A Set of Performance and Dependability Analysis Components for CADP

Holger Hermanns

Department of Computer Science
Universiteit Twente
The Netherlands

and

Department of Computer Science
Universität des Saarlandes
Germany

Christophe Joubert

VASY
INRIA Rhône-Alpes
France

April 10, 2002

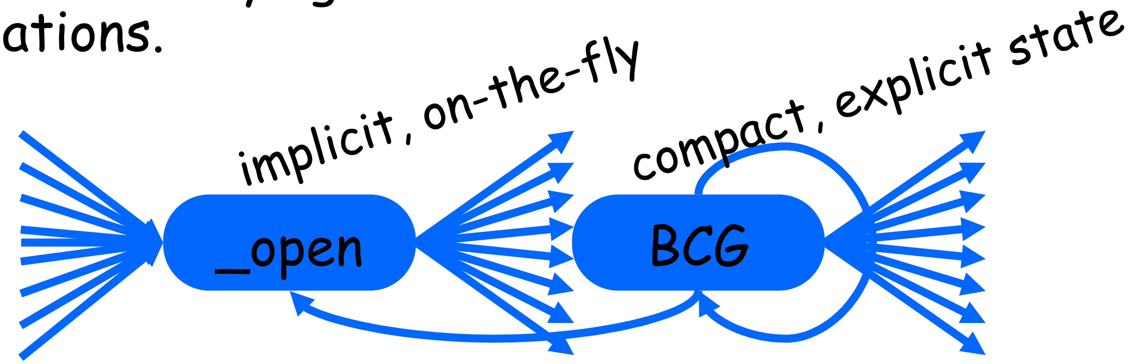
Overview

- What's the tool?
- What's the extension?
- What's the basis?
- What's the example?
- What's under the hood?

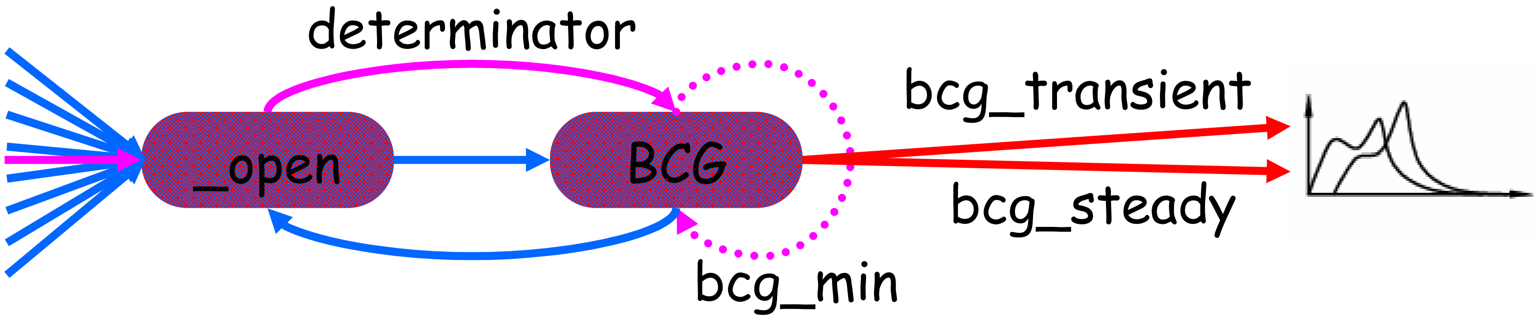
What's the tool?

CADP

- One of the leading verification toolboxes in academia.
- Offers various tools for
  - visualization, simulation,
  - equivalence checking,
  - model checking.
- Open platform supporting integration of other specification, verification and analysis techniques.
- Originally designed for verifying correctness of LOTOS specifications.



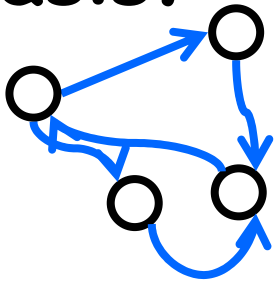
What's the extension?



- Means to specify performance and dependability characteristics.
- Algorithms to construct performance and dependability models.
- Basic analysis algorithms.

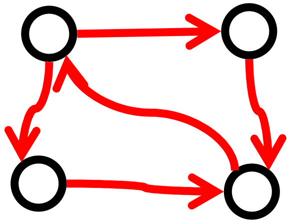
What's the basis?

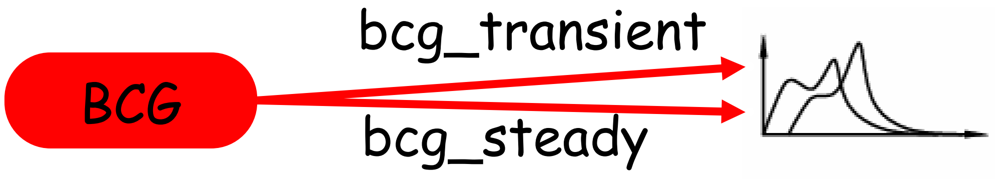
- Labelled transition systems



BCG

- Markov chains





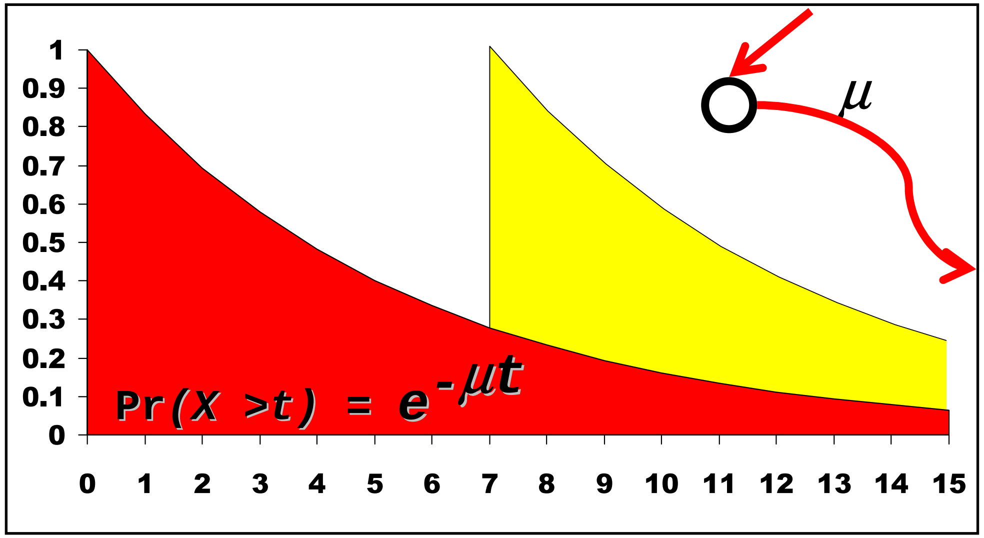
BCG

bcg_transient

bcg_steady

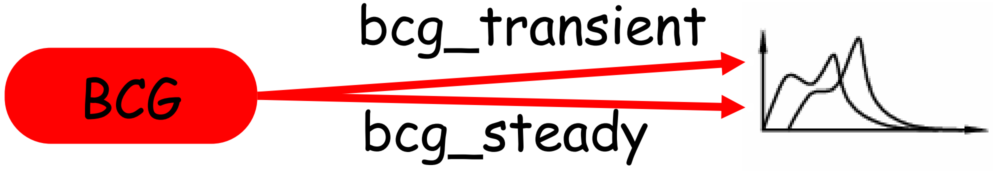
(Continuous-time) Markov chains (MC)

- (finite-state) automata,
- all times are *exponentially distributed*,



$Pr(X > t) = e^{-\mu t}$

- sojourn time in states are *memory-less*,



bcg_transient

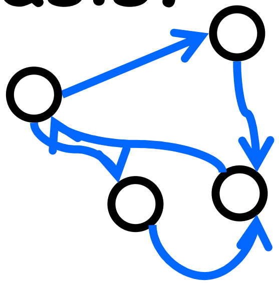
BCG

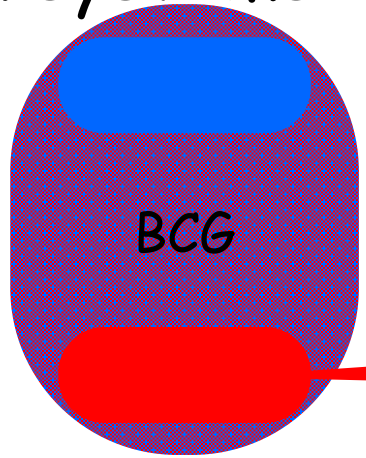
bcg_steady

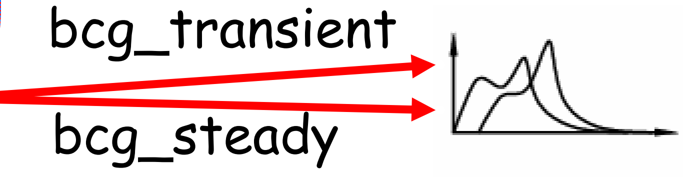
- very well investigated class of stochastic processes,
- widely used in practice,
- best guess, if only mean values are known,
- superpositions can approximate arbitrary continuous distributions,
- *efficient* and numerically *stable* algorithms for *steady-state* and *transient* analysis are available.

What's the basis?

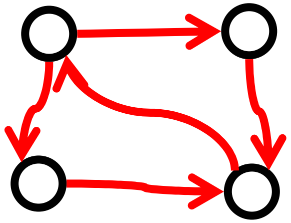
- Labelled transition systems



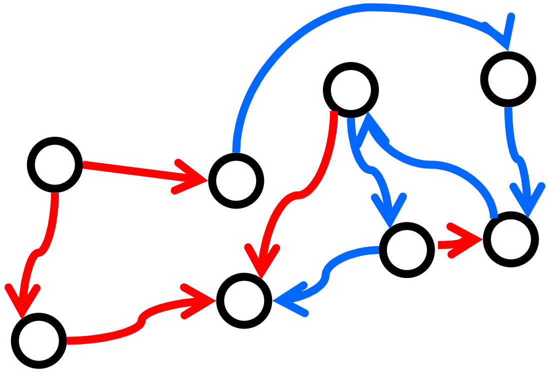


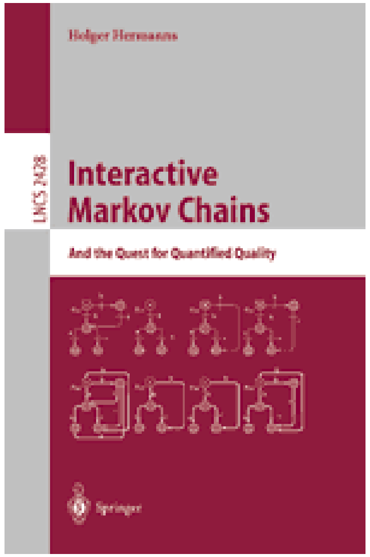


- Markov chains



- Interactive Markov Chains





Holger Hermanns

LNCS 2428

Interactive Markov Chains

And the Quest for Quantified Quality

Springer

Interactive Markov Chains in CADP

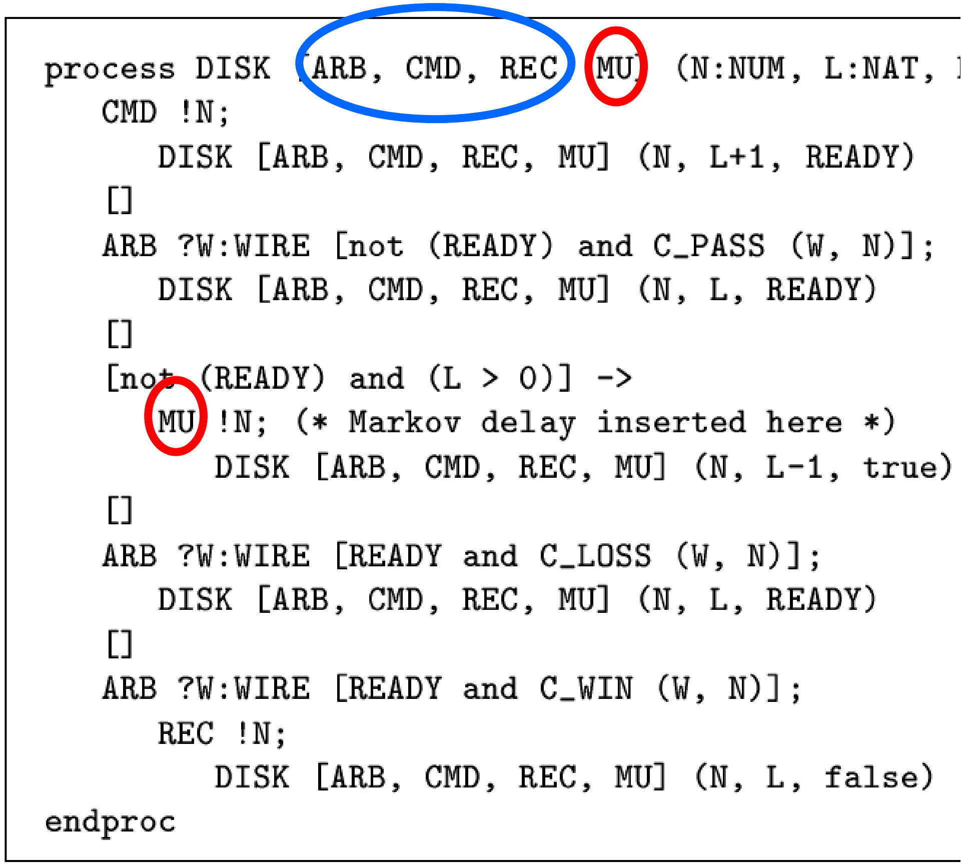
BCG level

- two types of transititions
  - $\underset{\texttt{CMD}}{\longrightarrow}$
  - $\underset{\texttt{rate 24.0}}{\longrightarrow}$

  in the state space representation

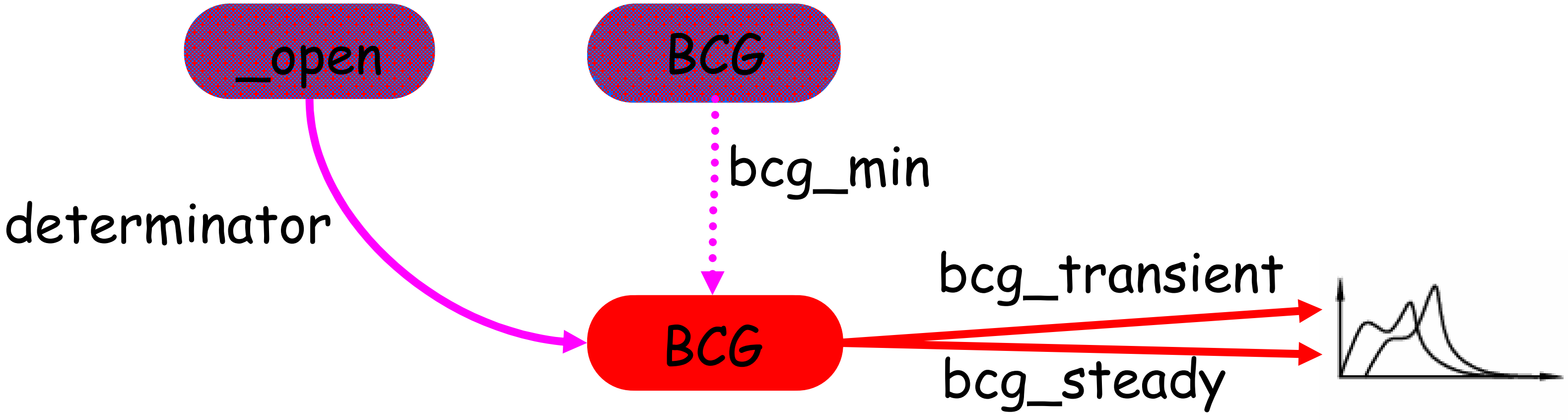
Specification level

- pragmatic
- user defined
  (and user maintained)
  separation of 'gates' into two types
  - gates
  - rates
- mapped on concrete values via generalised renaming (bcg_labels)



```
process DISK [ARB, CMD, REC, MU] (N:NUM, L:NAT, I
   CMD !N;
      DISK [ARB, CMD, REC, MU] (N, L+1, READY)
   []
   ARB ?W:WIRE [not (READY) and C_PASS (W, N)];
      DISK [ARB, CMD, REC, MU] (N, L, READY)
   []
   [not (READY) and (L > 0)] ->
      MU !N; (* Markov delay inserted here *)
         DISK [ARB, CMD, REC, MU] (N, L-1, true)
   []
   ARB ?W:WIRE [READY and C_LOSS (W, N)];
      DISK [ARB, CMD, REC, MU] (N, L, READY)
   []
   ARB ?W:WIRE [READY and C_WIN (W, N)];
      REC !N;
         DISK [ARB, CMD, REC, MU] (N, L, false)
endproc
```

What's under the hood?



determinator

- on-the-fly generation of *MC*
- implements a determinacy check *partial!*

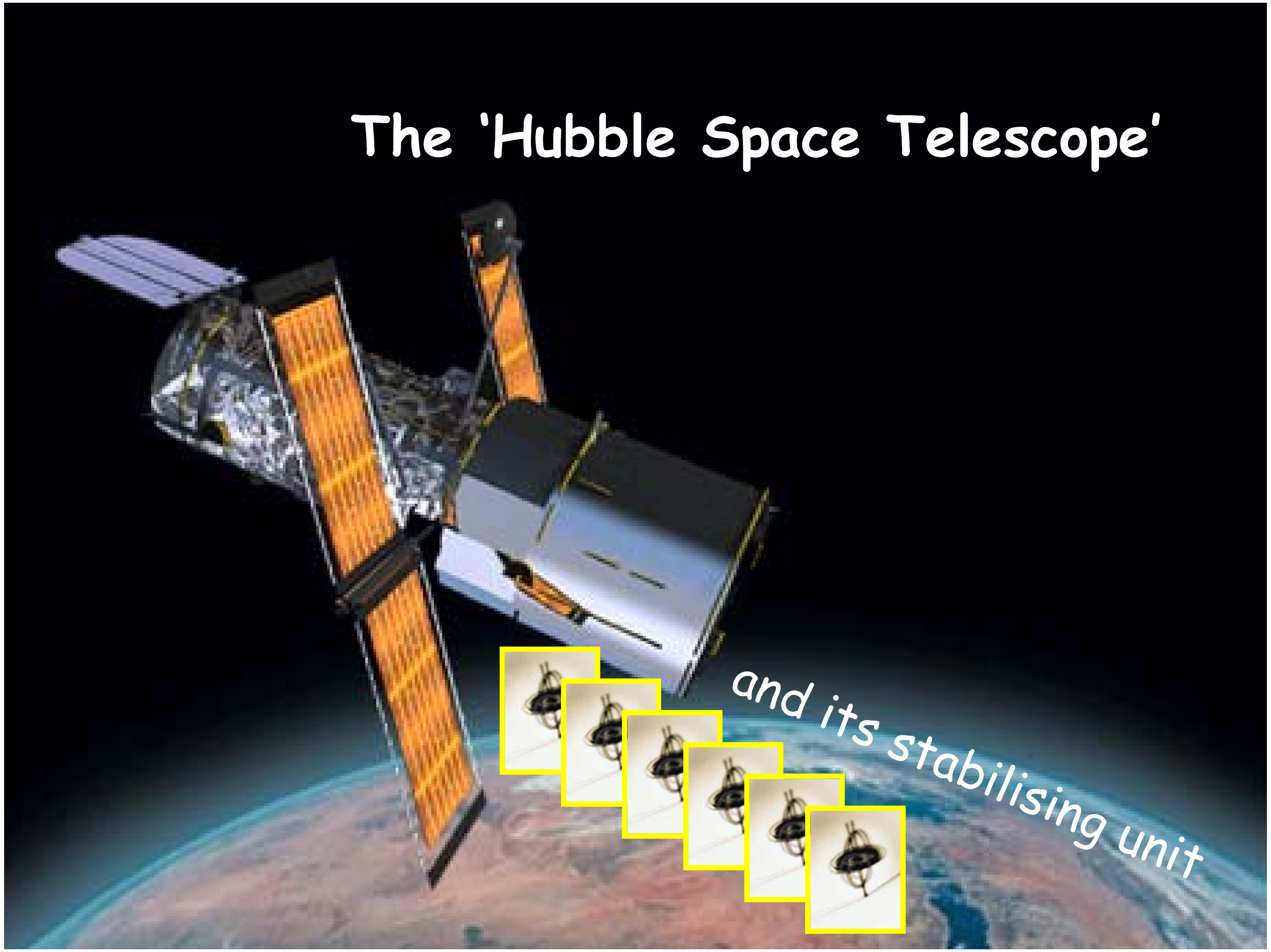
  based on [Ciardo/Zijal,Deavours/Sanders]

bcg_min

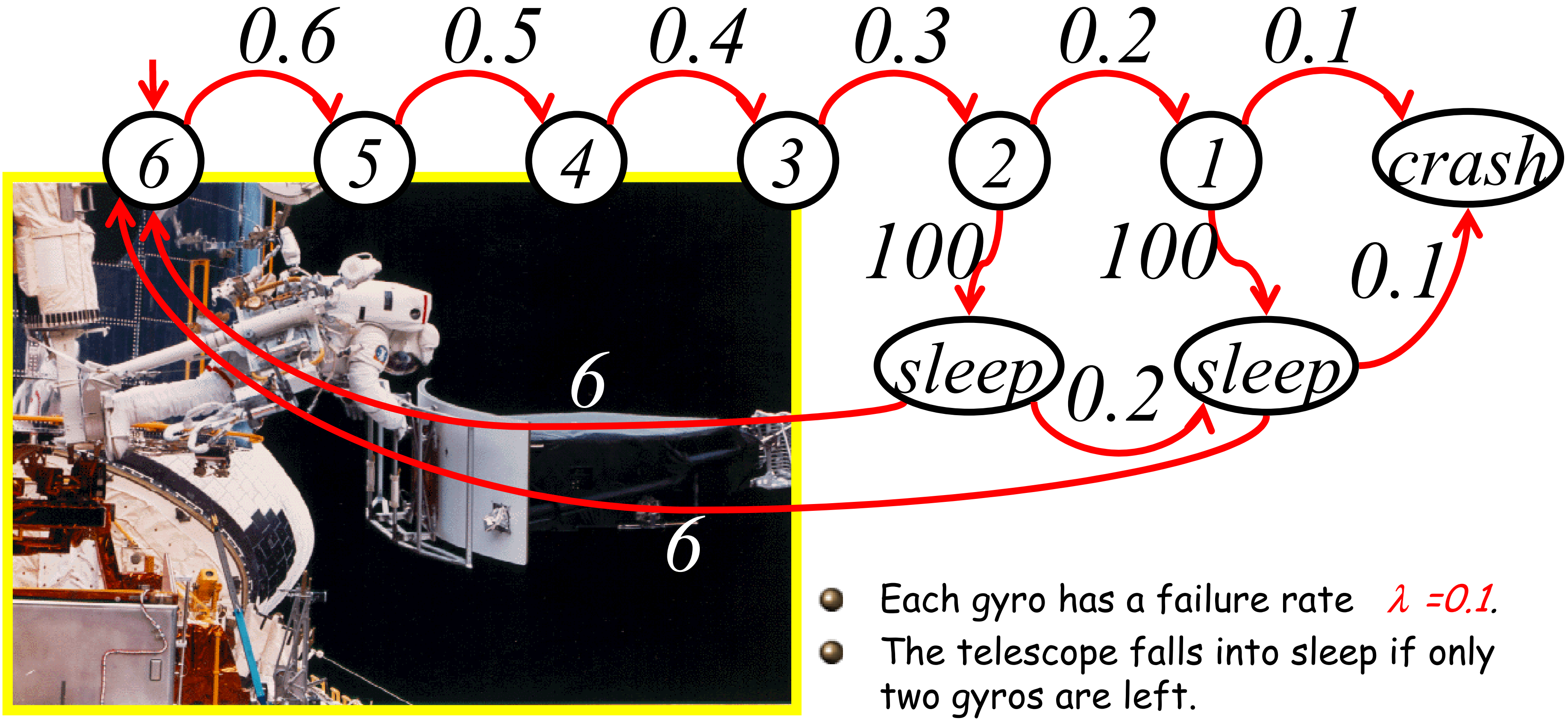
- branching bisimulation minimizer for LTS, for IMC and MC
- can turn an IMC into a minmal MC *partial!*

What's the example?

- A toy example we can demo online



A simple Markov model of the Hubble

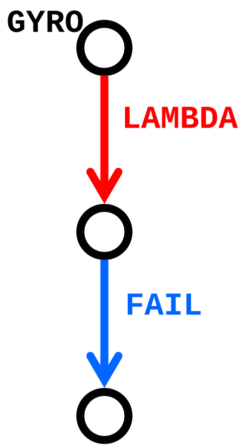


- Each gyro has a failure rate λ =0.1.
- The telescope falls into sleep if only two gyros are left.
- To turn on sleep mode requires some time (μ = 100).
- Without operational gyro, the telescope crashes.

- The base station prepares a shuttle mission to repair the telescope which takes about two months (ν =6).

The Hubble in LOTOS

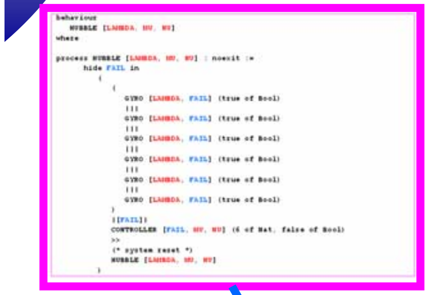
```
behaviour
   HUBBLE [LAMBDA, MU, NU]
where

process HUBBLE [LAMBDA, MU, NU] : noexit :=
      hide FAIL in
         (
            (
               GYRO [LAMBDA, FAIL] (true of Bool)
               |||
               GYRO [LAMBDA, FAIL] (true of Bool)
               |||
               GYRO [LAMBDA, FAIL] (true of Bool)
               |||
               GYRO [LAMBDA, FAIL] (true of Bool)
               |||
               GYRO [LAMBDA, FAIL] (true of Bool)
               |||
               GYRO [LAMBDA, FAIL] (true of Bool)
            )
            |[FAIL]|
            CONTROLLER [FAIL, MU, NU] (6 of Nat, false of Bool)
            >>
            (* system reset *)
            HUBBLE [LAMBDA, MU, NU]
         )
```



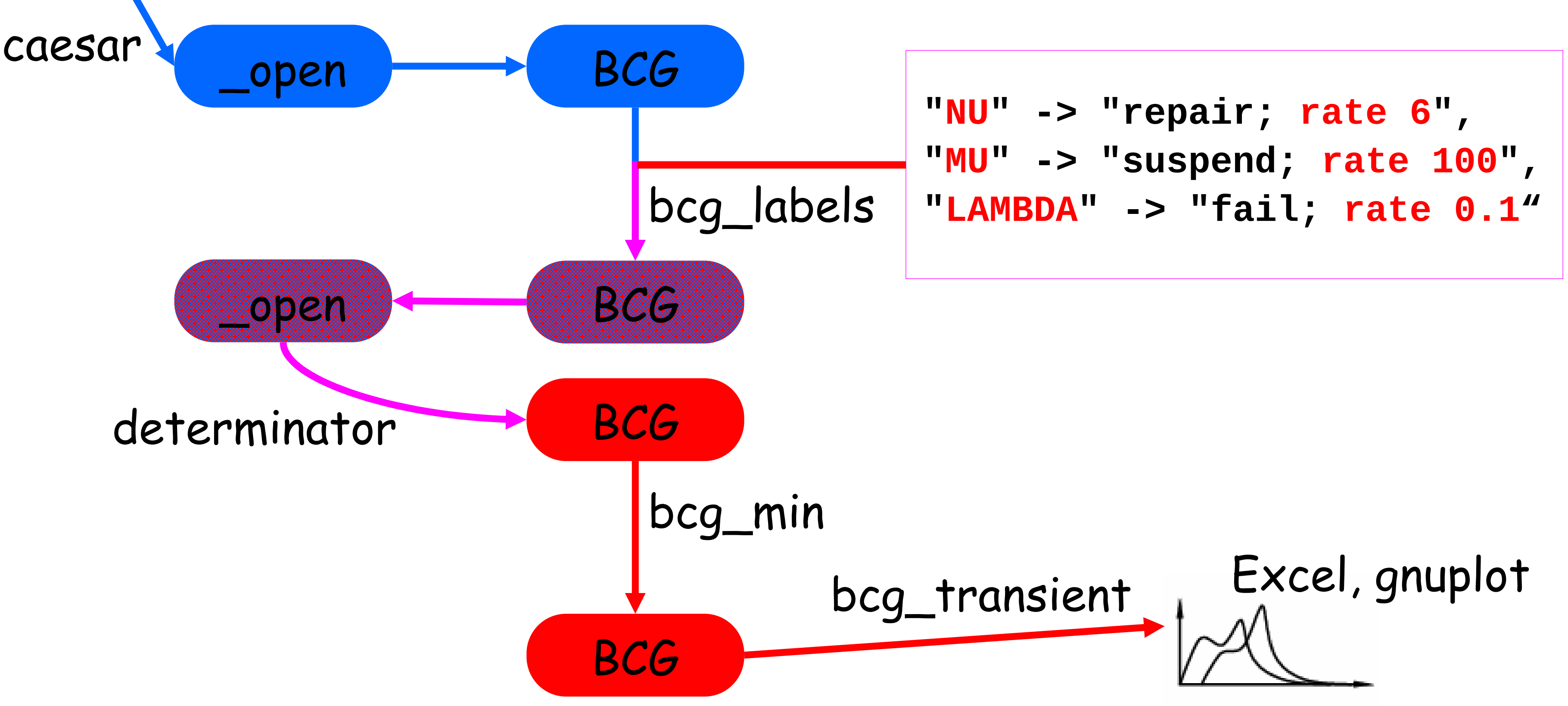
The Hubble in LOTOS (cont.)

```
process CONTROLLER [FAIL, MU, NU] (C : Nat, SLEEP : Bool) : exit :=
         (* Still gyros left *)
         [(C > 0)] ->
            (* Ah, a gyro failed. Let's count down. *)
            FAIL;
               CONTROLLER [FAIL, MU, NU] (C - 1, SLEEP)
         []
         (* Hubble starts tumbling. *)
         [(C < 3) and (SLEEP eq false)] ->
            (* Time to turn on the SLEEP mode. *)
            MU;
               CONTROLLER [FAIL, MU, NU] (C, true)
         []
         (* Sleep mode is on. *)
         [(SLEEP eq true)] ->
            (* Let's wait for the space mission to reset the system. *)
            NU;
               exit
         []
         (* No gyros left. *)
         [C == 0] ->
            (* Crash! *)
            i;
               stop
      endproc
   endproc
```

Analysis trajectory for the Hubble example



```
behaviour
   HUBBLE [LAMBDA, MU, NU]
where

process HUBBLE [LAMBDA, MU, NU] : noexit :=
      hide FAIL in
         (
            (
               GYRO [LAMBDA, FAIL] (true of Bool)
               |||
               GYRO [LAMBDA, FAIL] (true of Bool)
               |||
               GYRO [LAMBDA, FAIL] (true of Bool)
               |||
               GYRO [LAMBDA, FAIL] (true of Bool)
               |||
               GYRO [LAMBDA, FAIL] (true of Bool)
               |||
               GYRO [LAMBDA, FAIL] (true of Bool)
            )
            |[FAIL]|
            CONTROLLER [FAIL, MU, NU] (6 of Nat, false of Bool)
            >>
            (* system reset *)
            HUBBLE [LAMBDA, MU, NU]
         )
```



```
"NU" -> "repair; rate 6",
"MU" -> "suspend; rate 100",
"LAMBDA" -> "fail; rate 0.1“
```

…a good canditate for the SVL scripting language

Conclusion

- Broaden the CADP toolkit to performance and dependability modelling and analysis.

- Pragmatic approach
  - no syntax extension;
  - uses various parts of the toolset and SVL scripting;
  - partial algorithms to construct MC;
  - MC analysis algorithms.

- Part of forthcoming CADP 2003

  http://www.inrialpes.fr/vasy/cadp

- Future work
  - MC Model Checking
  - direct IMC analysis