TECHNISCHE UNIVERSITÄT DRESDEN

# At a glance:

# LiQuor / ProbMela

in context of
Lorentz Workshop 11/07
Leiden NL

Christel Baier
Frank Ciesinski
Marcus Größer
Joachim Klein

# Overview

- ProbMela
  - Guarded command language
  - Strongly inspired by PROMELA (SPIN)
  - Probabilistic extensions

- LiQuor
  - Explicit model checker for quant. LTL and MDPs

$$\mathcal{M} \overset{?}{\models} [\varphi]_{\leq p} \equiv \mathcal{M} \overset{?}{\models} [\neg\varphi]_{\geq (1-p)}$$

MDP

LTL

# ProbMela

- Imperative probabilistic guarded command language
- Similarities to *PROMELA*:
  - atomic, numeric datatypes, **`do...od`**, **`if...fi`** (nondeterministic choices), channels (synchronous and asynchronous), assertions, partial order reduction

- Differences /Extensions to *PROMELA*:
  - Minor syntactical differences
  - Probabilistic choice (**`pif...fip`**),
  - Lossy channels
  - MDP semantics
  - Complex data structures (members of classes)
  - Recursive functions (functions of classes)
  - Numerical solution is performed

- Lossy channels
  - `lossy 0.3 channel of {int} size 8`
- `c!x` loses message (i.e. = `skip`) with probability 0.3
- Probabilistic choice
  - `pif :0.3:-> value = 3`
    `:0.7:->goto exit`
    `fip`

# ProbMela process handling:

```
constant int N = 2;

bool p_bootstrapped=false;

proctype p(int par){
   p_bootstrapped;
   skip
}

active proctype starter()
{
int i=0;
p[N] procs;
    do::i<N ~>
        procs[i] = p.create(i);
        procs[i].start(); i++
      ::i==N~>
        atomic{
               p_bootstrapped = true;
               me.destroy()
        }
    od;
}
```

wait

create processes

start processes

destroy myself

# ProbMela defines:

```
define tryEat(l,r)={
   if   ::forks[l]==false~>forks[l]=true;
           if
             ::forks[r]==false~>forks[r]=true;
               eat++;eat--;
               forks[r]=false;forks[l]=false
             ::forks[r]==true~>forks[l]=false
           fi
        :: forks[l]==true~>skip
    fi
}

...
bool[N] forks;
...
tryEat(_mypid,_mypid+1%N)
...
```

# ProbMela functions:

```
function fak(int n)
{
  if::n==1 returnurn(n)
    ::n> 1 return(n*fak(n-1))
  fi
}
```

# Bytecode principle

```
// Probmela
proctype p(){
...

i=42;

...
}
```

```
// PASM

_14_0_guard:
   guard[src = "i = 42 {15,0
       permit;
       loadca1 "_14_0";
       loadc16 "1";
       ndetenable;
   }

_14_0:
   code[src = "i = 42 {15,0}"]{
       loadc "42";
       storev "i";
       loadca1 "null";
   }

null:
   guard[src = "end_of_process"]{
       permit;
   }
```

Beispiel:
pushretaddr

- Semantical clarity “at least” at PASM level
- Very flexible to implement

# Visualization of state space

processcreation

File Help

Model | Model Checking Method | Statistics and Results | Other Options | Developer's options

Program Graph Construction | DFS on MDP | Quantitative Reachability in MDP | Quantitative A

**A depth first search on the whole state space will be performed. (Properties are proce**

Maximum search depth: 1 Millions — Apply partial order reduction

Size of hashtable: 1 Millions — LTL property: property.de

**Visualization:**

Graphviz | JFF export | State vector dump

LTL formula is "F a" with the fol
propositions:
a --> global.i==1

create graphviz file (enable only for "small" systems!)

Graphviz Output File: transition_system.graphviz ... Edi

speed variables:

profiling counters:
text fields:
model checking was completed
*** execution complete

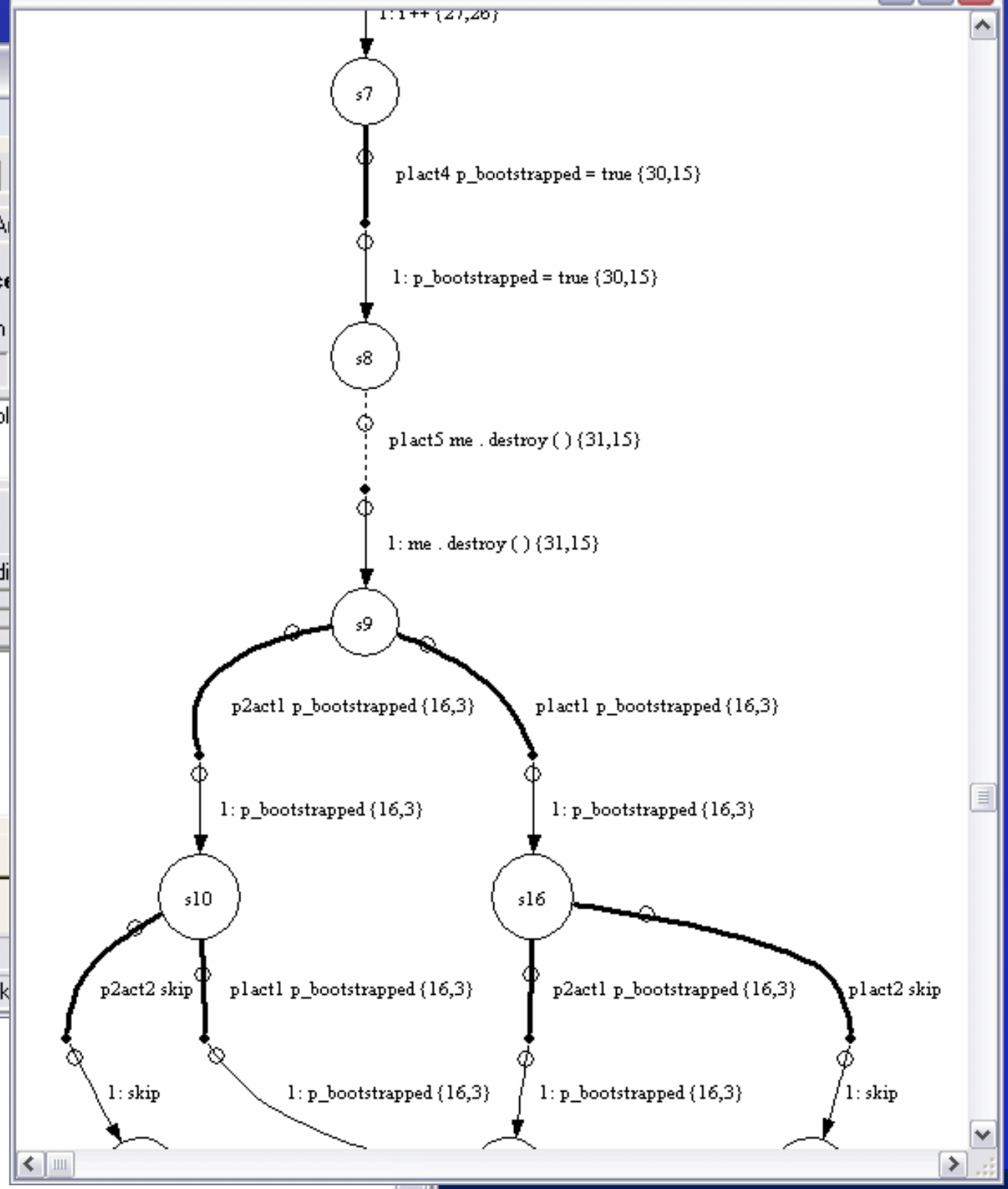
Open shell | Explorer | Full model checking | Go

2029230 bytes

DFS only | unchanged | C:\Dokumente und Einstellungen\ciesinsk\Desktop\LiQuorCheck

```
        atomic{
                p_bootstrapped = true;
                me.destroy()
        }
    od;
}
```

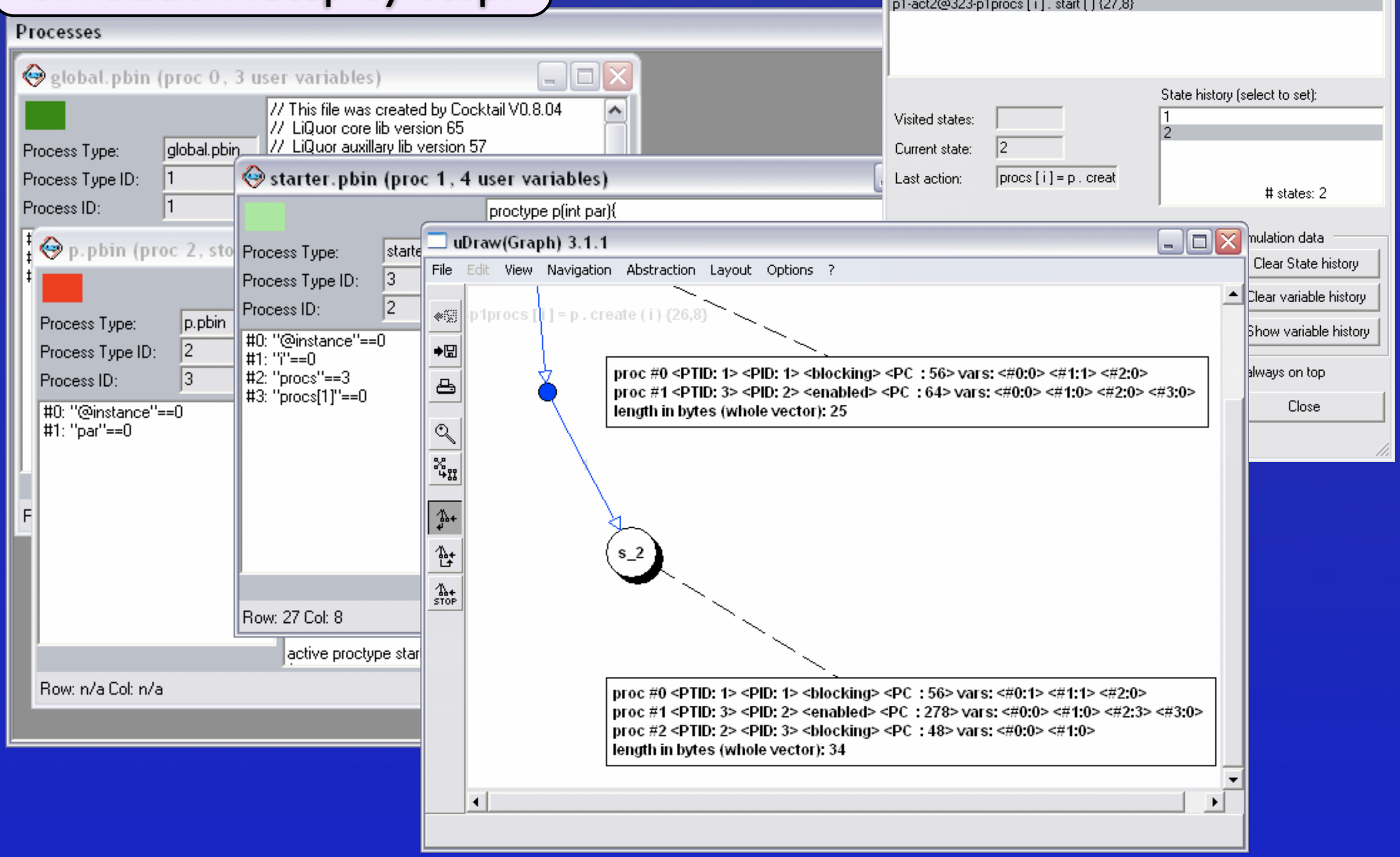
DOTTY



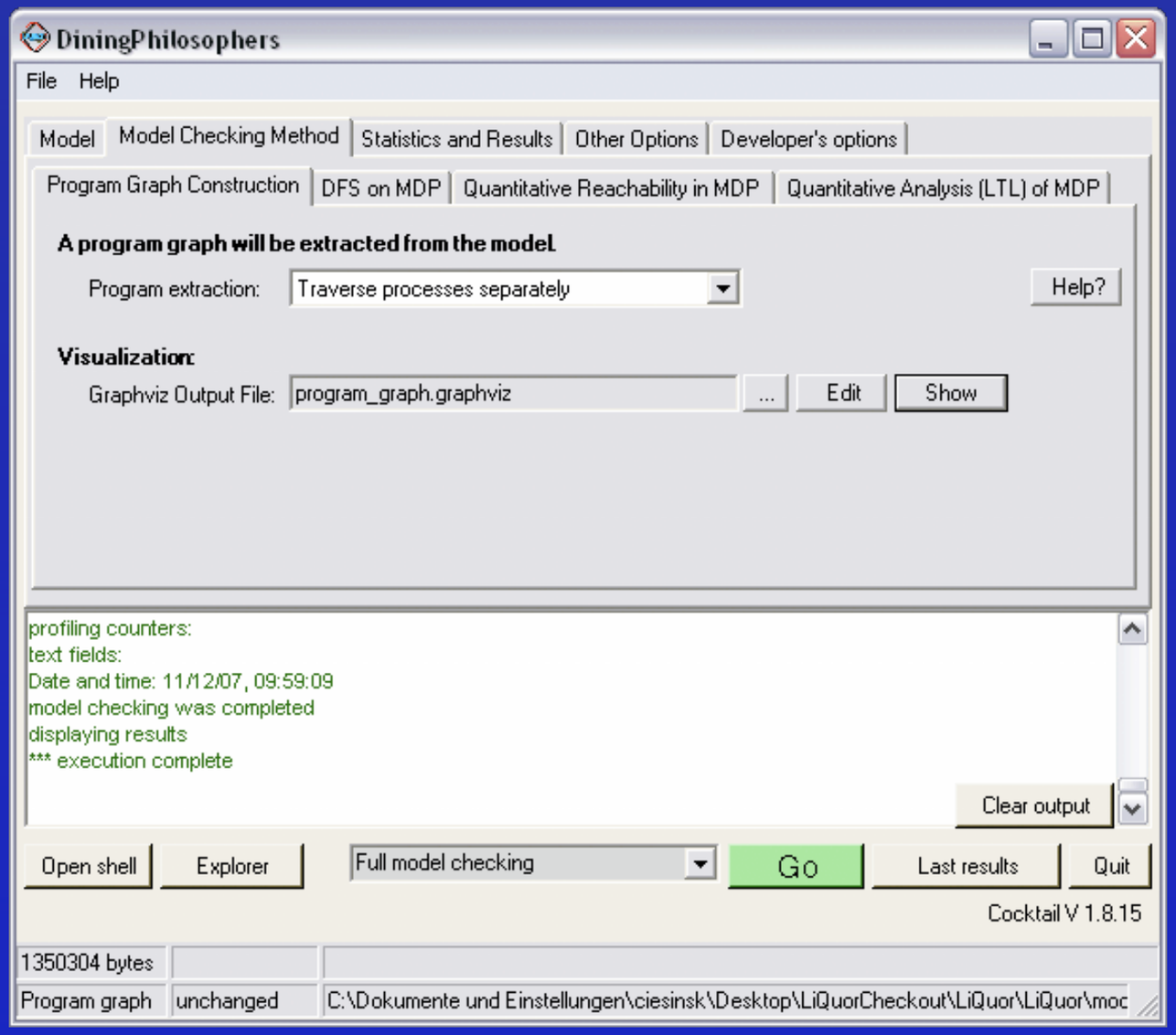
# Quantitative analysis:

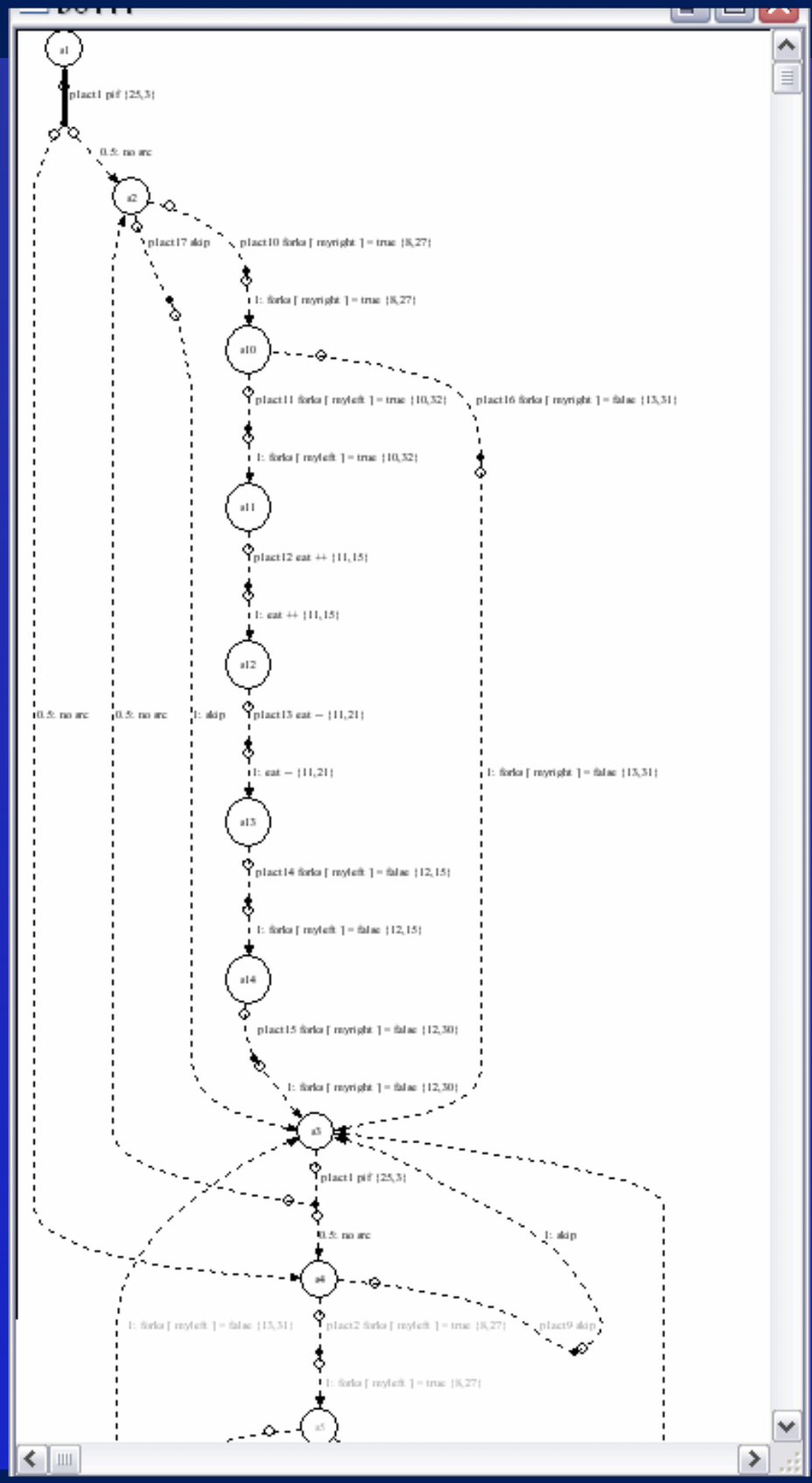
- Numerical Solution in LiQuor:
  - Value Iteration
  - lpsolve (free LP lib)
  - export to file

# Simulation /Step by step:

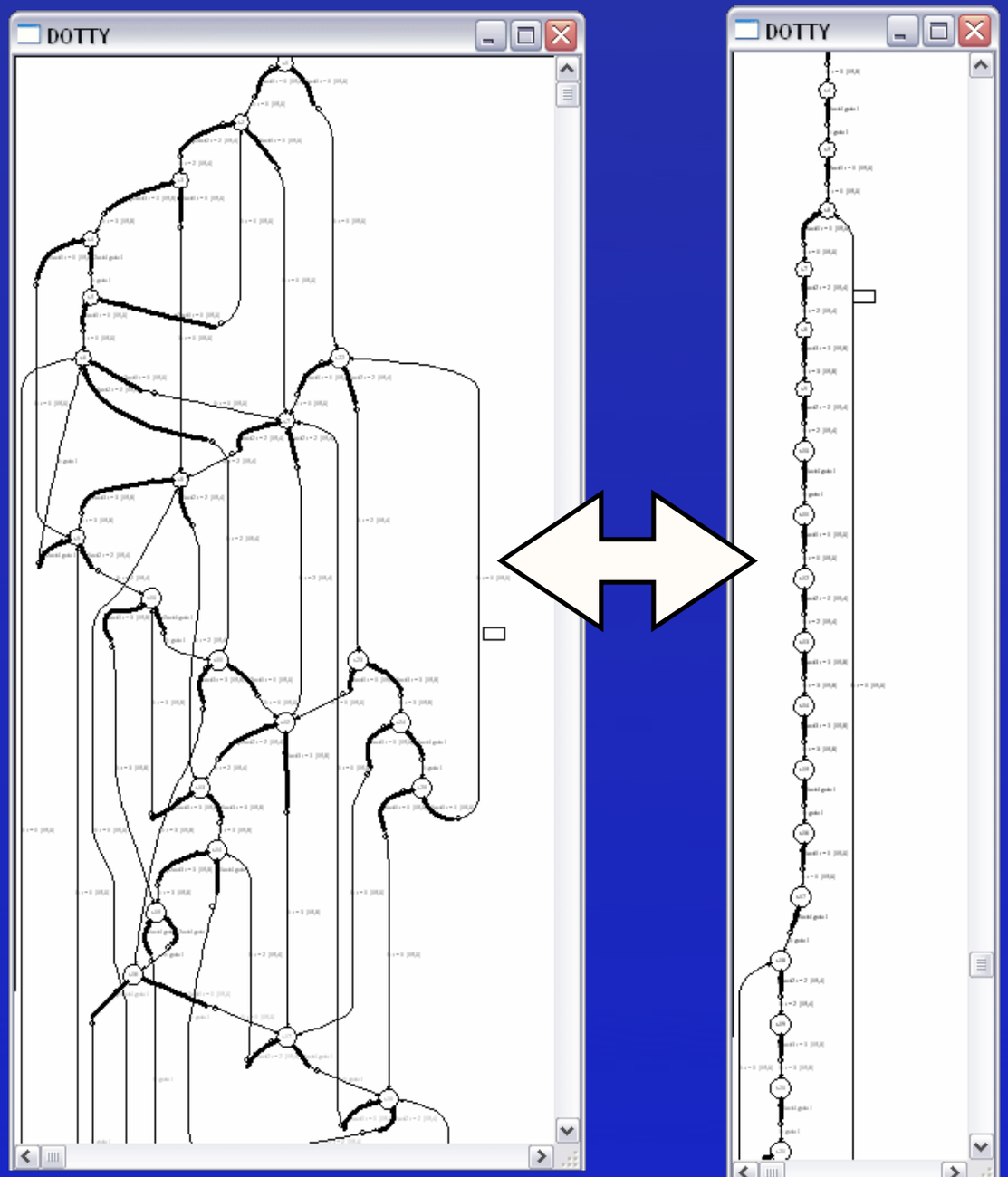


# Program Graph analysis:



# Partial Order Reduction:







| 6 Dining Philosophers | w/o POR | with POR |
|---|---|---|
| states | 411255 | 305757 |
| actions | 2875869 | 1133974 |

| 5 Leader Election (as) | w/o POR | with POR |
|---|---|---|
| states | >4 Mio | 967855 |
| actions | >12 Mio | 1.5 Mio |

Reachability analysis

- **LiQuor for Research:**
  - **A flexible and dynamic language**
  - **Process handling is very dynamic**
  - **LTL**
  - **explicit ( :-) or :-( ???)**

- **Extending LiQuor**
  - **Cooperations are very welcome**

- **Availability of LiQuor**:
  - http://wwwtcs.inf.tu-dresden.de/~ciesinsk/transfer_open/liquor_download.html

- **Available for Windows only**