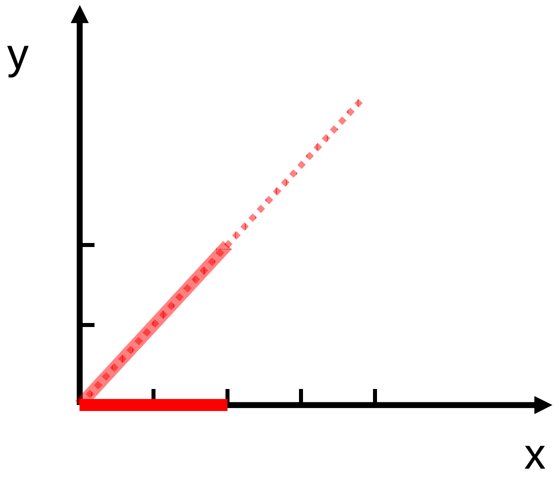


Left

Symbolic Exploration

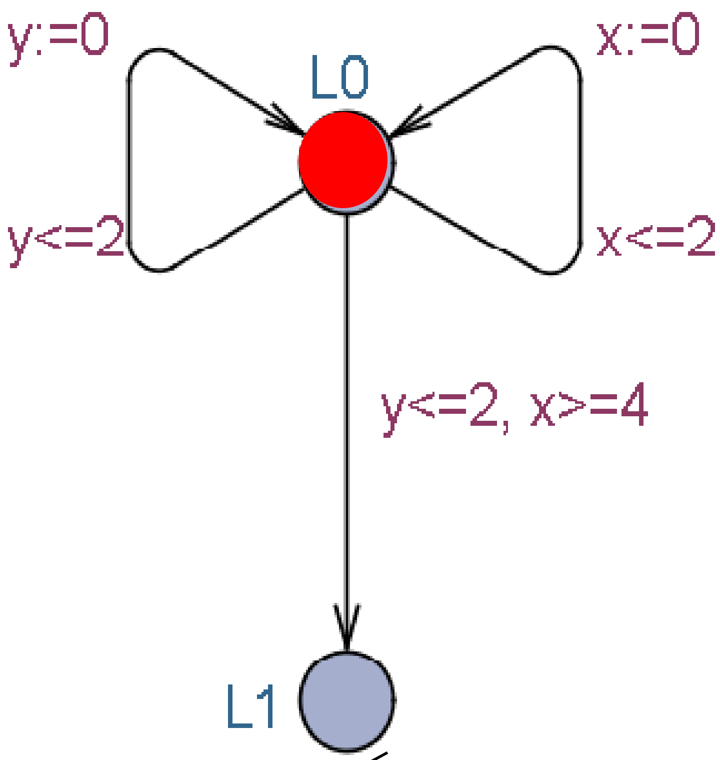


Reachable?

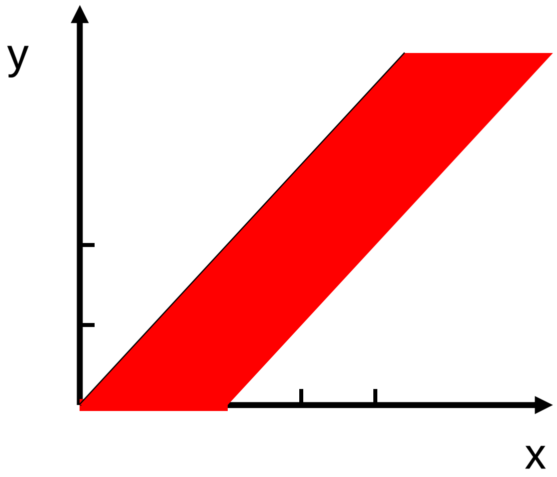


Left

Symbolic Exploration

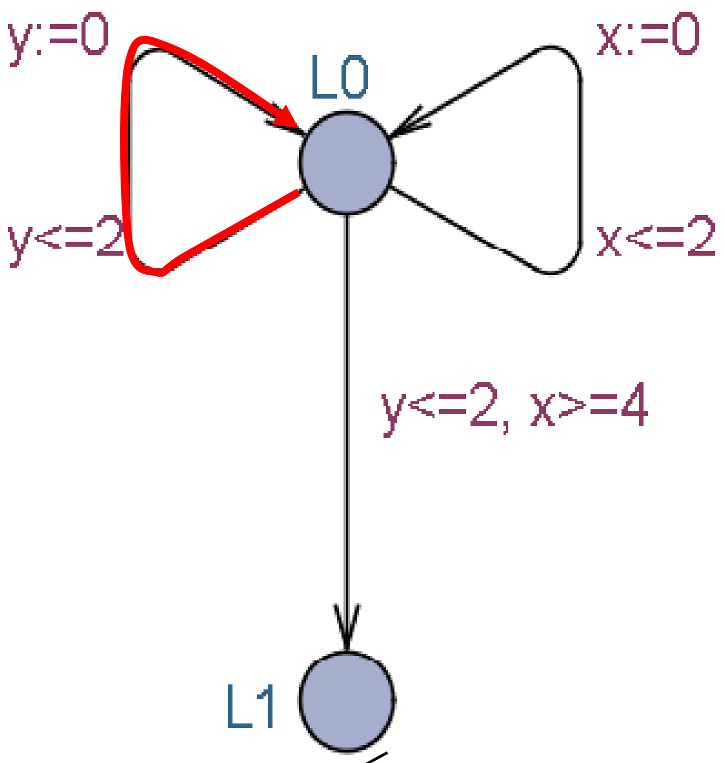


Reachable?

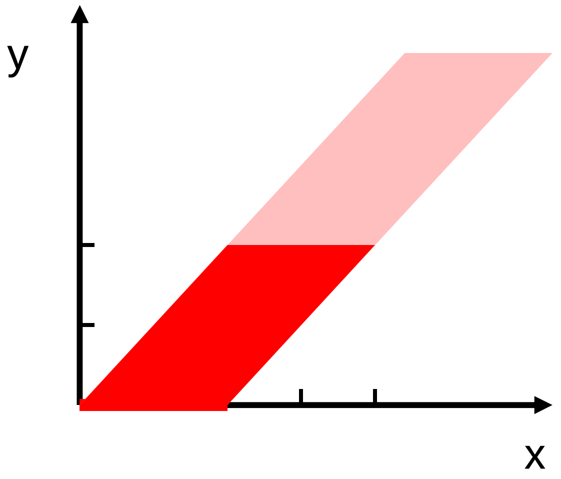


Delay

Symbolic Exploration

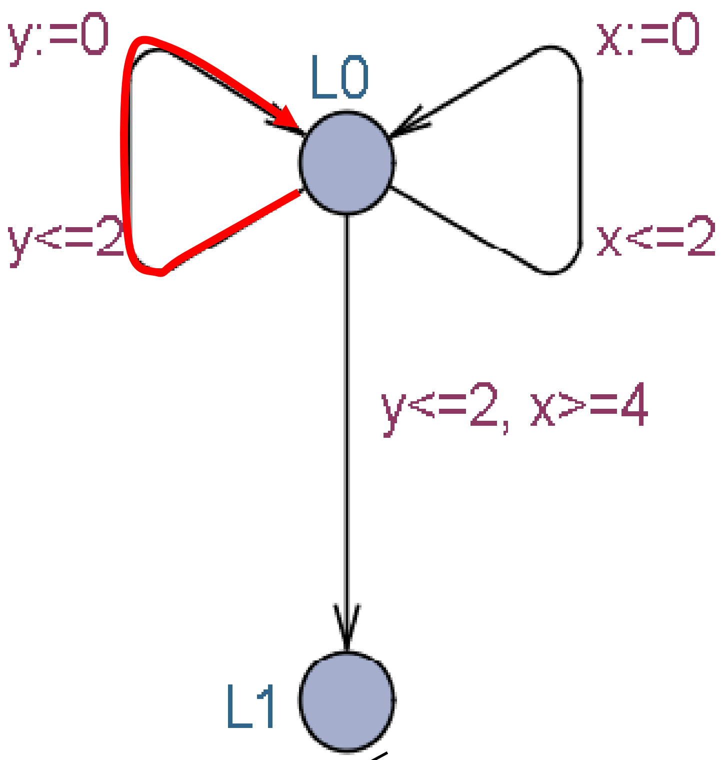


Reachable?

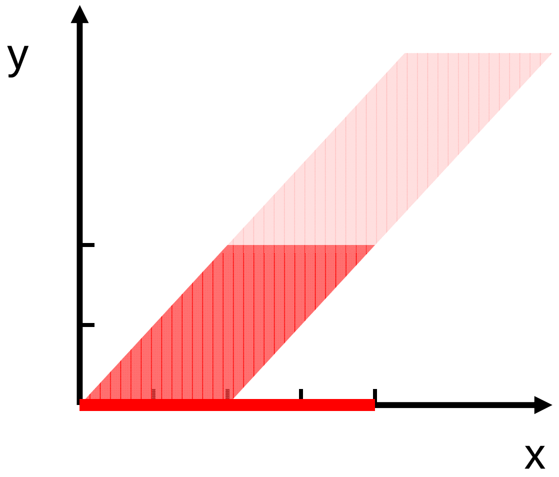


Left

Symbolic Exploration

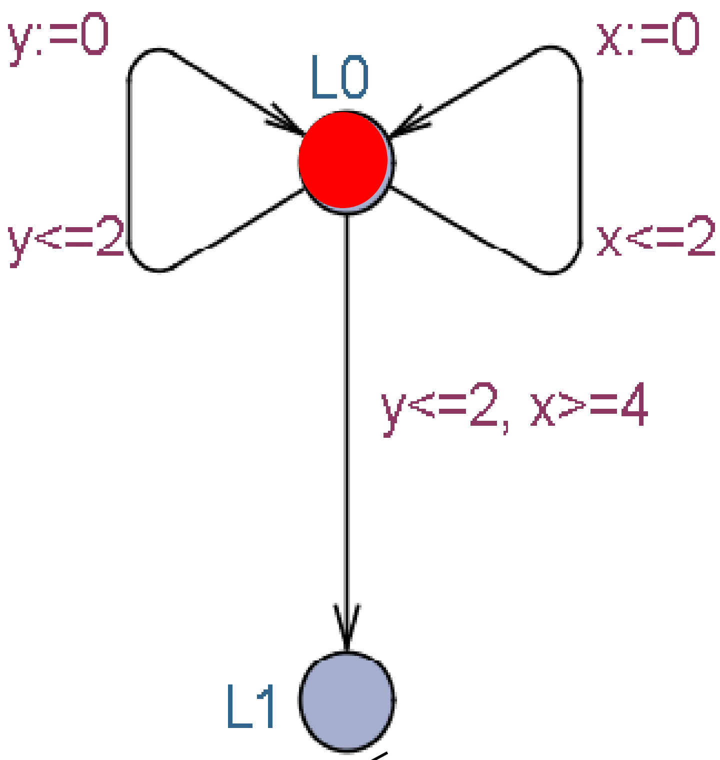


Reachable?

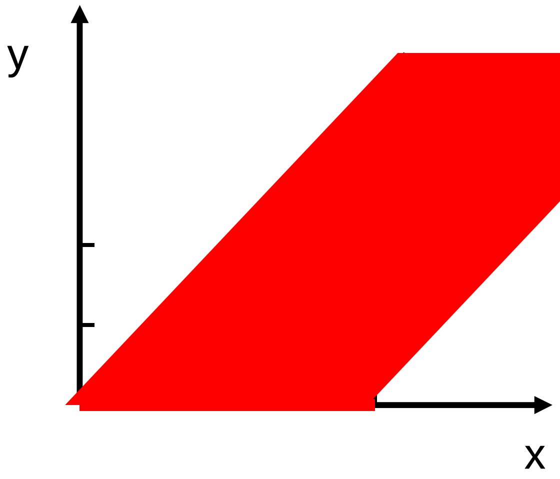


Left

Symbolic Exploration

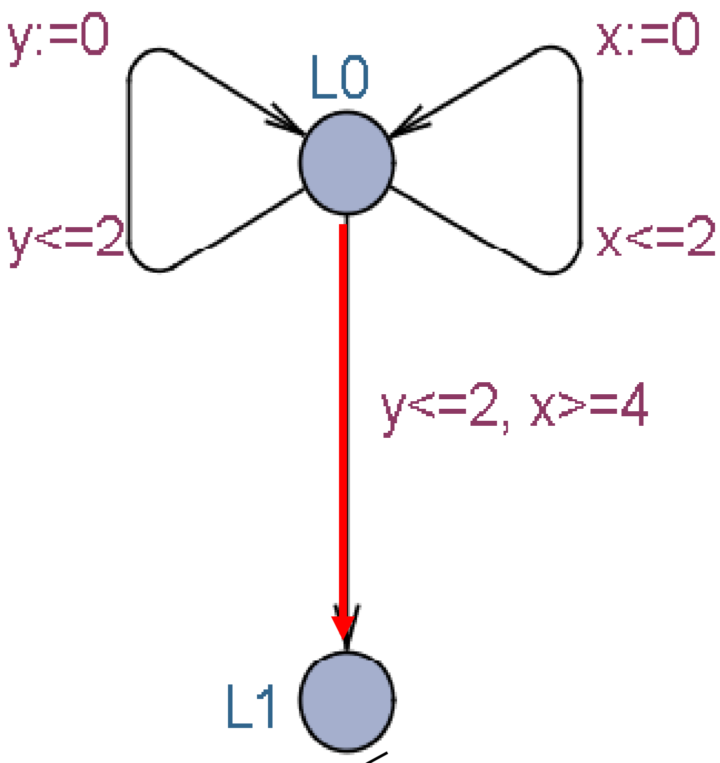


Reachable?

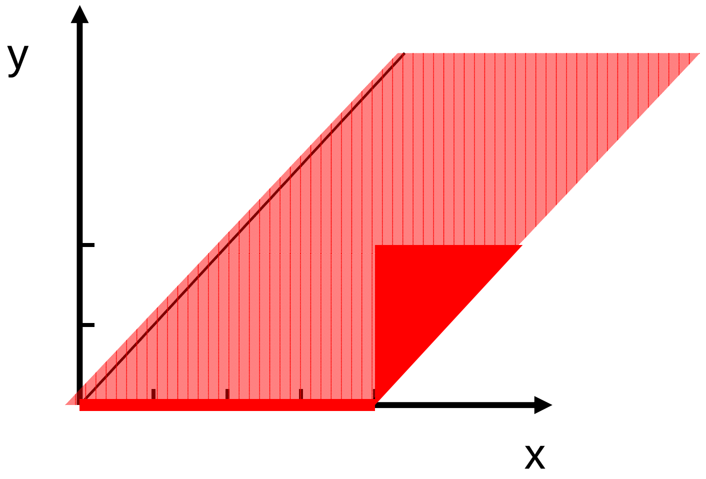


Delay

Symbolic Exploration



Reachable?



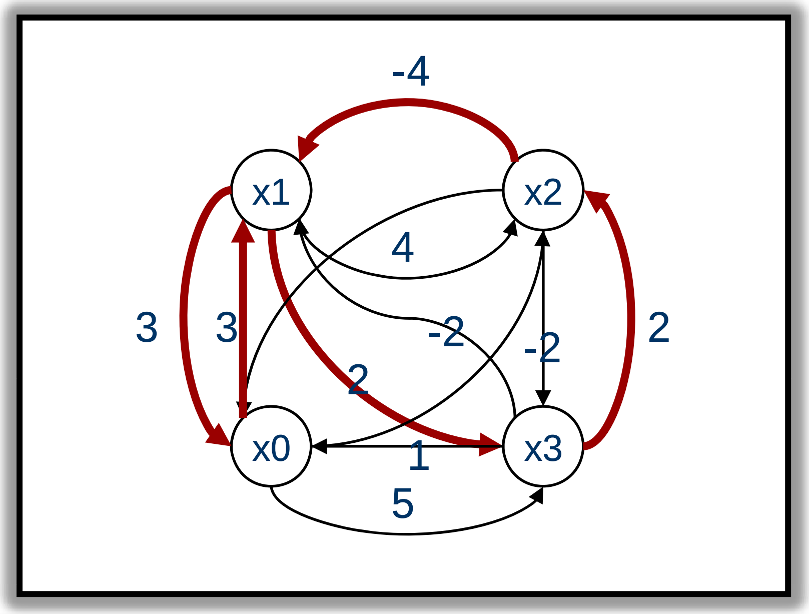
Down

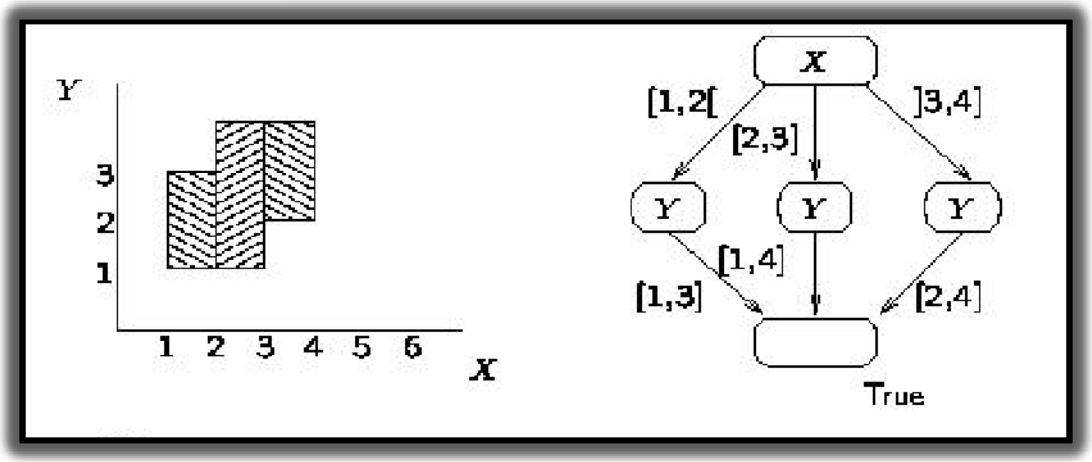
Datastructures for Zones

- Difference Bounded Matrices (DBMs)
- Minimal Constraint Form

  [RTSS97]
- Clock Difference Diagrams

  [CAV99]





Inclusion Checking (DBMs)

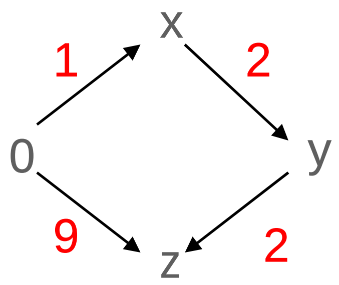
Bellman 1958, Dill 1989

Inclusion

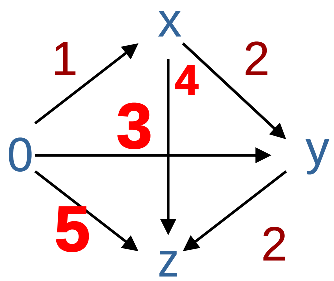
D1

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

Graph



Shortest Path Closure

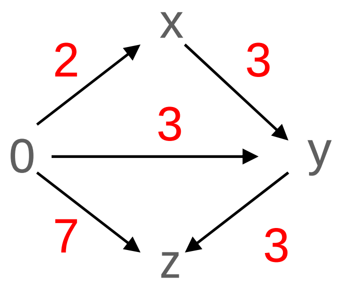


$? \subseteq ?$

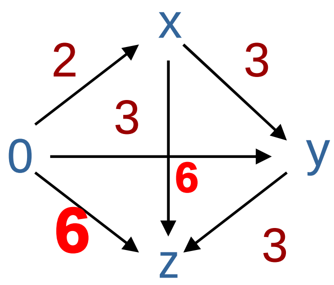
D2

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

Graph

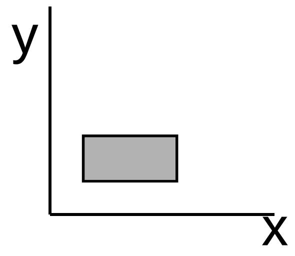


Shortest Path Closure

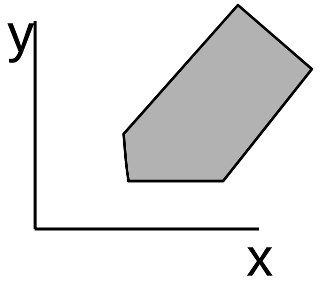


Future (DBMs)

D

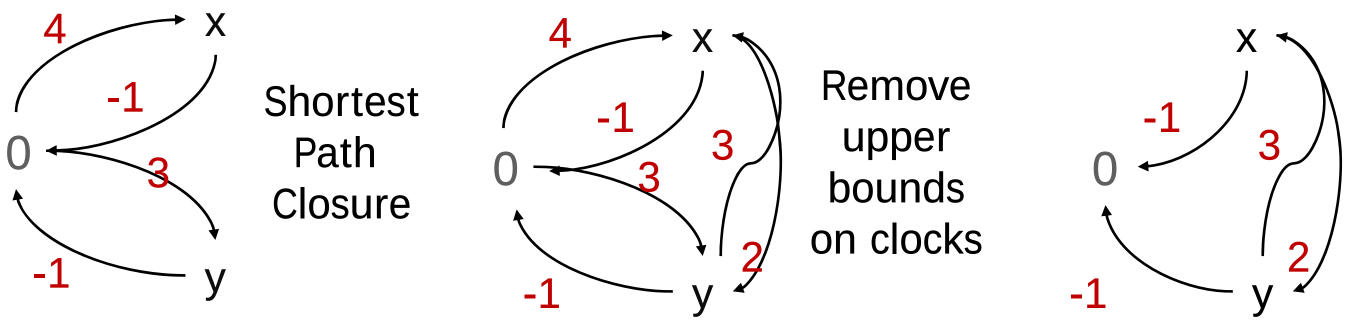


1<= x <=4
1<= y <=3



Future D

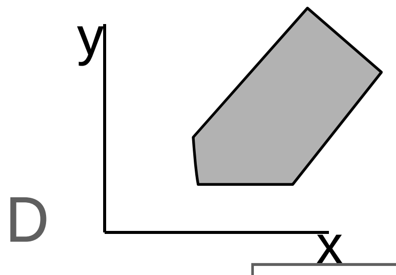
1<=x, 1<=y
-2<=x-y<=3



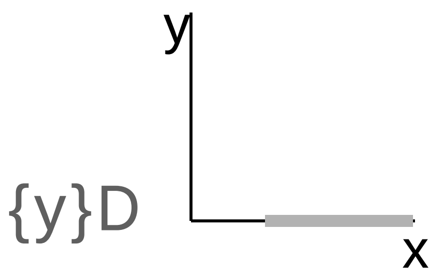
Shortest Path Closure

Remove upper bounds on clocks

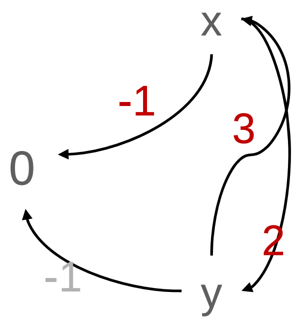
Reset (DBMs)



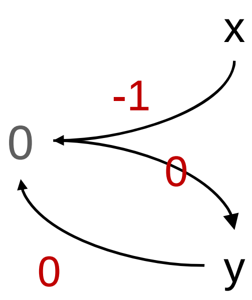
1<=x, 1<=y
-2<=x-y<=3



y=0, 1<=x



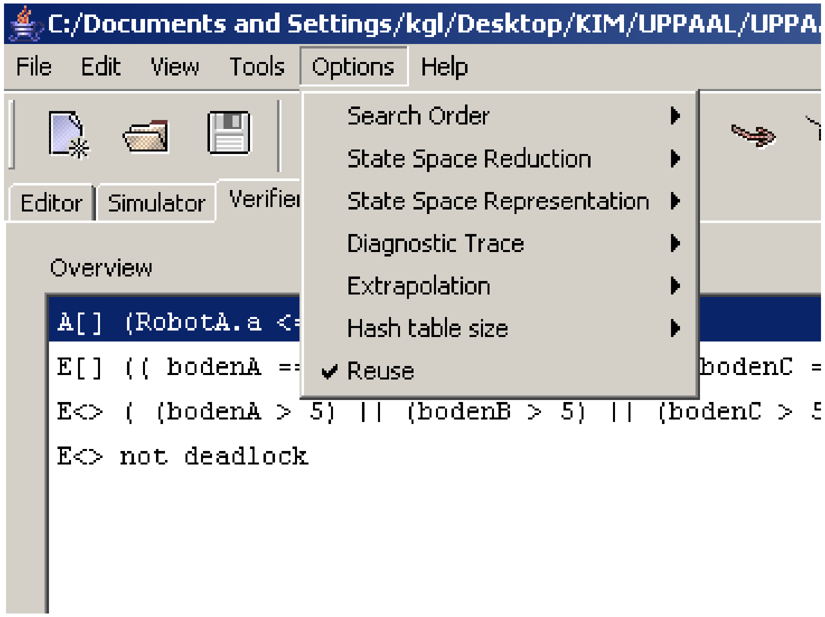
Remove all
bounds
involving y
and set y to 0



UPPAAL

Verification Options

Verification Options



```
A[] (RobotA.a <=
E[] (( bodenA ==                          bodenC =
E<> ( (bodenA > 5) || (bodenB > 5) || (bodenC > 5
E<> not deadlock
```

Search Order
- Depth First
- Breadth First

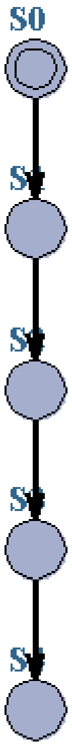
State Space Reduction
- None
- Conservative
- Aggressive

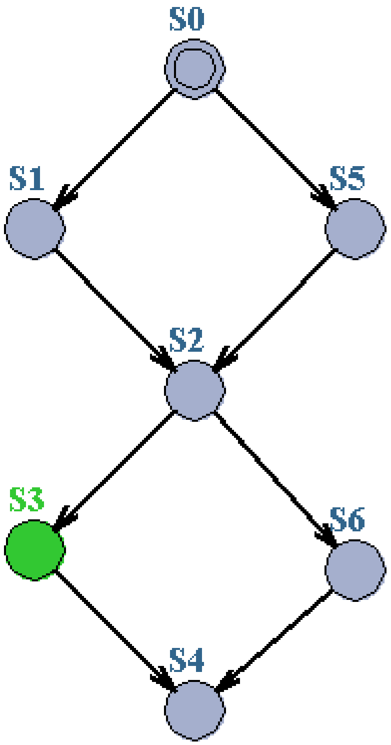
State Space Representation
- DBM
- Compact Form
- Under Approximation
- Over Approximation

Diagnostic Trace
- Some
- Shortest
- Fastest

Extrapolation
Hash Table size
Reuse

State Space Reduction

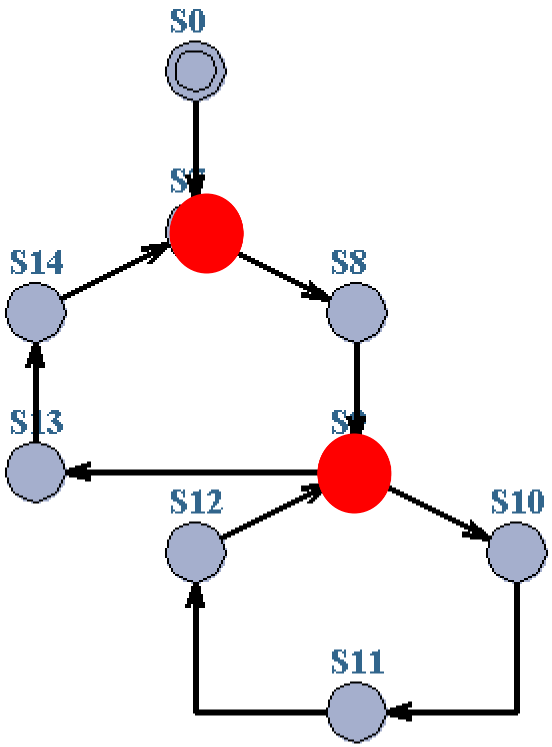




However,
Passed list useful for efficiency

No Cycles: Passed list not needed for *termination*

State Space Reduction



Cycles:

Only symbolic states involving loop-entry points need to be saved on Passed list

To Store or Not To Store

Behrmann, Larsen, Pelanek 2003

117 $\text{states}_{\text{total}}$

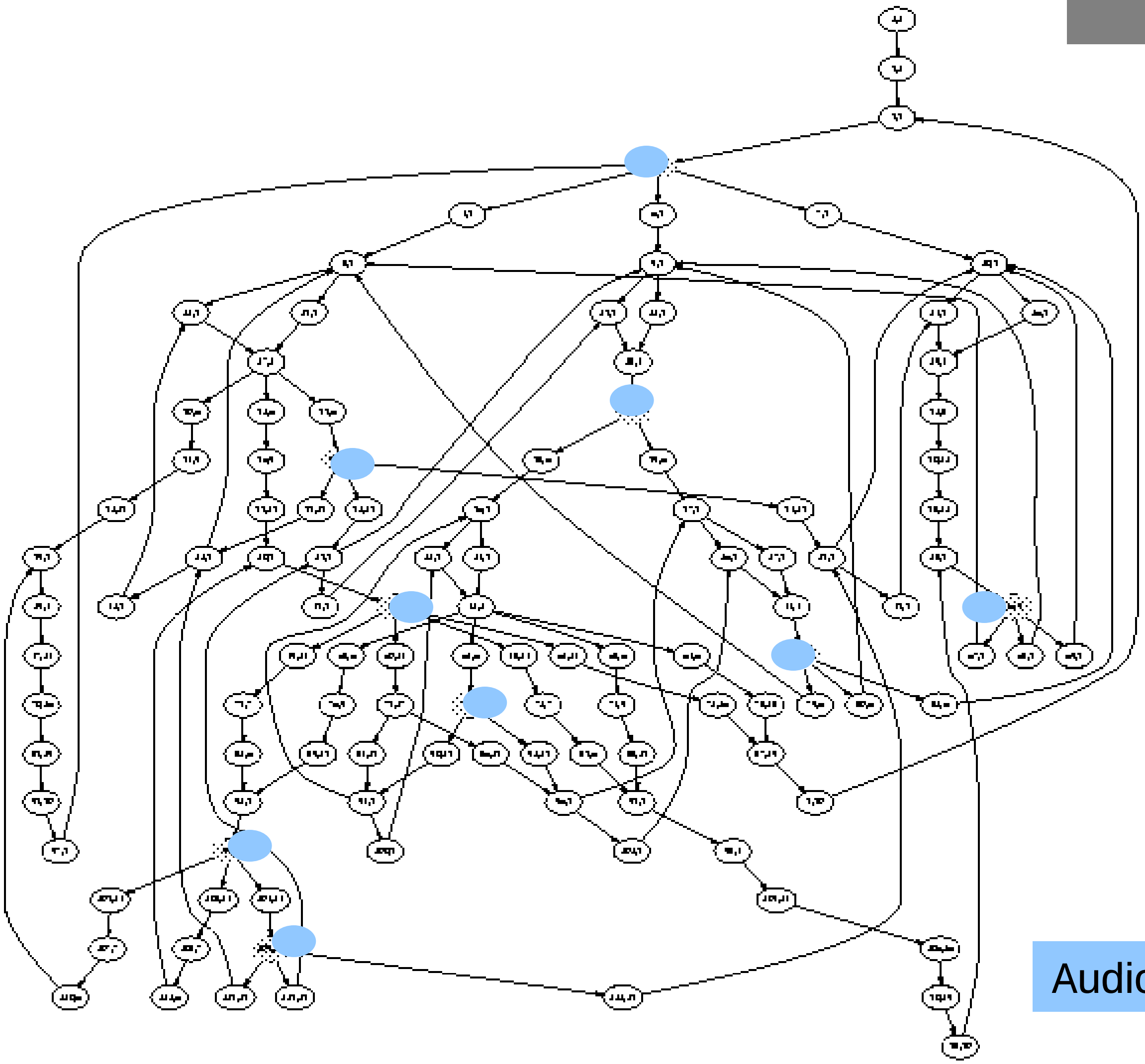
→

81 $\text{states}_{\text{entrypoint}}$

→

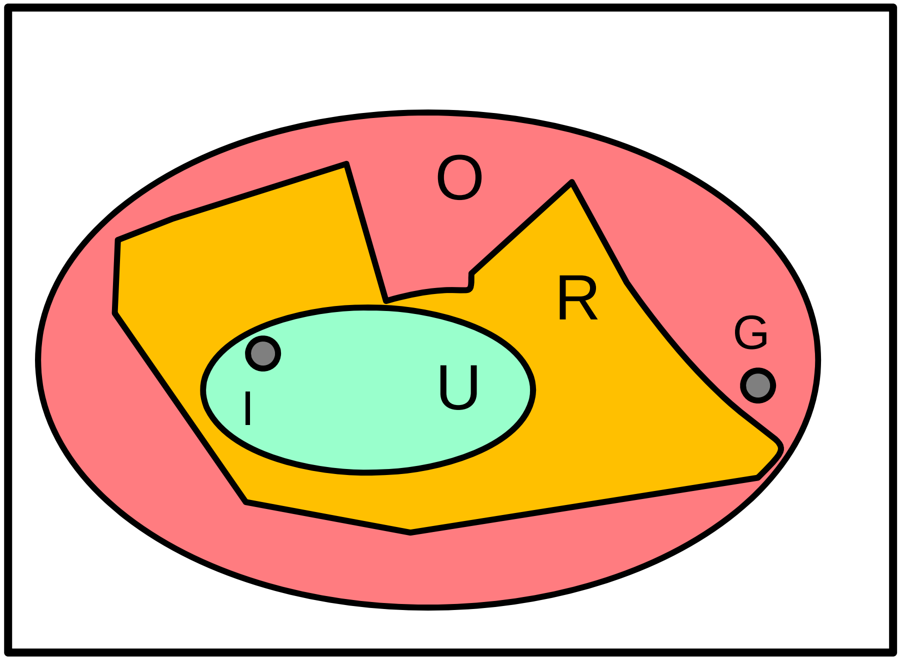
9 states

Time OH less than 10%



Audio Protocol

Over/Under Approximation



Declared State Space

Question:

$G \in R$?

How to use:

$G \in O$?

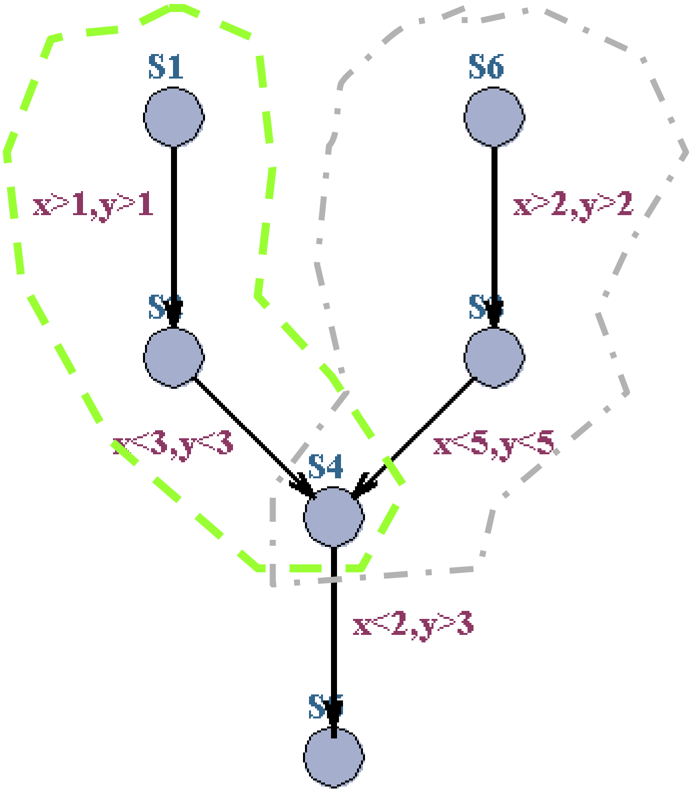
$G \in U$?

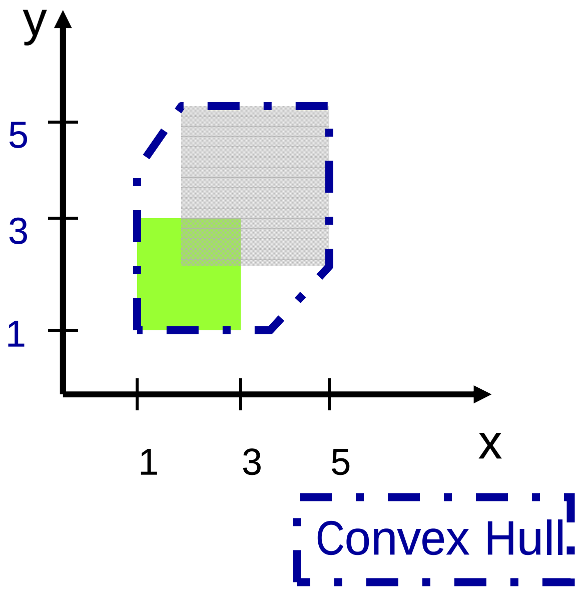
$G \in U \Rightarrow G \in R$

$\neg(G \in O) \Rightarrow \neg(G \in R)$

Over-approximation

Convex Hull

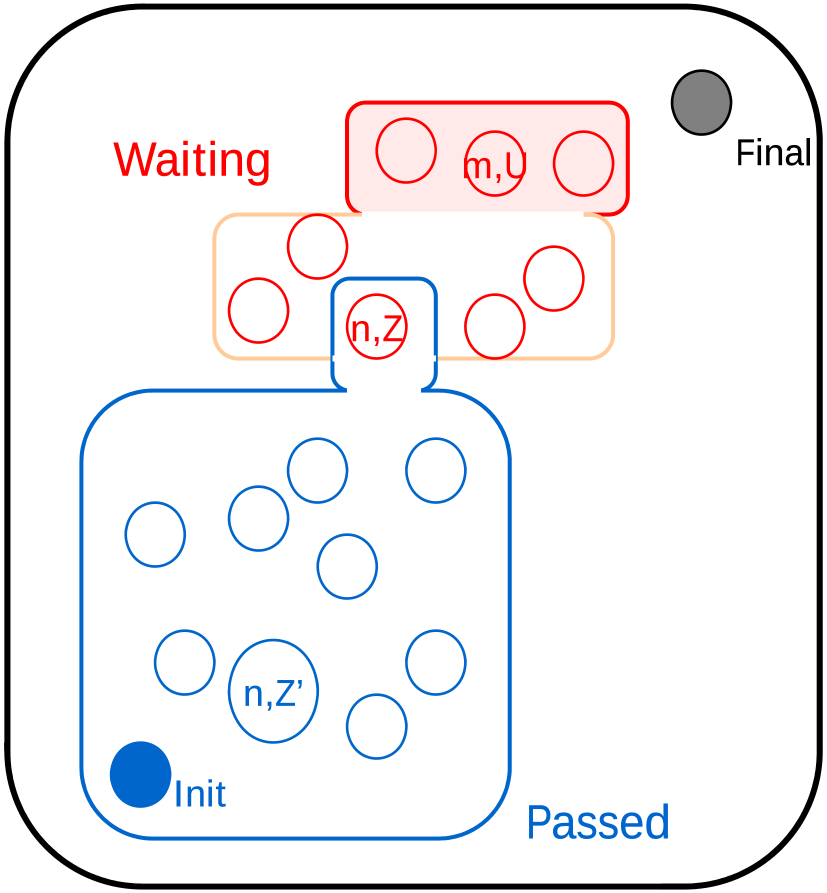




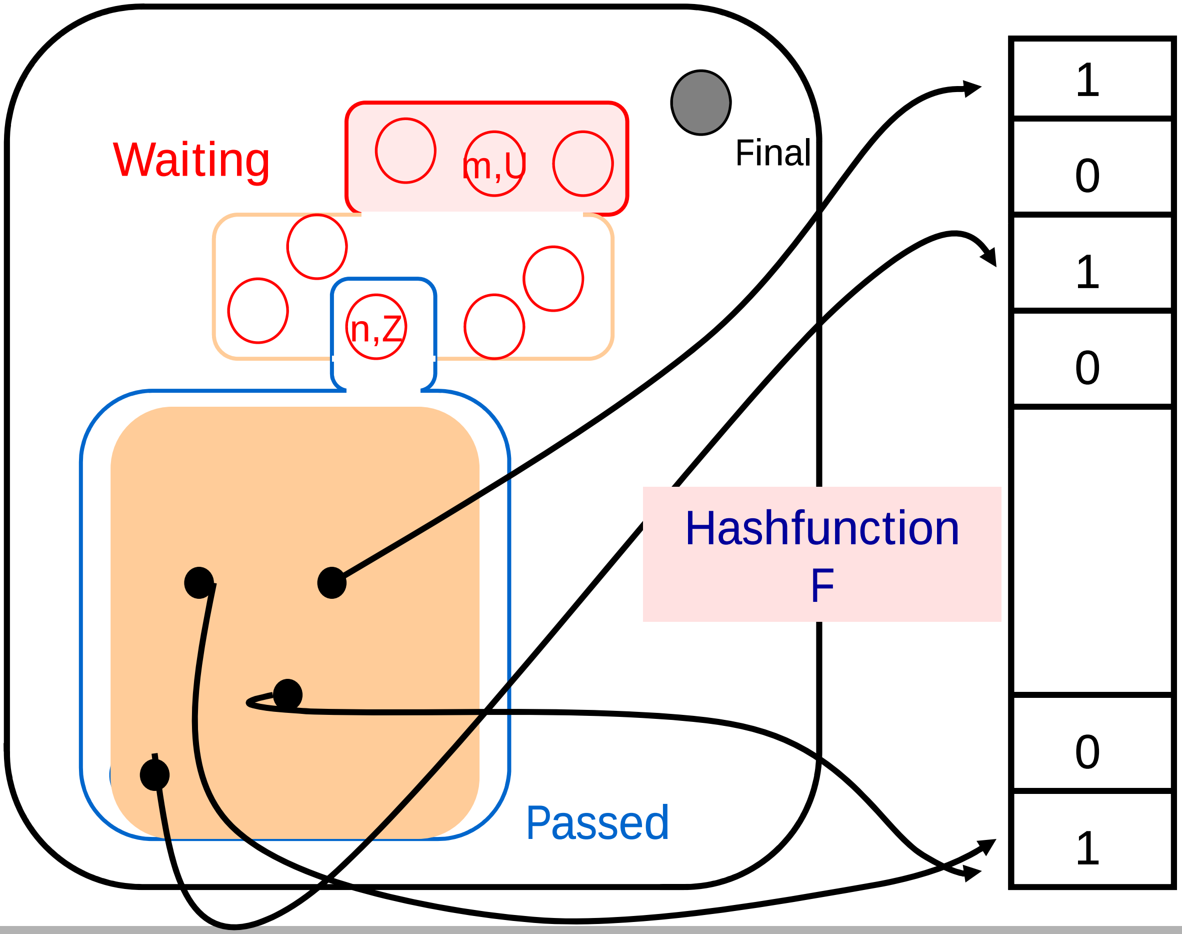
TACAS04: An EXACT method performing as well as Convex Hull has been developed based on abstractions taking max constants into account distinguishing between clocks, locations and ≤ & ≥

Under-approximation

Bitstate Hashing



Under-approximation
Bitstate Hashing



Passed= Bitarray

UPPAAL
4 - 512 Mbits

LAB-Exercises

http://www.cs.aau.dk/~kgl/QMC2010/exercises/

Exercise 19

Exercise 2

Exercise 1