Reachability for Dynamic Parametric Processes

Helmut Seidl, Anca Muscholl, Igor Walukiewicz

TU München, U. Bordeaux

Dagstuhl, Nov. 25, 2016

Reachability for Dynamic Parametric Processes w/o Weak Memory

Helmut Seidl, Anca Muscholl, Igor Walukiewicz

TU München, U. Bordeaux

Dagstuhl, Nov. 25, 2016

Dynamic Process

```
root()  {
        spawn(p);
                switch (x) {
                  case 2 :   result = #;
                }
}
```

Dynamic Process

```
root()  {
        spawn(p);
                switch (x) {
                  case 2 :   result = #;
                }
}
```

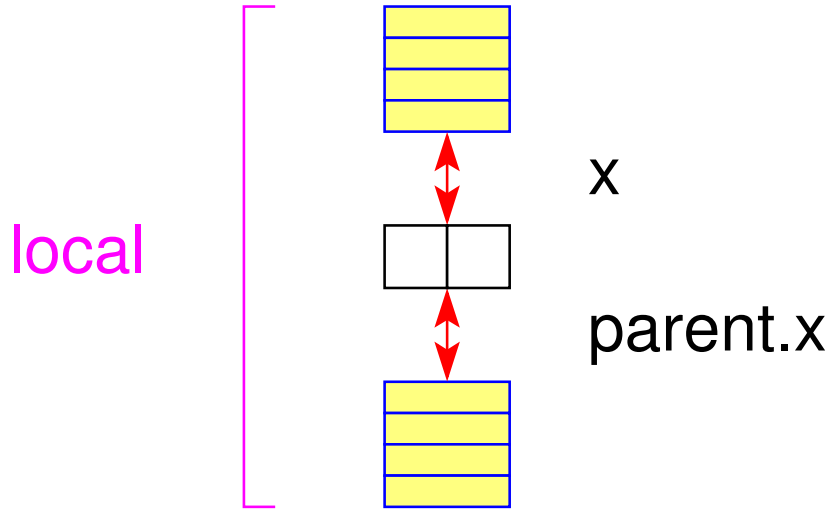
- thread creation

Dynamic Process

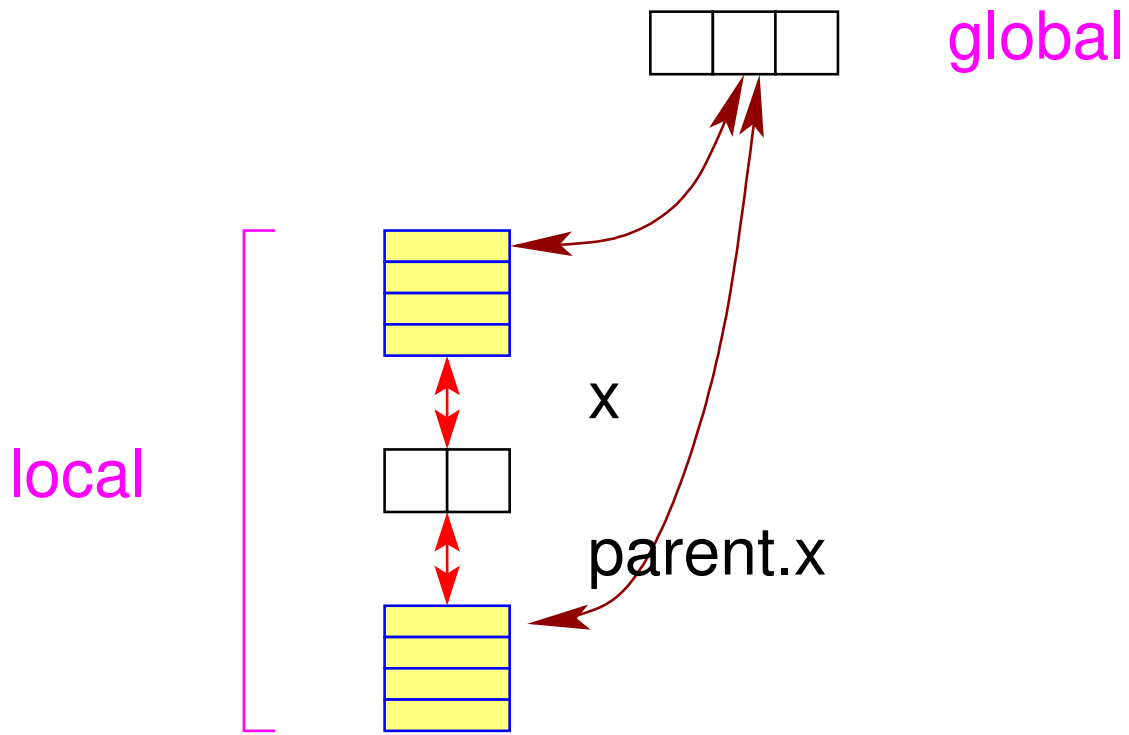
```
root()  {
        spawn(p);
                switch (x) {
                  case 2 :   result = #;
                }
}
```

- thread creation
- access to data

Data Access



Data Access



Dynamic Process (cont.)

```
p()  {
     switch (parent.x) {
      case 0 :  spawn(p);
                if (*)  parent.x = 1
                else switch (x) {
                       case 1 :  parent.x = 1; break;
                       case 2 :  break;
                     }
                break;
      case 1 :  spawn(p);
                if (*)  parent.x = 0
                else switch (x) {
                       case 1 :  parent.x = 2; break;
                       case 2 :  parent.x = 2; break;
                     }
     }
}
```

Bad News

Reachability for finitely many finite processes is difficult.

Bad News

Reachability for finitely many finite processes is difficult.

Reachability for finitely many recursive processes is undecidable in presence of

Bad News

Reachability for finitely many finite processes is difficult.

Reachability for finitely many recursive processes is undecidable in presence of

- 2 bit globals — Folklore
- rendezvous — Ramalingam 2000

Bad News

Reachability for finitely many finite processes is difficult.

Reachability for finitely many recursive processes is undecidable in presence of

- 2 bit globals — Folklore
- rendezvous — Ramalingam 2000
- arbitrary locking — Kahlon et al. 2005

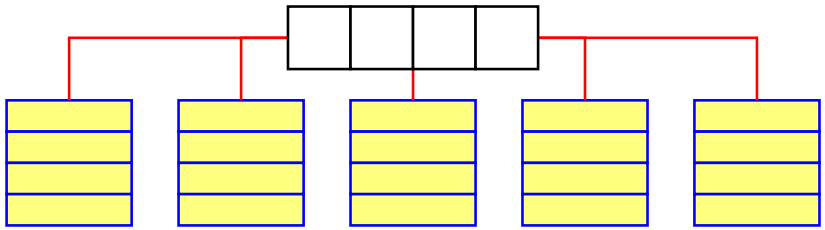
Good News

Restricting locking disciplines helps! Kahlon et al. 2005

Good News

Restricting locking disciplines helps! Kahlon et al. 2005

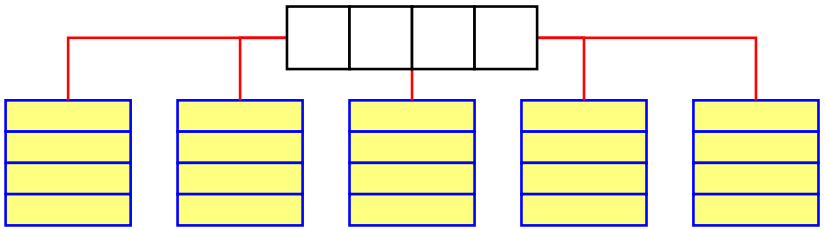
Parametrization helps! Kahlon 2008

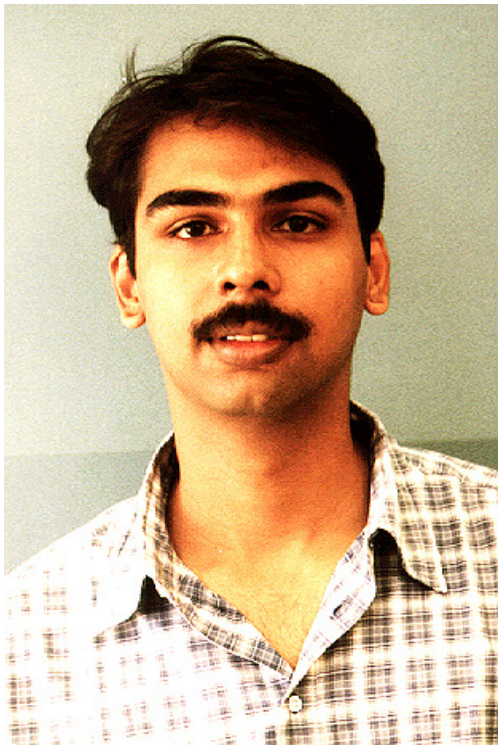


Good News

Restricting locking disciplines helps! Kahlon et al. 2005

Parametrization helps! Kahlon 2008

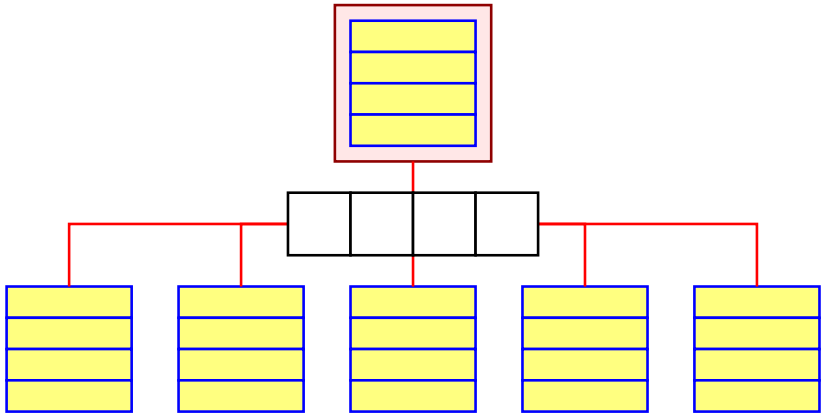




More Good News

Parametric contributors + one master

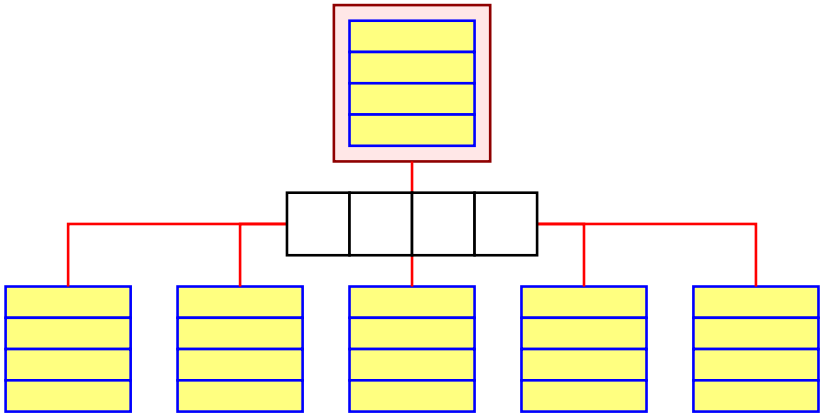
Hague 2011
Esparza et al. 2015



More Good News

Parametric contributors + one master

Hague 2011
Esparza et al. 2015



Dynamic Processes

no data, no locking

Bouajjani et al. 2005

Dynamic Processes

no data, no locking — Bouajjani et al. 2005

no data, restricted locking — Müller-Olm and friends

Dynamic Processes

no data, no locking

Bouajjani et al. 2005

no data, restricted locking

Müller-Olm and friends

Our Model of Processes

- recursion
- dynamic parametric creation of child processes
- asynchronous execution (no test-and-set)

Our Model of Processes

- recursion
- dynamic parametric creation of child processes
- asynchronous execution (no test-and-set)
- Atomic global variables

Our Model of Processes

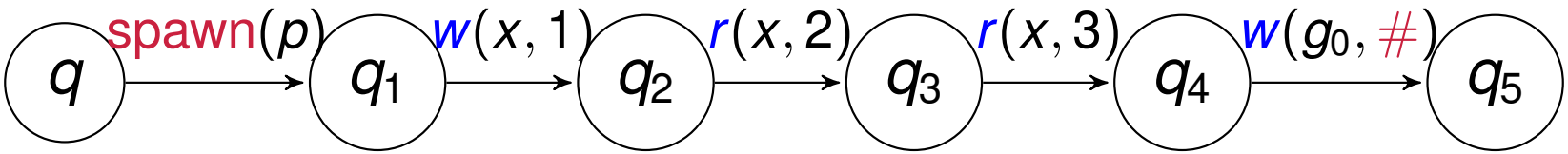
- recursion
- dynamic parametric creation of child processes
- asynchronous execution (no test-and-set)
- Atomic global variables
- Atomic local variables

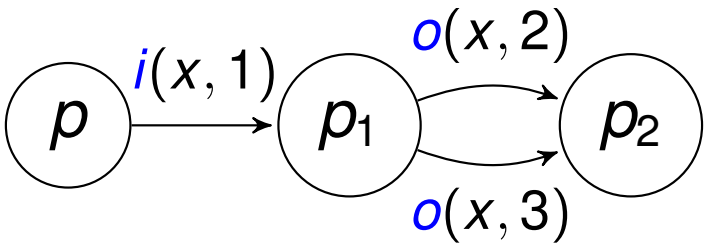
Our Model of Processes

- recursion
- dynamic parametric creation of child processes
- asynchronous execution (no test-and-set)
- Atomic global variables
- Atomic local variables

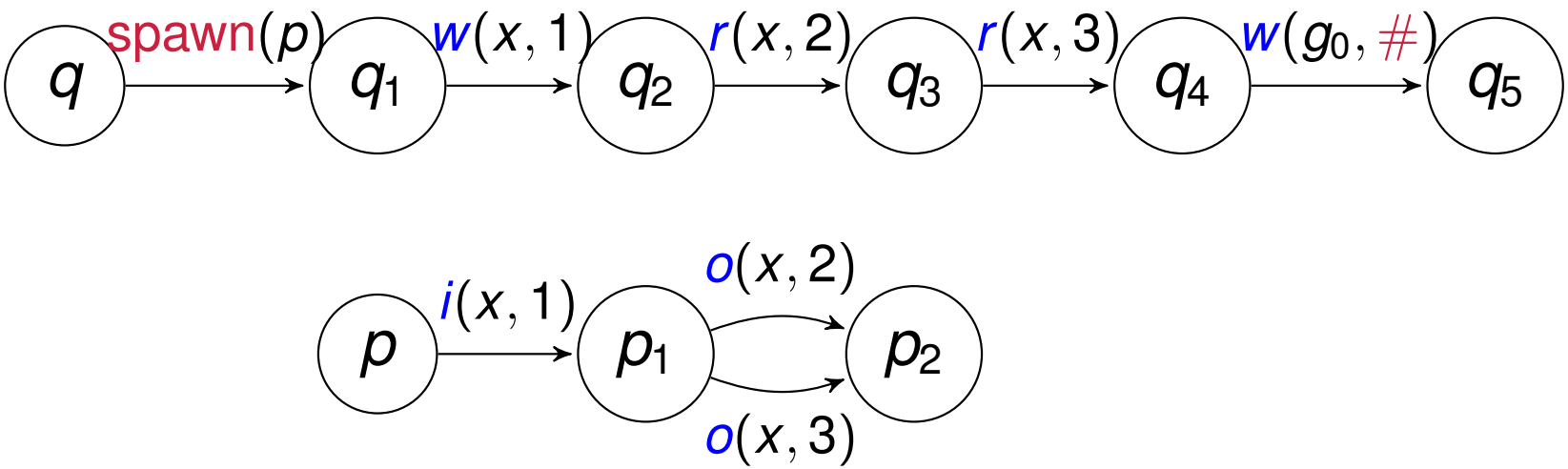
Dynamic parametric processes

Dynamic Process



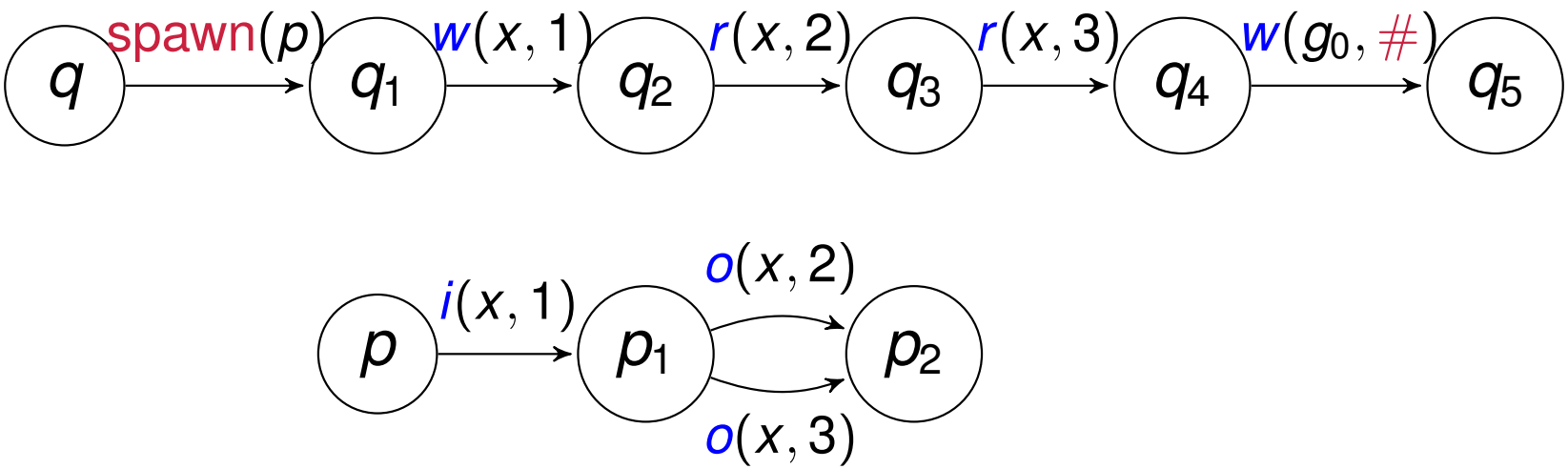


Dynamic Process



- spawn creates new process;
- r, w access own variable x;
- i, o access parent variable x.

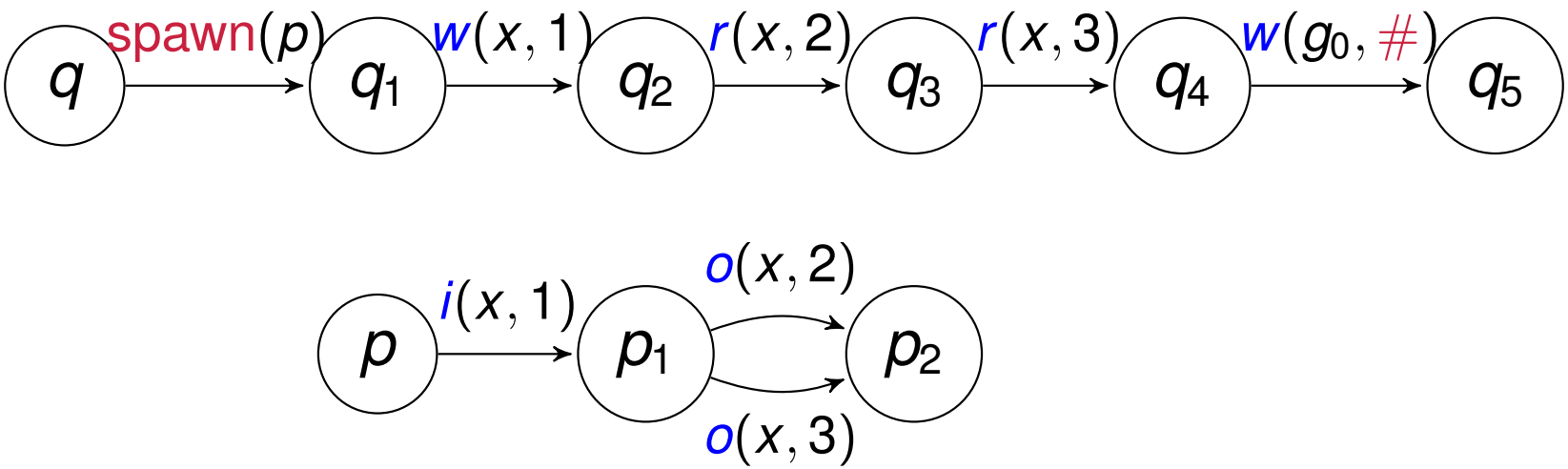
Dynamic Process



- spawn creates new process;
- r, w access own variable x;
- i, o access parent variable x.

Reachability: may # ever be written?

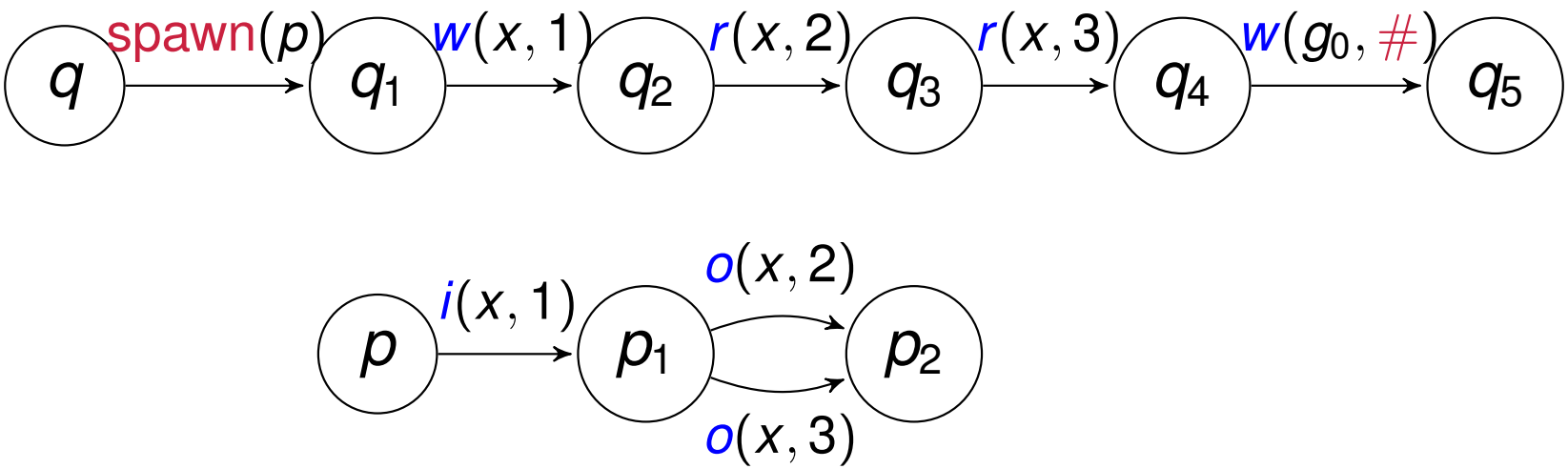
Dynamic Process



- spawn creates new process;
- r, w access own variable x;
- i, o access parent variable x.

Reachability: may # ever be written? **No**

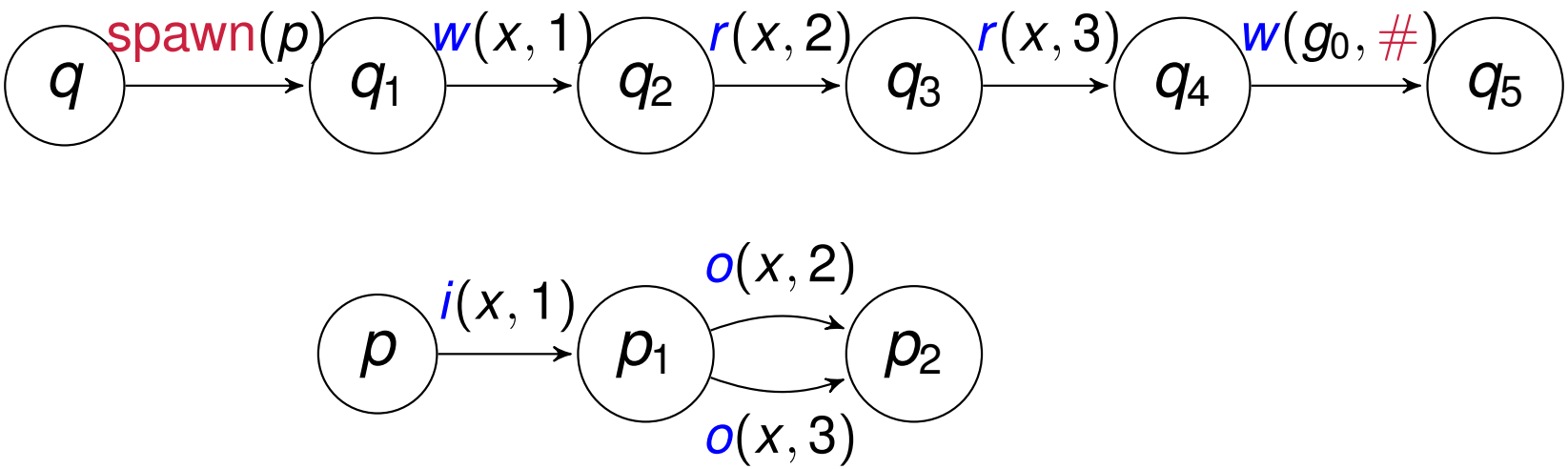
Dynamic Parametric Process



- spawn creates multiple new processes;
- r, w access own variable x;
- i, o access parent variable x.

Reachability: may # ever be written?

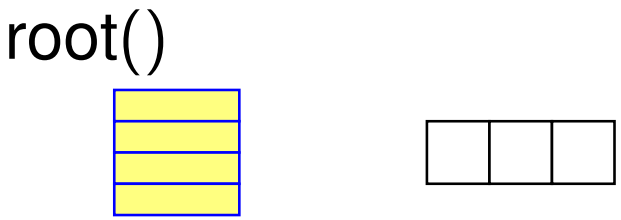
Dynamic Parametric Process



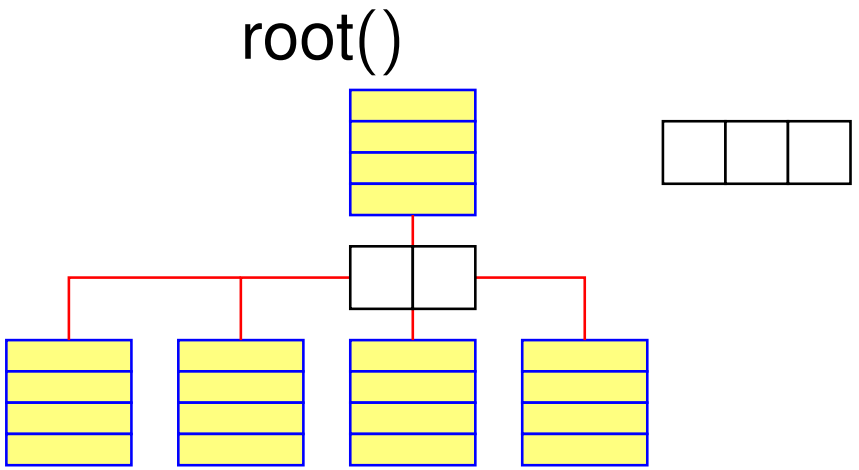
- spawn creates multiple new processes;
- r, w access own variable x;
- i, o access parent variable x.

Reachability: may # ever be written? **Yes**

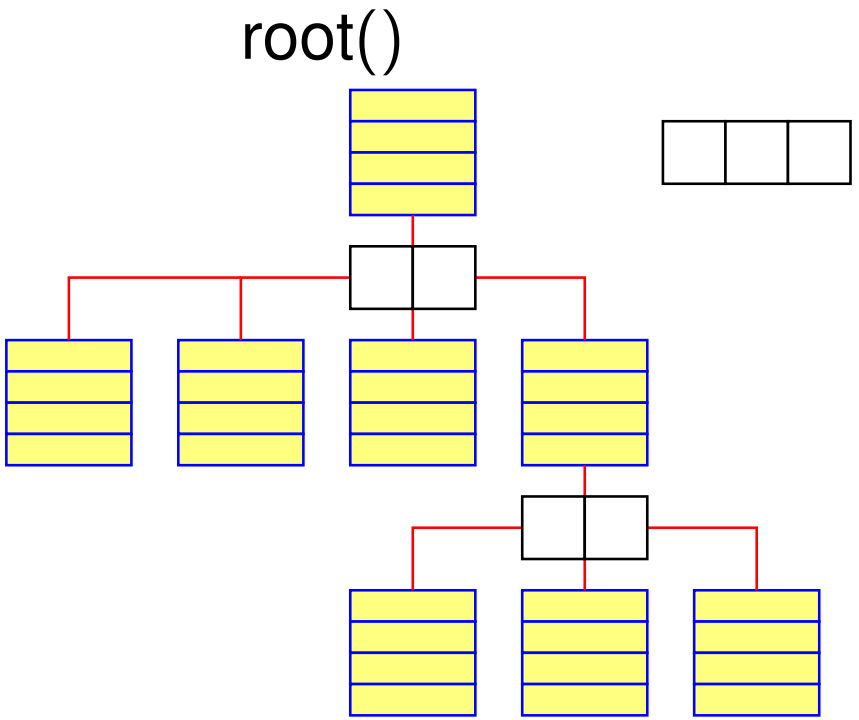
A Run



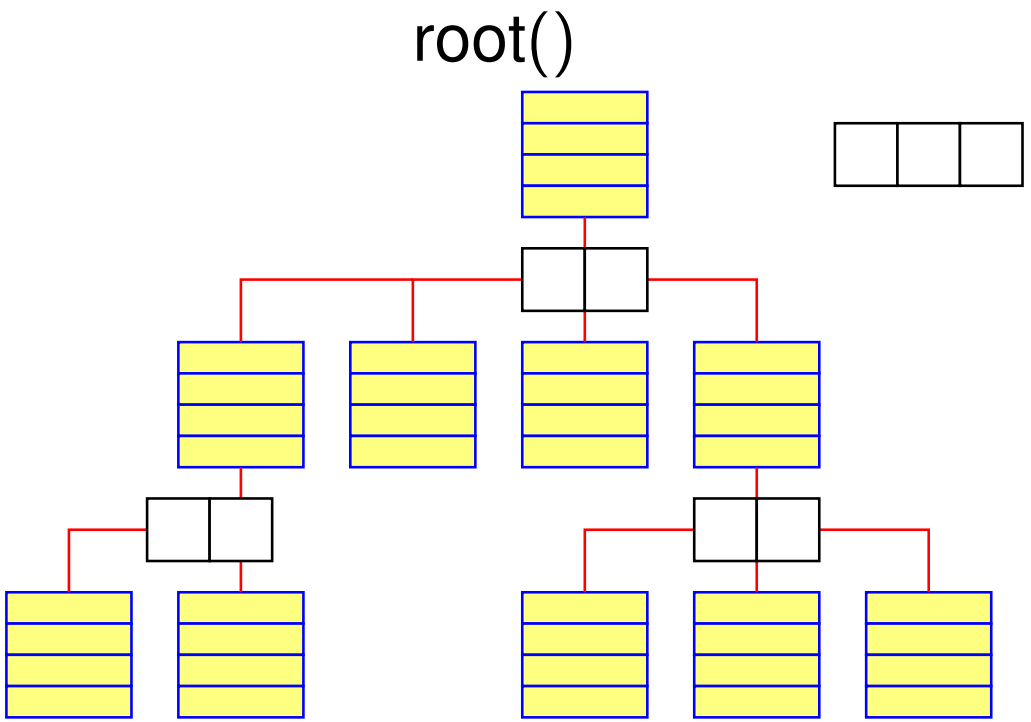
A Run



A Run



A Run



Our Results

Theorem 1

Reachability for dynamic processes is decidable.

Our Results

Theorem 1

Reachability for dynamic processes is decidable.

Theorem 2

In absence of locals, reachability is PSPACE complete.

Our Results

Theorem 1

Reachability for dynamic processes is decidable.

Theorem 2

In absence of locals, reachability is PSPACE complete.

Theorem 3

For generalized futures, reachability is DEXPTIME complete.

Abstraction 1: Set Semantics

Multiset Configuration

$$\langle p, ABBC, \{x \mapsto 1, y \mapsto 2\}, \{5 \cdot t_1, 2 \cdot t_2, 17 \cdot t_3\}\rangle$$

Abstraction 1: Set Semantics

Multiset Configuration

$$\langle p, ABBC, \{x \mapsto 1, y \mapsto 2\}, \{5 \cdot t_1, 2 \cdot t_2, 17 \cdot t_3\}\rangle$$

Set Configuration

$$\langle p, ABBC, \{x \mapsto 1, y \mapsto 2\}, \{ t_1 , t_2 , t_3 \}\rangle$$

Abstraction 1: Set Semantics

Multiset Configuration

$$\langle p, ABBC, \{x \mapsto 1, y \mapsto 2\}, \{5 \cdot t_1, 2 \cdot t_2, 17 \cdot t_3\}\rangle$$

Set Configuration

$$\langle p, ABBC, \{x \mapsto 1, y \mapsto 2\}, \{\mathrm{set}(t_1), \mathrm{set}(t_2), \mathrm{set}(t_3)\}\rangle$$

Abstraction 1: Set Semantics

Intuition

Child actions may be repeated arbitrarily.

Abstraction 1: Set Semantics

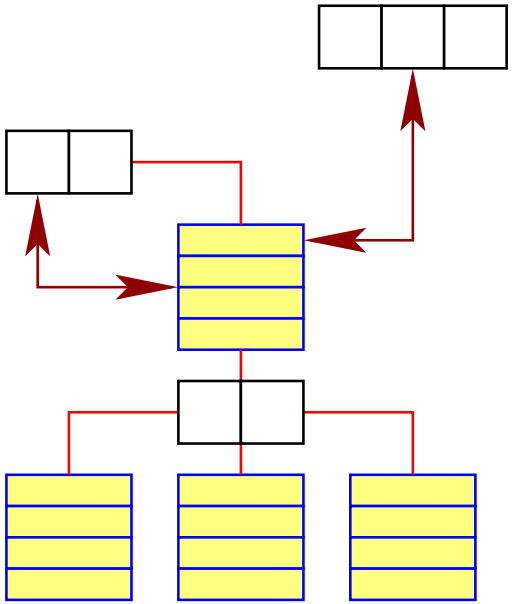
Intuition

Child actions may be repeated arbitrarily.

Proposition

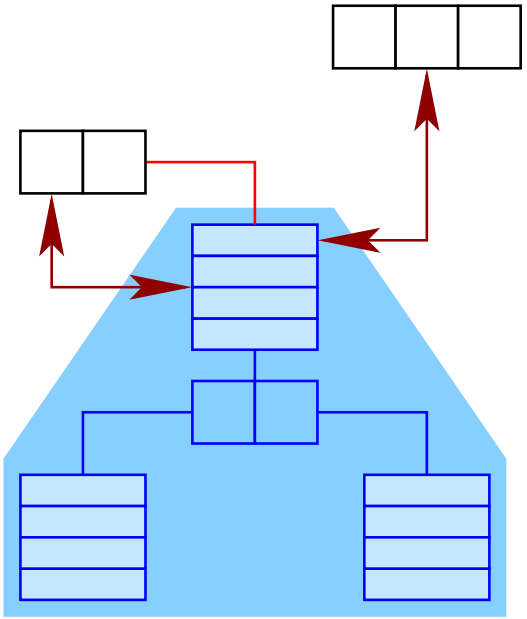
Multset Reachability $\equiv$ Set Reachability

External Behavior



Forget about internal actions!

External Behavior



Forget about internal actions!

Abstraction 2: Hypotheses

Hypothesis

$$\{\ \mathrm{spawn}(\mathrm{p}),\ \mathrm{spawn}(\mathrm{p})\ i(x,1),\ \mathrm{spawn}(\mathrm{p})\ i(x,1)\ o(x,2),\ \mathrm{spawn}(\mathrm{p})\ i(x,1)\ o(x,3)\ \}$$

$\Longleftrightarrow$ all external behaviors of child processes.

Abstraction 2: Hypotheses

Hypothesis

$$\{\ \mathrm{spawn}(\mathrm{p}),\ \mathrm{spawn}(\mathrm{p})\, i(x,1),\ \mathrm{spawn}(\mathrm{p})\, i(x,1)\, o(x,2),\ \mathrm{spawn}(\mathrm{p})\, i(x,1)\, o(x,3)\ \}$$

$\Longrightarrow$ all external behaviors of child processes.

Abstraction

Set of child processes $\mapsto$ Set of external behaviors

Abstraction 2: Hypotheses

Intuition

Internal activities of child processes are irrelevant.

Abstraction 2: Hypotheses

Intuition

Internal activities of child processes are irrelevant.

Problem

Hypotheses may be

- infinite

Abstraction 2: Hypotheses

Intuition

Internal activities of child processes are irrelevant.

Problem

Hypotheses may be

- infinite
- complicated

Abstraction 3: Lifted Hypotheses

Arbitrarily insert earlier actions ...

Abstraction 3: Lifted Hypotheses

Arbitrarily insert earlier actions ...

$$\mathrm{spawn}(\mathrm{p})\; i(x,1)\; o(x,2) \quad \in$$

$$\mathrm{lift}\ \{\ \ \mathrm{spawn}(\mathrm{p}),\ \ \mathrm{spawn}(\mathrm{p})\; i(x,1),\ \ \mathrm{spawn}(\mathrm{p})\; i(x,1)\; o(x,2),\ \ \mathrm{spawn}(\mathrm{p})\; i(x,1)\; o(x,3)\ \ \}$$

Abstraction 3: Lifted Hypotheses

Arbitrarily insert earlier actions ...

$$\mathrm{spawn}(\mathrm{p})\; i(x,1)\; o(x,2)\; i(x,1) \quad \in$$

$$\mathrm{lift}\ \{\ \ \mathrm{spawn}(\mathrm{p}),$$
$$\mathrm{spawn}(\mathrm{p})\; i(x,1),$$
$$\mathrm{spawn}(\mathrm{p})\; i(x,1)\; o(x,2),$$
$$\mathrm{spawn}(\mathrm{p})\; i(x,1)\; o(x,3)\ \ \}$$

Abstraction 3: Lifted Hypotheses

Proposition

H_1 is equivalent to H_2 iff $\mathrm{lift}(H_1) = \mathrm{lift}(H_2)$.

Abstraction 3: Lifted Hypotheses

Proposition

H_1 is equivalent to H_2 iff $\mathrm{lift}(H_1) = \mathrm{lift}(H_2)$.

Observation

- Consider minimal set $\mathrm{core}(H) \subseteq H$ with

$$\mathrm{lift}(\mathrm{core}(H)) = \mathrm{lift}(H)$$

Abstraction 3: Lifted Hypotheses

Proposition

H_1 is equivalent to H_2 iff $\mathrm{lift}(H_1) = \mathrm{lift}(H_2)$.

Observation

- Consider minimal set $\mathrm{core}(H) \subseteq H$ with

$$\mathrm{lift}(\mathrm{core}(H)) = \mathrm{lift}(H)$$

- $\mathrm{core}(H)$ is finite.

Deciding Reachability

$$\mathsf{Ext}_k = \{\text{external behaviors of conf. of depth} \leq k\}$$

Deciding Reachability

$$\mathrm{Ext}_k = \{\text{external behaviors of conf. of depth} \leq k\}$$

- Determine $\mathrm{core}(\mathrm{Ext}_0)$.

Deciding Reachability

$$\mathrm{Ext}_k = \{\text{external behaviors of conf. of depth} \leq k\}$$

- Determine $\mathrm{core}(\mathrm{Ext}_0)$.
  // plain pushdown system

Deciding Reachability

$\mathrm{Ext}_k = \{\text{external behaviors of conf. of depth} \leq k\}$

- Determine $\mathrm{core}(\mathrm{Ext}_0)$.
  // plain pushdown system
- Determine $\mathrm{core}(\mathrm{Ext}_k)$ for $k > 0$.
  // plain pushdown system

Deciding Reachability

$\mathrm{Ext}_k = \{\text{external behaviors of conf. of depth} \leq k\}$

- Determine $\mathrm{core}(\mathrm{Ext}_0)$.
  // plain pushdown system
- Determine $\mathrm{core}(\mathrm{Ext}_k)$ for $k > 0$.
  // plain pushdown system
- Stop when $\mathrm{core}(\mathrm{Ext}_k) = \mathrm{core}(\mathrm{Ext}_{k-1})$.

Wrap-up

Theorem 1

Reachability for dynamic parametric processes is decidable.

Wrap-up

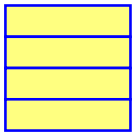
Theorem 1

Reachability for dynamic parametric processes is decidable.

The same holds true for higher-level dynamic processes.

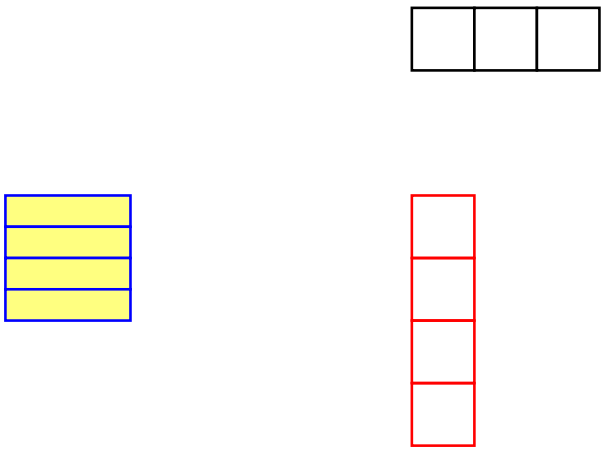
Extension to Weak Memory

PSO



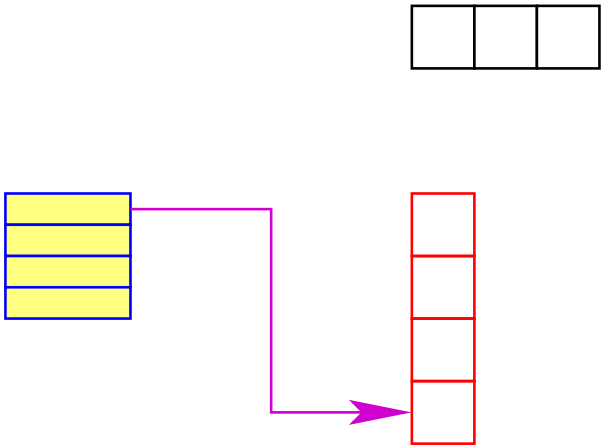
Extension to Weak Memory

PSO



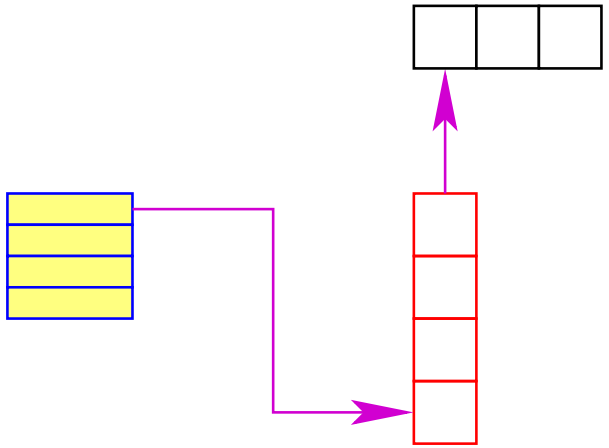
Extension to Weak Memory

PSO



Extension to Weak Memory

PSO



Extension to Weak Memory

Key construction of overall approach now becomes

Extension to Weak Memory

Key construction of overall approach now becomes

- Pushdown
- channels
- finite hypothesis on children

Extension to Weak Memory

Key construction of overall approach now becomes

- Pushdown
- channels — which are lossy!
- finite hypothesis on children

Extension to Weak Memory

Key construction of overall approach now becomes

- Pushdown
- channels — which are lossy!
- finite hypothesis on children

Conjecture: Reachability is decidable

Summary

- Parametrization is a useful abstraction to deal with unbounded concurrency.

Summary

- Parametrization is a useful abstraction to deal with unbounded concurrency.
- It is applicable to PSO.

Summary

- Parametrization is a useful abstraction to deal with unbounded concurrency.
- It is applicable to PSO.
- How to parameterize model / results with various kinds of WMs ?

Thank you!