

Adding Symmetry Reduction to UPPAAL

M. Hendriks¹ G. Behrmann² K.G. Larsen²
P. Niebert³ F. Vaandrager¹

¹University of Nijmegen, The Netherlands

²Aalborg University, Denmark

³Université de Provence, France

Introduction

Motivation

- Exploitation of full symmetry can give factorial gain
- Full symmetry occurs in many timed systems
 - ▷ Fischer's mutex protocol, CSMA/CD protocol (UPPAAL benchmarks)
 - ▷ Dynamic configuration IPv4 addresses (Zhang & Vaandrager)
 - ▷ Distributed agreement algorithm (Attiya, Dwork, Lynch & Stockmeyer)

Introduction

Motivation

- Exploitation of full symmetry can give factorial gain
- Full symmetry occurs in many timed systems
 - ▷ Fischer's mutex protocol, CSMA/CD protocol (UPPAAL benchmarks)
 - ▷ Dynamic configuration IPv4 addresses (Zhang & Vaandrager)
 - ▷ Distributed agreement algorithm (Attiya, Dwork, Lynch & Stockmeyer)

Approach

- Ip & Dill: *Better Verification Through Symmetry* (1993)
 - ▷ Scalarsets as fully symmetric data type in description language
- Successfully used in several model checkers
 - ▷ MUR φ , SPIN, SMV

Outline

(1) Some theory (Ip & Dill, 1993)

(2) Implementation

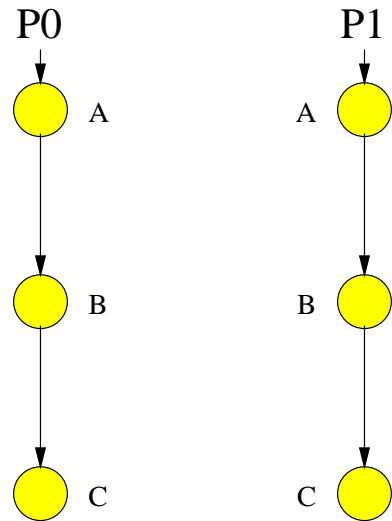
- UPPAAL language enhancement
- Representative computation

(3) Results

(4) Conclusions

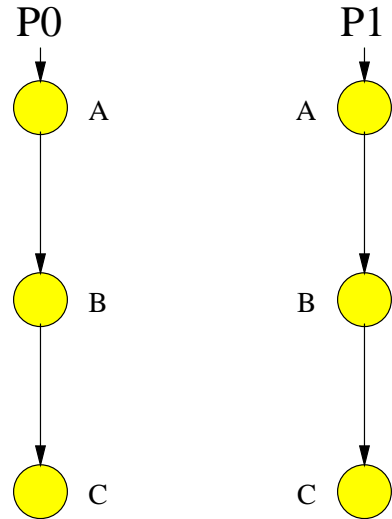
Theory (Ip & Dill, 1993)

Syntactical level: system
description

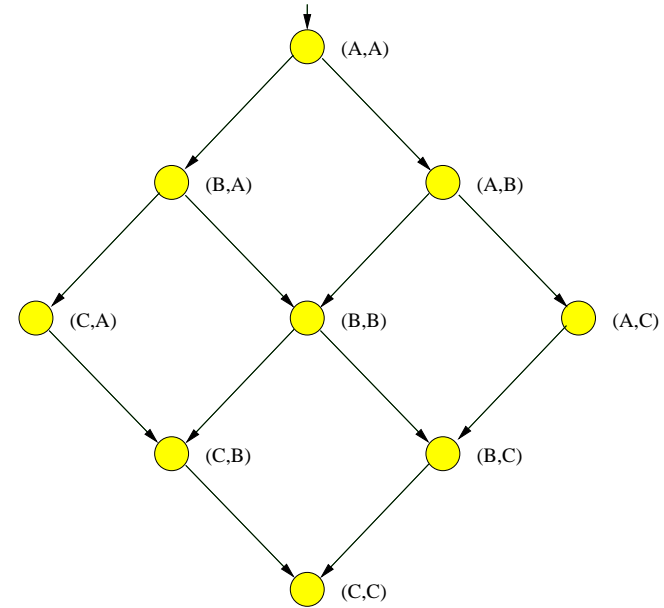


Theory (Ip & Dill, 1993)

Syntactical level: system description

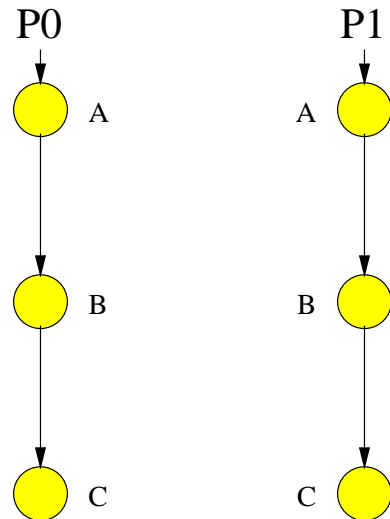


Semantical level: state graph
(Q, Q_0, Δ)

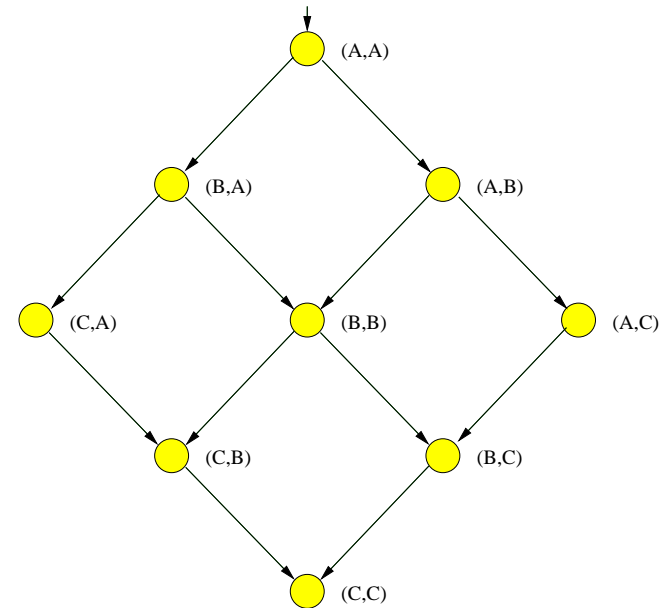


Theory (Ip & Dill, 1993)

Syntactical level: system description



Semantical level: state graph
(Q, Q_0, Δ)

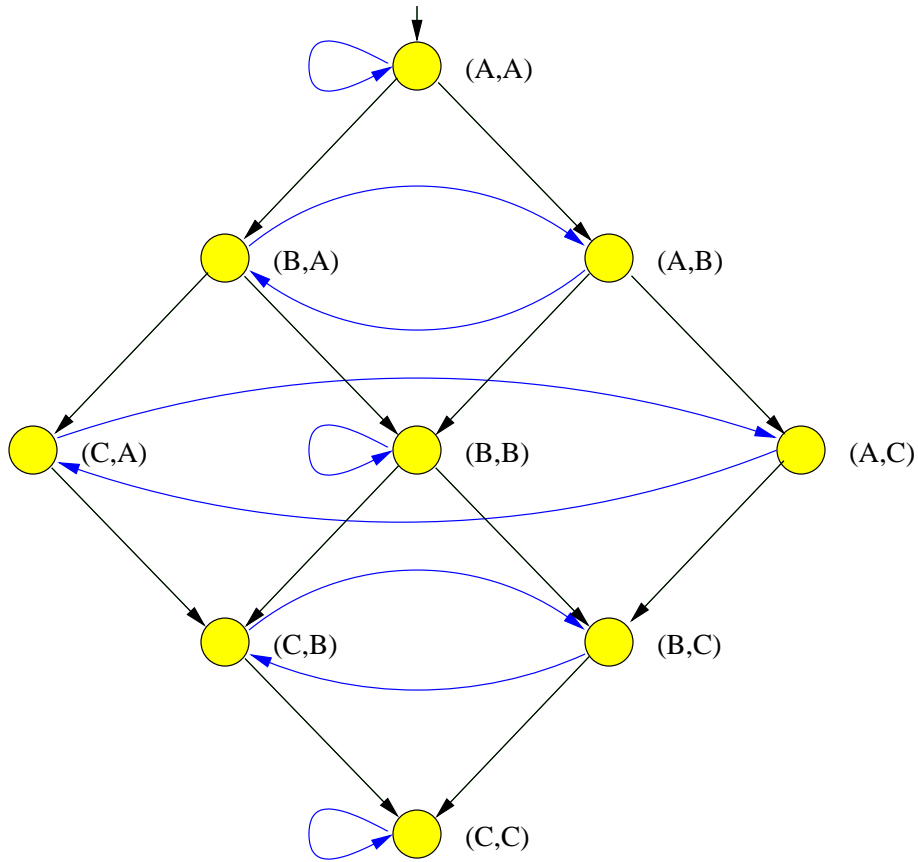


Detect bijections $h : Q \rightarrow Q$ in state graph from system description such that

- ▷ $q \in Q_0 \iff h(q) \in Q_0$
- ▷ $(q_1, q_2) \in \Delta \iff (h(q_1), h(q_2)) \in \Delta$

Theory (2)

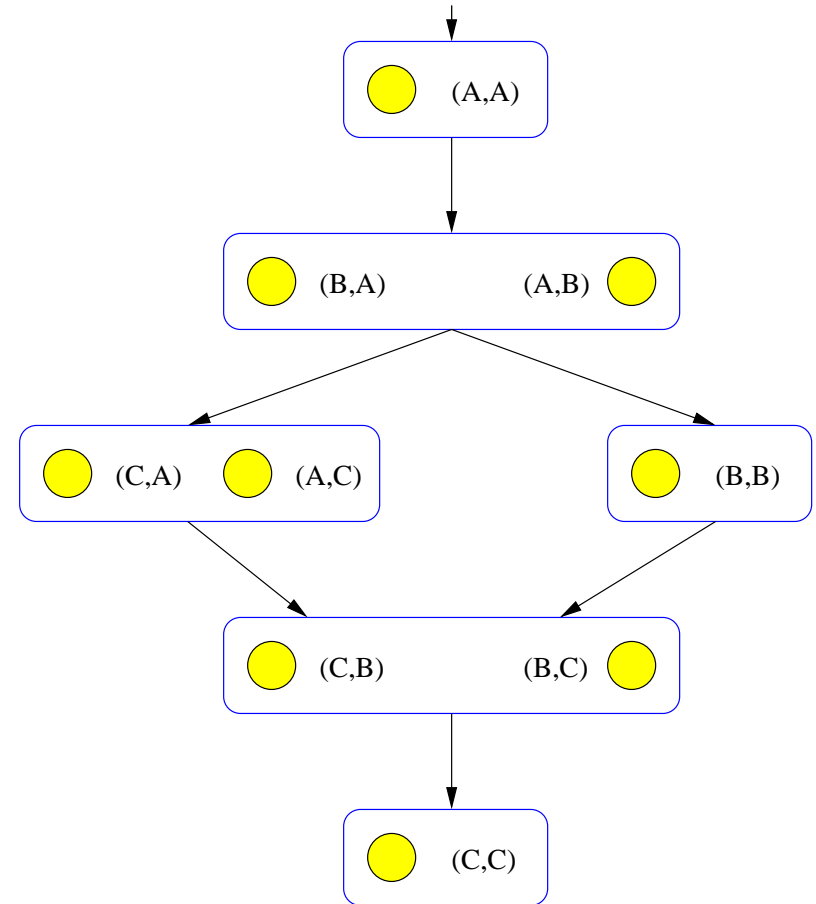
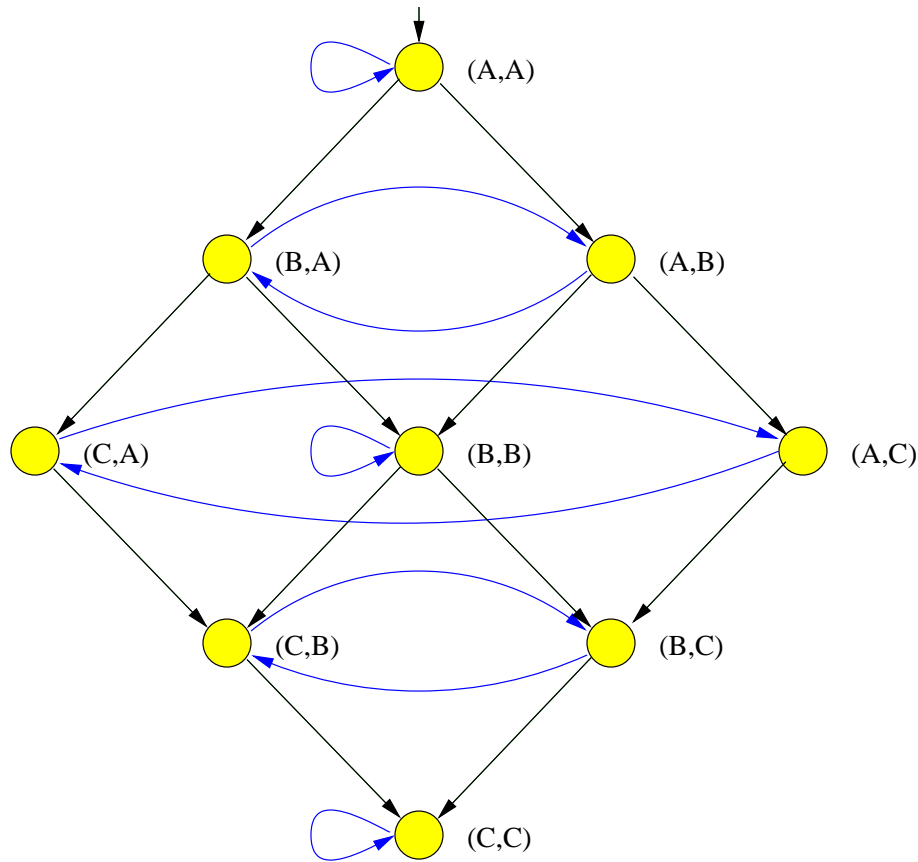
Automorphism h on state graph G



Theory (2)

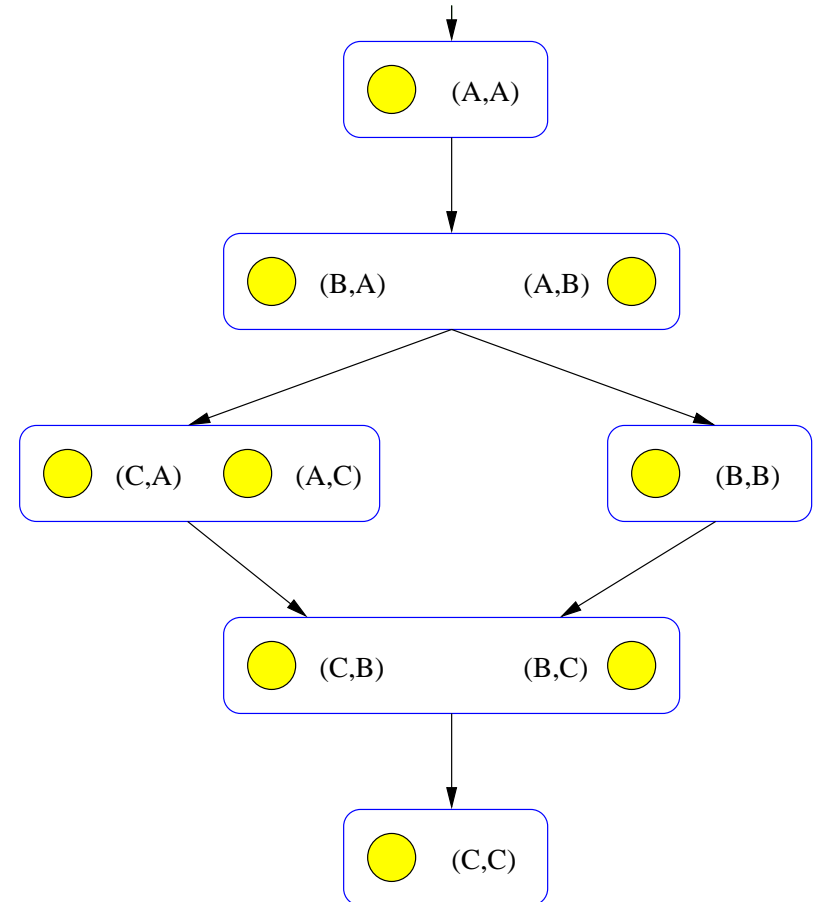
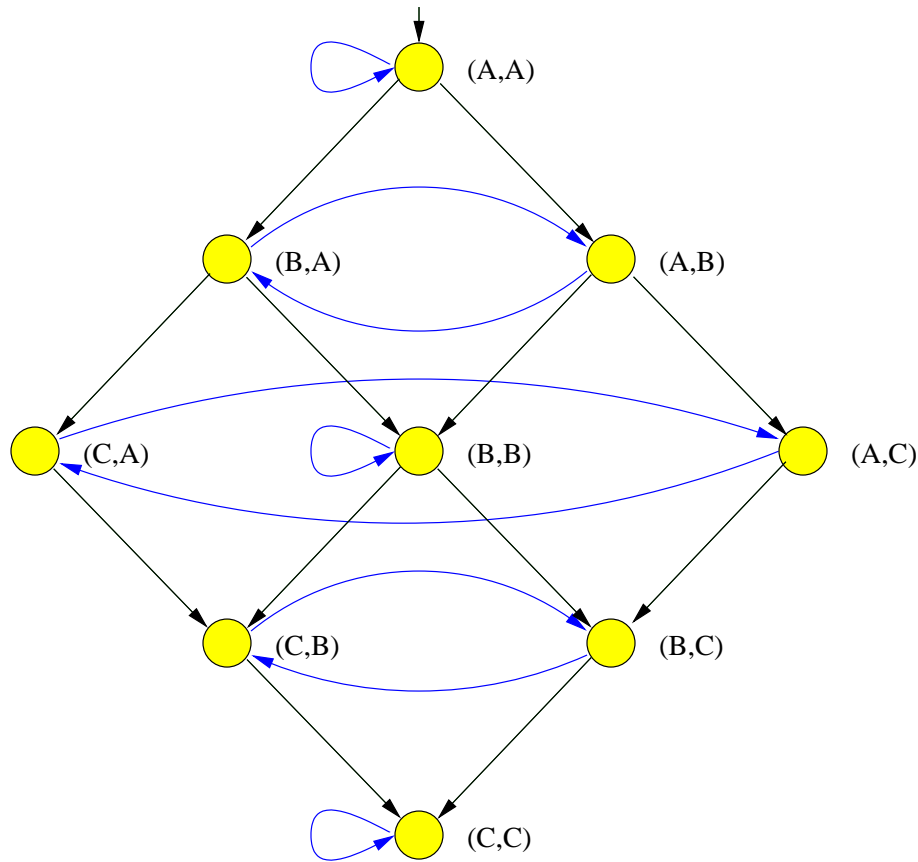
Automorphism h on state graph G

h induces quotient graph G'



Theory (2)

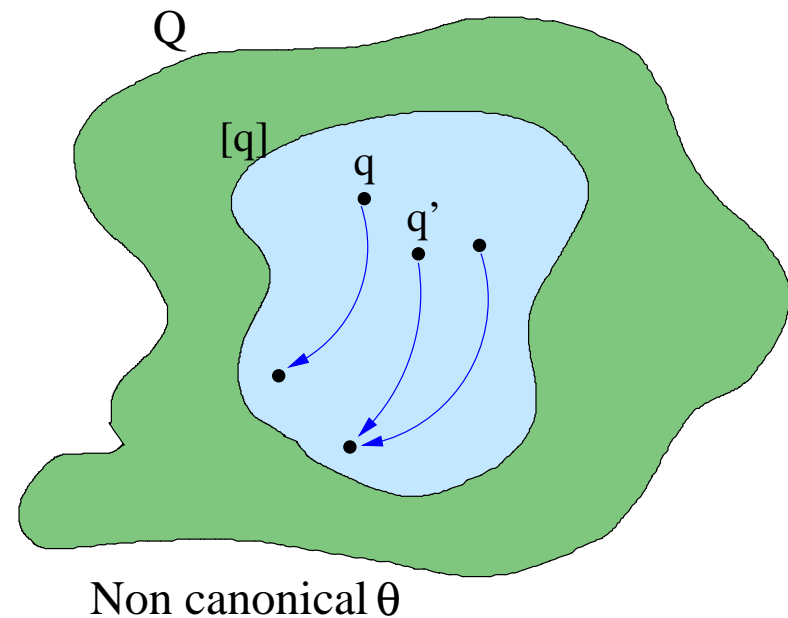
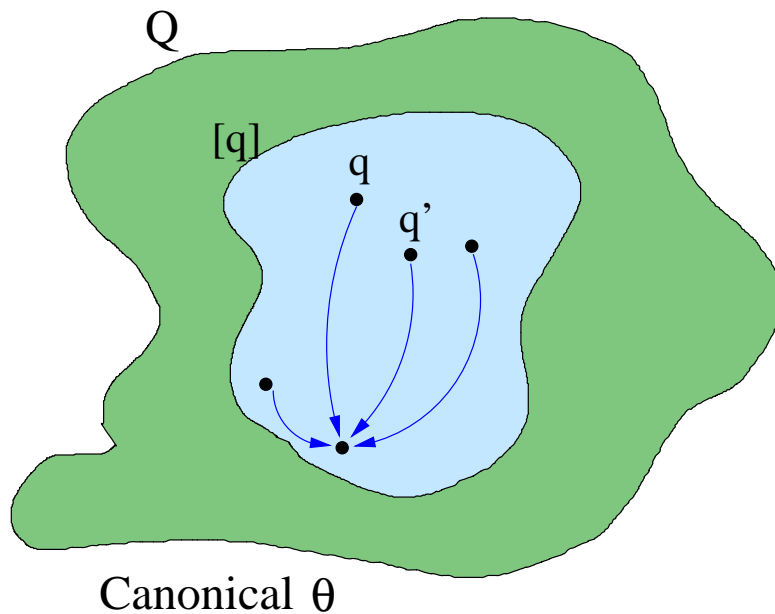
Automorphism h on state graph G h induces quotient graph G'



Then: q reachable in $G \iff [q]$ reachable in G'

Implementation

- (1) Find a set of automorphisms H from the system description
 - Introduce a symmetric data type, e.g., *scalarsets*
- (2) During state space exploration: $[q] \stackrel{?}{=} [q']$ (orbit problem)
 - Use a *representative* function $\theta : Q \rightarrow Q$



Language enhancements

Template header:

```
process F (const proc_id pid)
```

Local declarations:

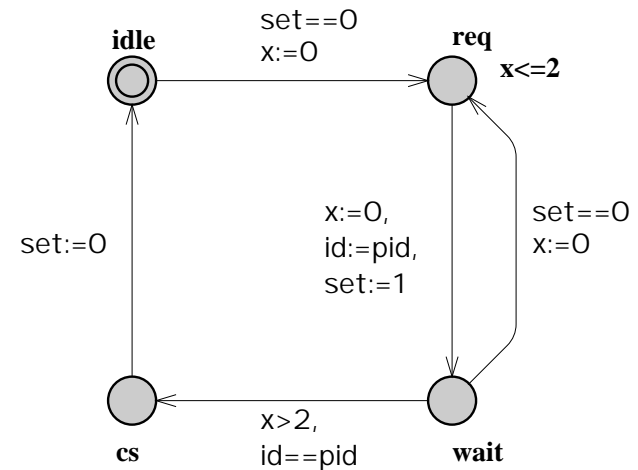
```
clock x;
```

Global declarations:

```
typedef scalarset[3] proc_id;
proc_id id;
bool set;
```

Process assignments:

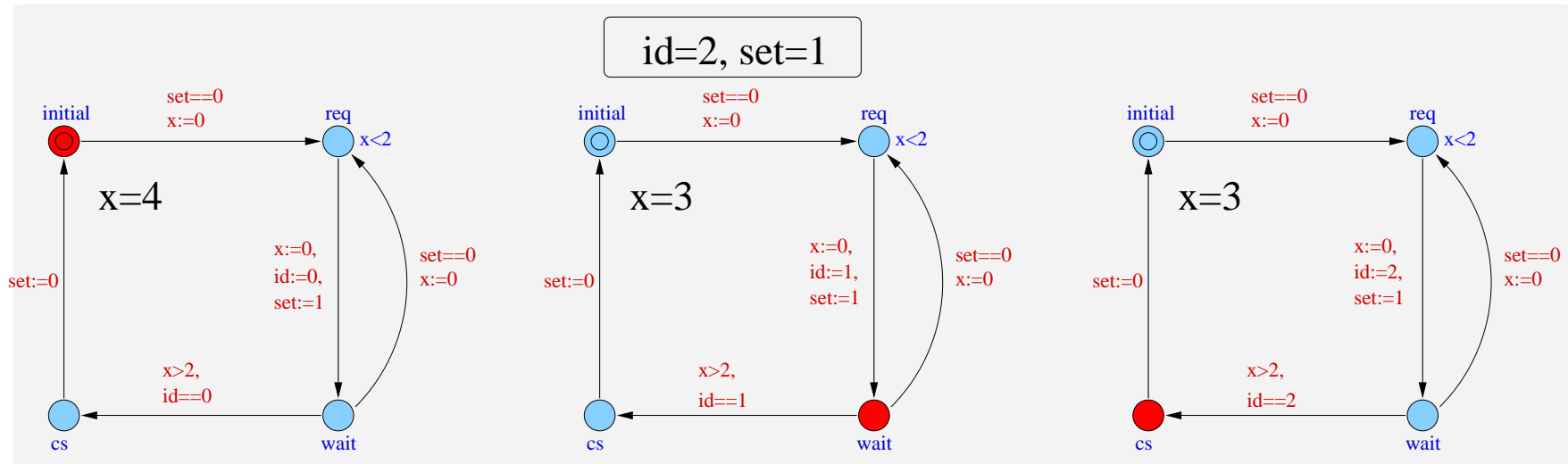
```
Procs = forall i in proc_id : F(i);
```



System description:

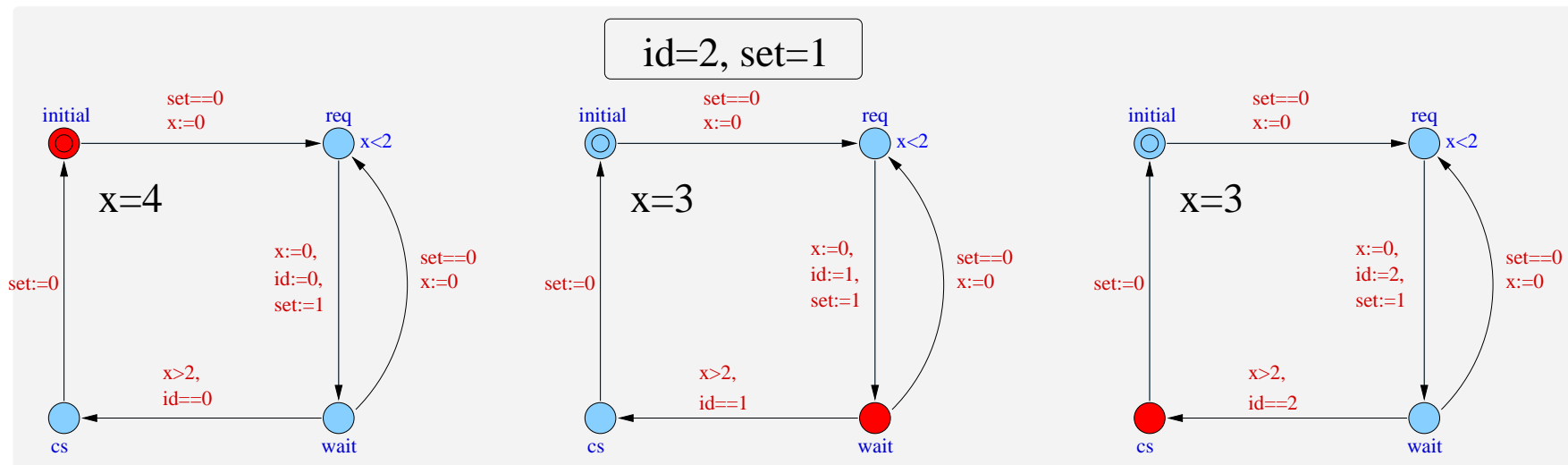
```
system Procs;
```

State swap example

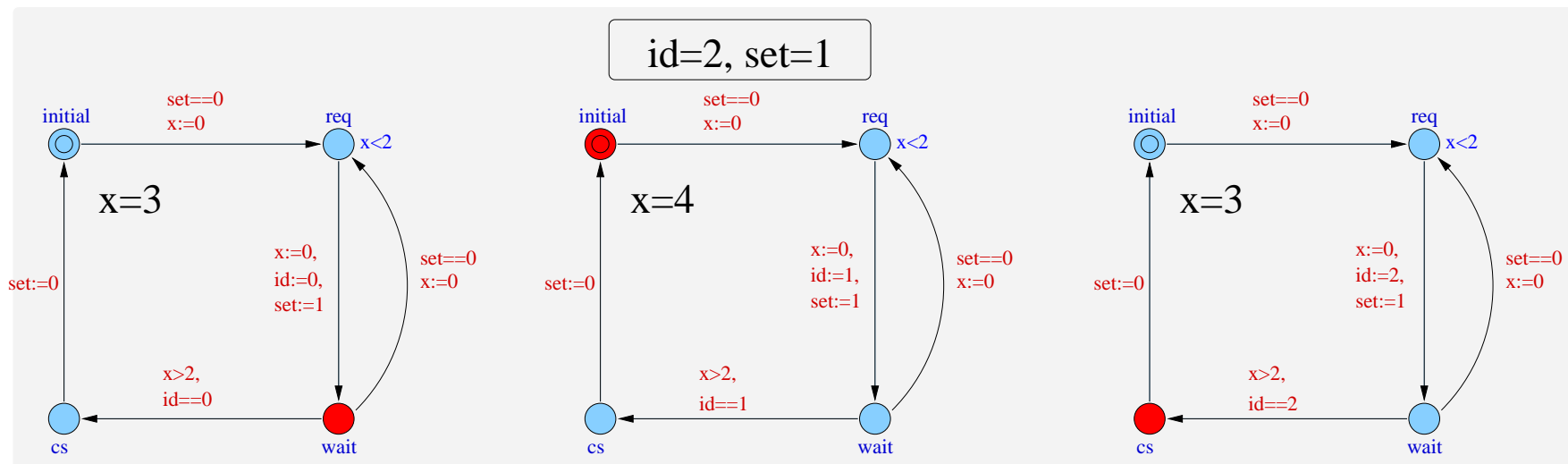


Swap process 0 with process 1

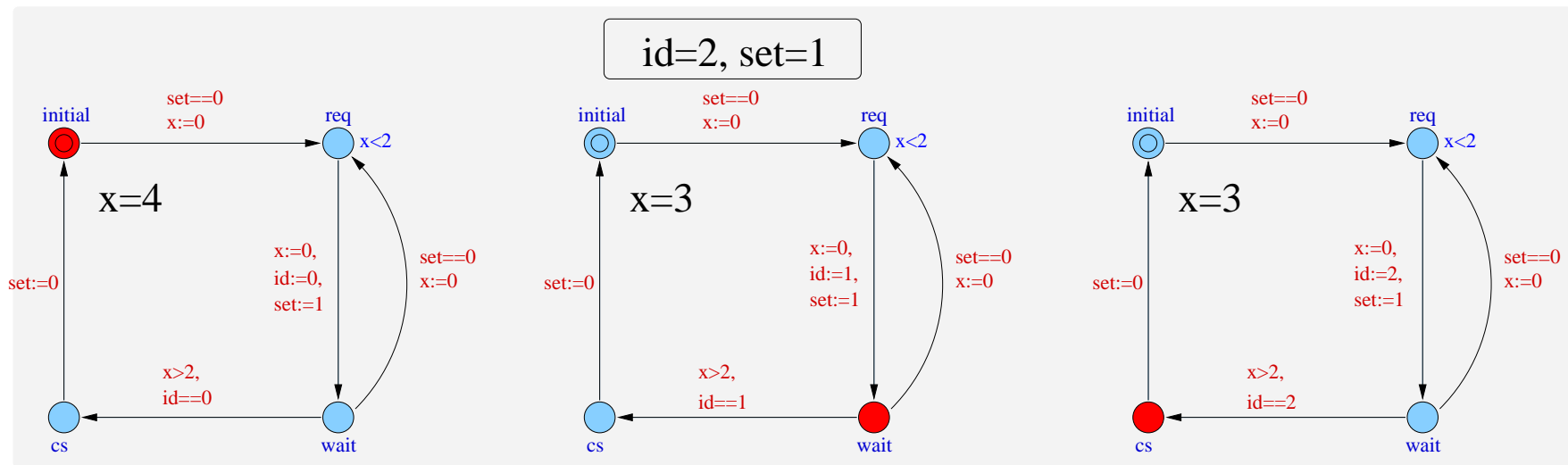
State swap example



Swap process 0 with process 1

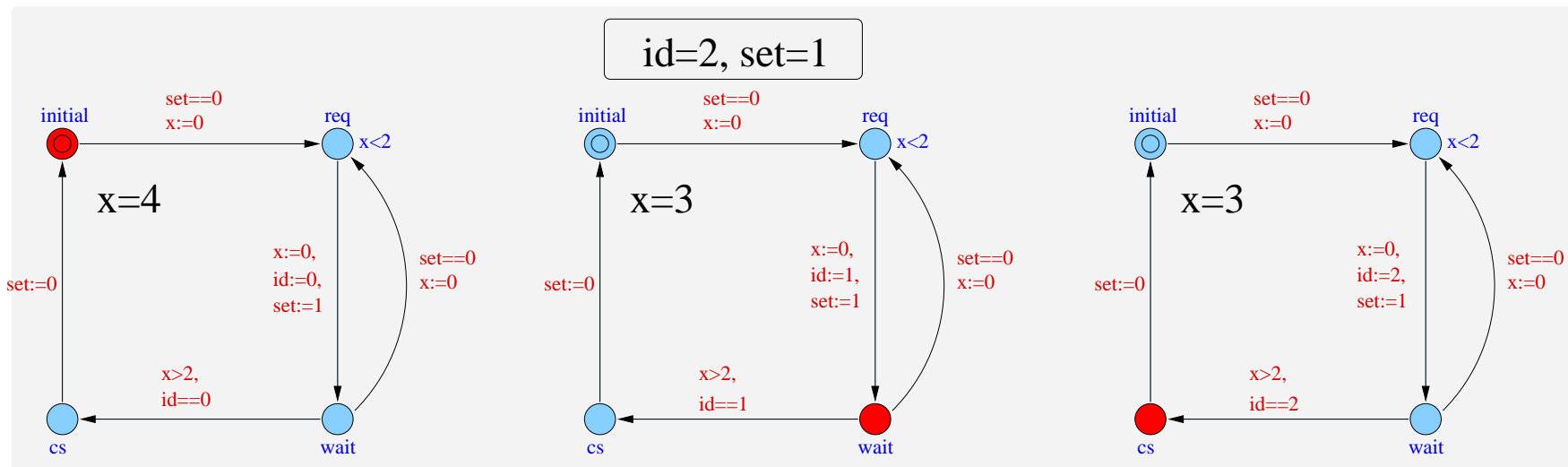


State swap example (2)

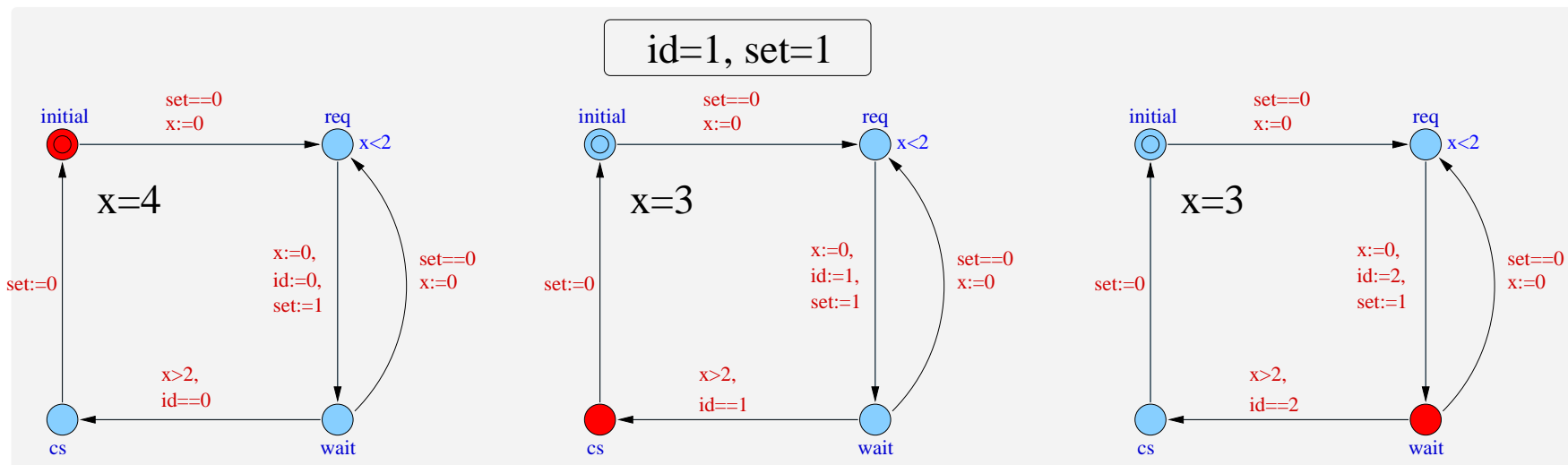


Swap process 1 with process 2

State swap example (2)



Swap process 1 with process 2



Representative computation

Idea: “minimize” state using state swaps w.r.t. some total order

Problem: symbolic representation of sets of clock valuations (zones)

Solution: *diagonal property* of zones

Diagonal property

Let x and y be clocks and let Z be a zone (set of clock valuations)

$$x \preceq_Z y \iff \forall \nu \in Z \nu(x) \leq \nu(y)$$

$$x \approx_Z y \iff \forall \nu \in Z \nu(x) = \nu(y)$$

$$x \prec_Z y \iff (x \preceq_Z y \wedge x \not\approx_Z y)$$

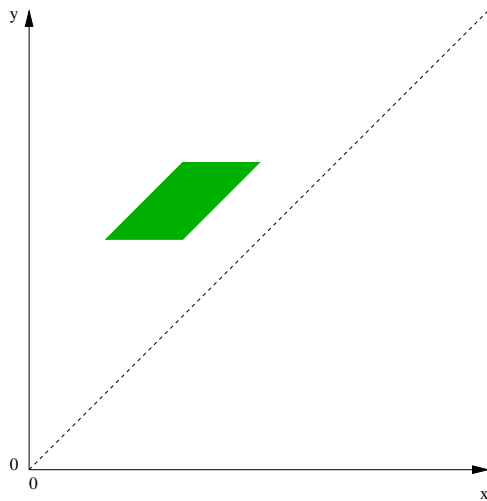
Lemma (diagonal property): Consider a symbolic forward state space exploration algorithm. Assume that the clocks are reset to the value 0 only. For all states (\vec{l}, ν, Z) stored in the waiting and passed list and for all clocks x and y holds that either $x \prec_Z y$, $y \prec_Z x$, or $x \approx_Z y$.

Diagonal property: proof sketch

(1) Initial zone satisfies diagonal property (all clocks equal 0)

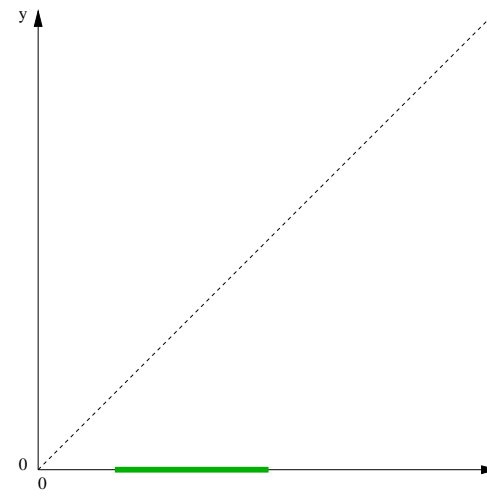
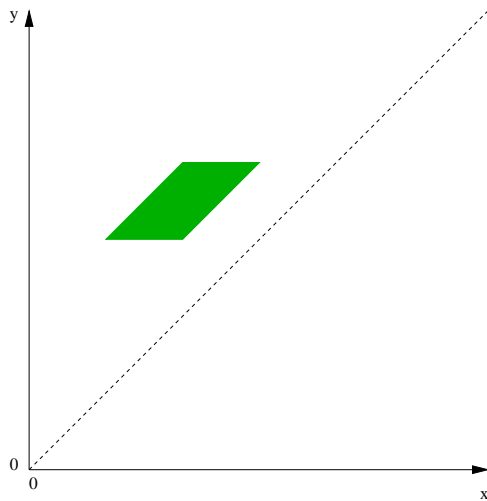
Diagonal property: proof sketch

- (1) Initial zone satisfies diagonal property (all clocks equal 0)
- (2) Clock reset



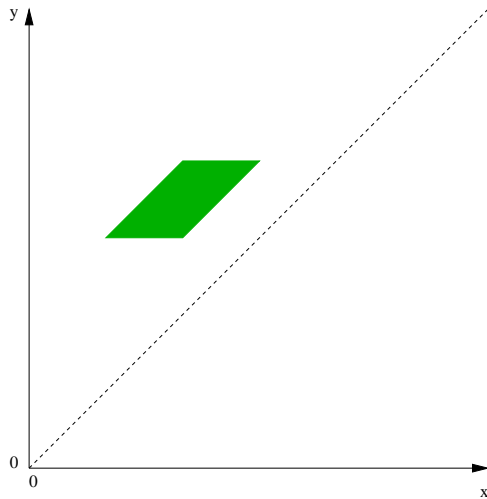
Diagonal property: proof sketch

- (1) Initial zone satisfies diagonal property (all clocks equal 0)
- (2) Clock reset



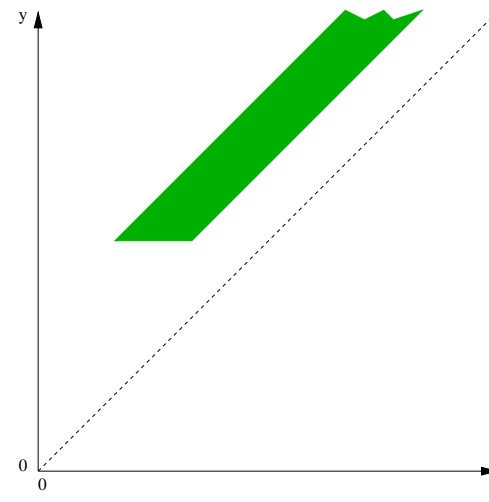
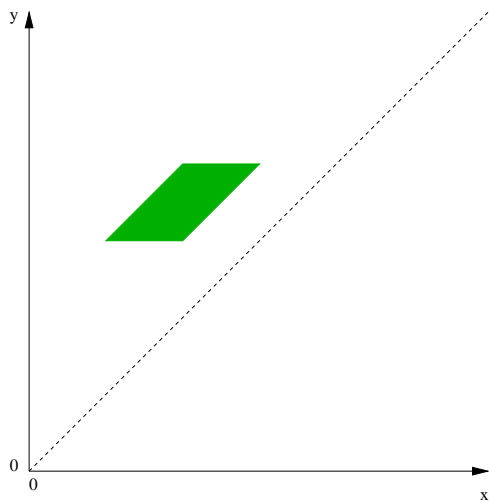
Diagonal property: proof sketch

- (1) Initial zone satisfies diagonal property (all clocks equal 0)
- (2) Clock reset
- (3) Time elapse



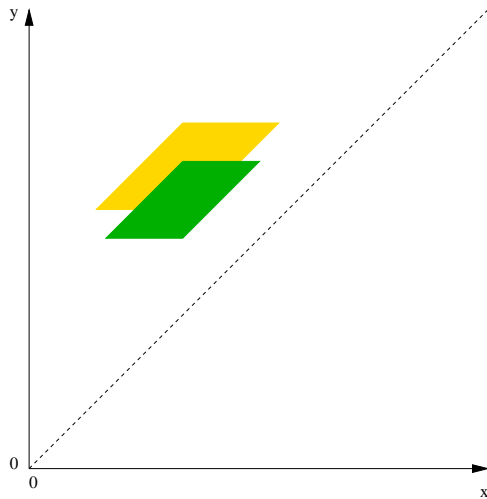
Diagonal property: proof sketch

- (1) Initial zone satisfies diagonal property (all clocks equal 0)
- (2) Clock reset
- (3) Time elapse



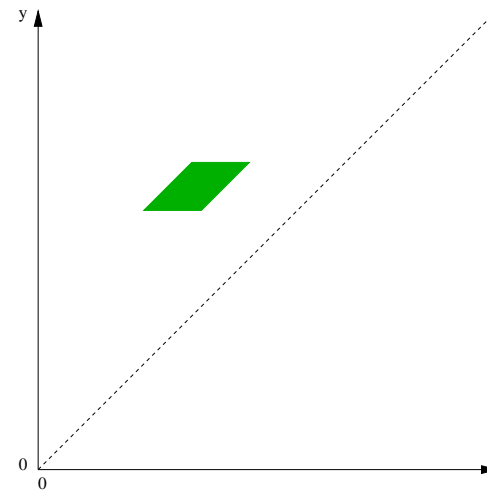
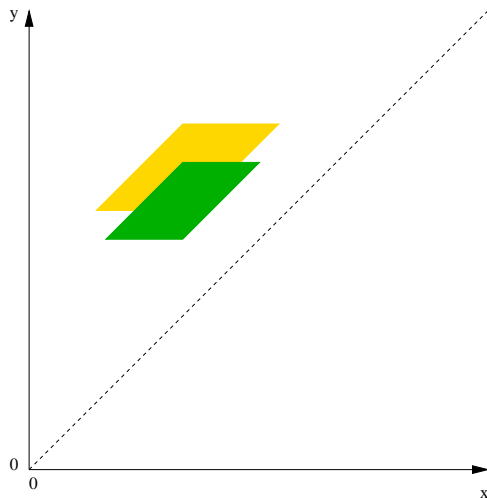
Diagonal property: proof sketch

- (1) Initial zone satisfies diagonal property (all clocks equal 0)
- (2) Clock reset
- (3) Time elapse
- (4) Intersection



Diagonal property: proof sketch

- (1) Initial zone satisfies diagonal property (all clocks equal 0)
- (2) Clock reset
- (3) Time elapse
- (4) Intersection



Representative computation (2)

Diagonal property gives a total order on clocks (and on states)

- Easily decidable using the DBM representation of zones

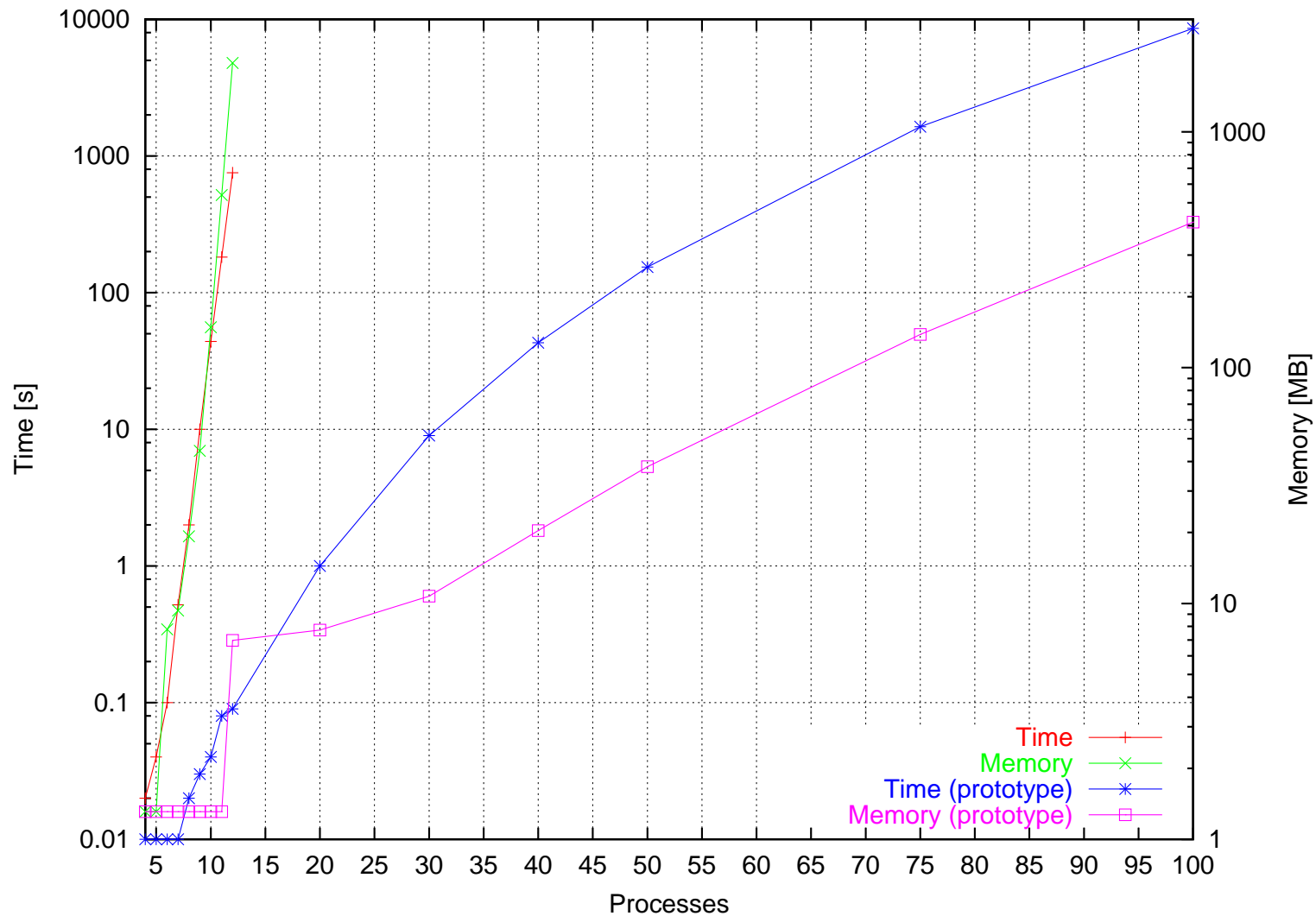
State swaps implement transpositions of scalarset elements

- All permutations of scalarset elements can be obtained

Representative computation by minimization of state

- “Bubble sort” the state with state swaps w.r.t. the total order
- *Canonical* under certain assumptions that involve the *discrete* part of the state

Results



Conclusions