Combining functional verification and performance evaluation using CADP

Hubert Garavel
INRIA / VASY (Grenoble, France)

joint work with
Holger Hermanns (Universität des Saarlandes)
and
Damien Bergamini, Moëz Cherif, Christophe Joubert (INRIA / VASY)

Motivations

Functional verification

- Properties characterize correct behaviours:
  - Does the system function properly?
  - Is the system safe? *(safety properties)*
  - Can the system progress? *(liveness properties)*

- Well-known verification techniques:
  - Model checking
  - Equivalence checking
  - Visual checking

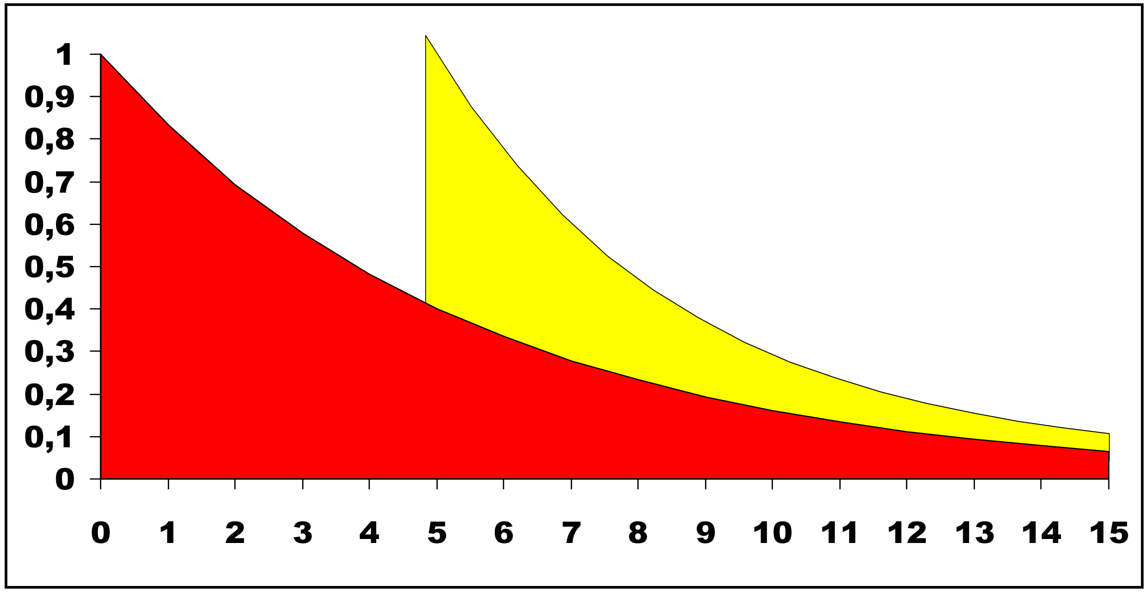
Performance evaluation

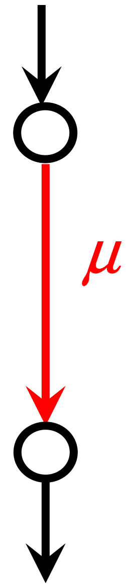
- Functional verification does not answer to all questions
- It does not answer to *quantitative questions* such as:
  - Is the system efficient? *(performance estimation)*
  - Which probability for a failure? *(dependability)*

- Well-studied evaluation techniques, e.g.:
  - Discrete-Time Markov Chains (DTMC): probabilistic
  - Continuous-Time Markov Chains (CTMC): stochastic

Continuous Time Markov Chains (CTMCs)

- All times are *exponentially distributed*
- Sojourn time in states are *memory-less*





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

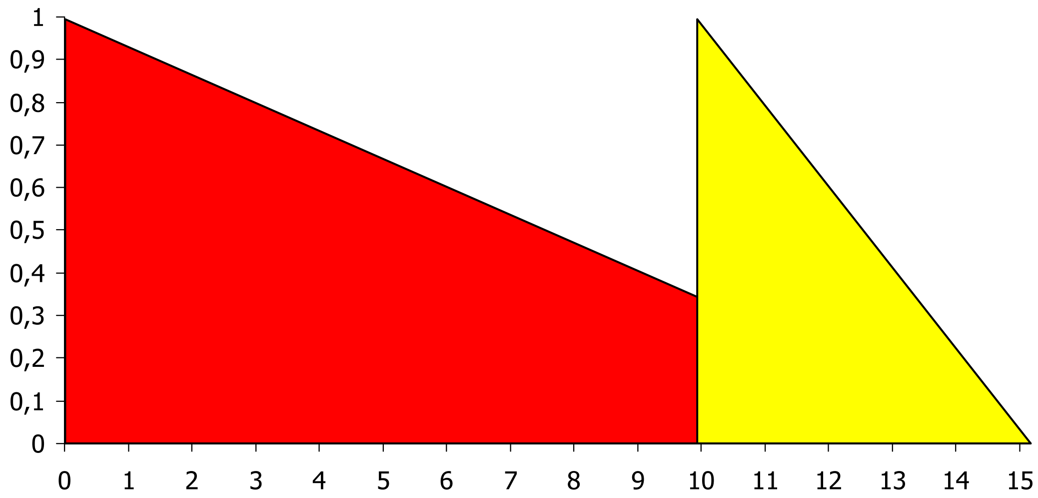
μ : *rate* (inverse of mean duration)

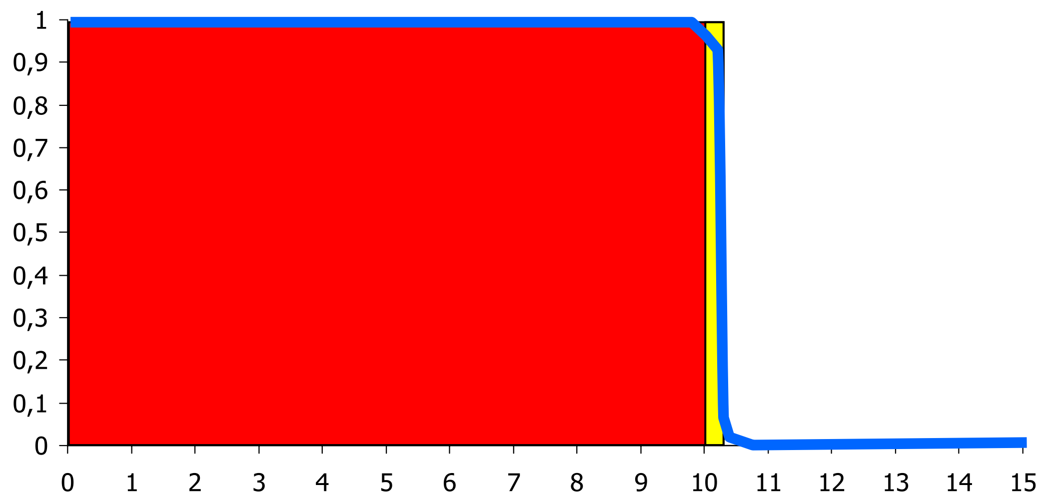
Advantages of CTMCs

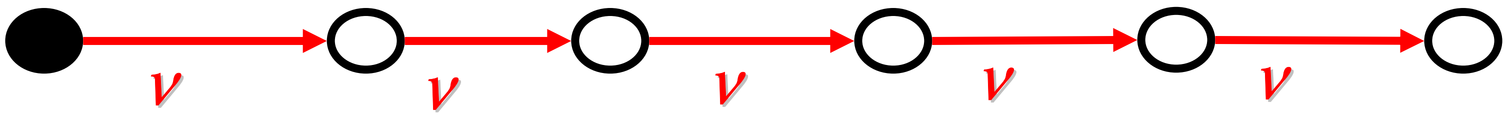
- Well known class of stochastic processes
- Widely used in practice
- Best guess, if only mean values are known
- Efficient, numerically stable algorithms for stationary and transient analysis

Isn't it too restrictive?

- Absence of memory is rare!
- But superpositions of exponential phases can approximate *arbitrary distributions*, still within the CTMC framework







Yet, a more general model is needed

- Main limitation of CTMCs: transitions carry no other information than rates (i.e., real numbers)

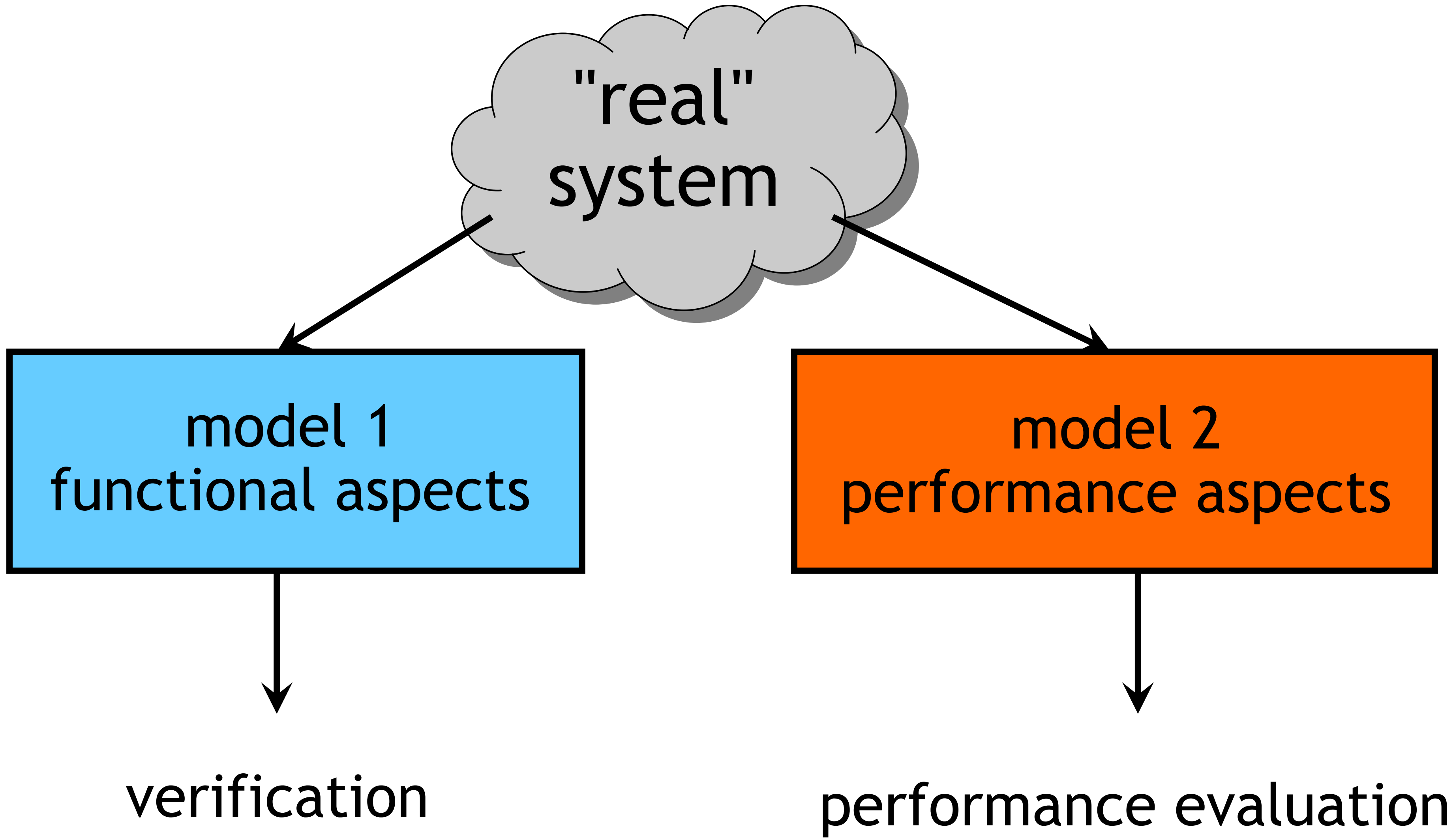
- This is sufficient for performance evaluation

- But this is not enough for
  - compositional modelling
  - functional verification

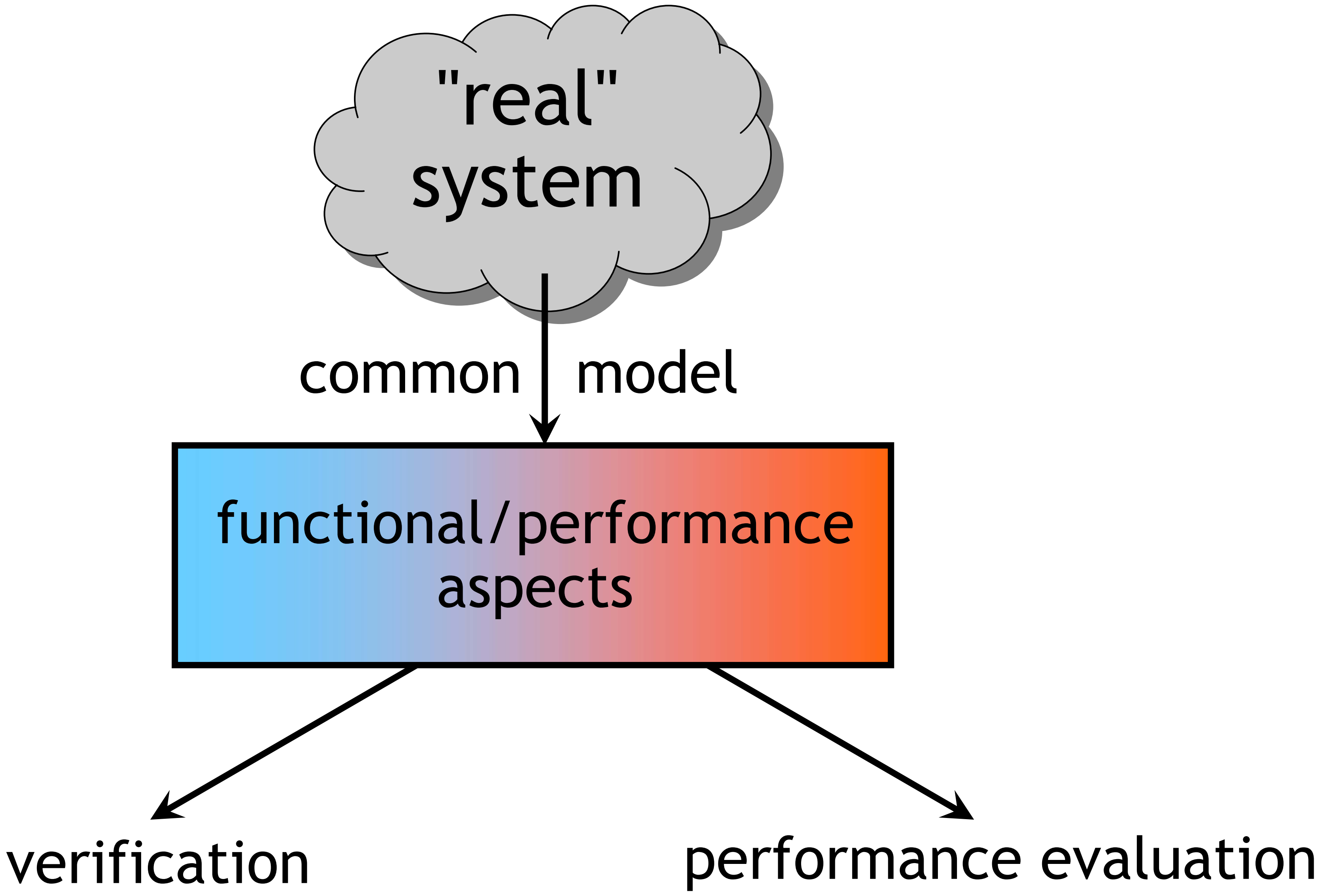
Why combining functional verification and performance evaluation?

- Reason #1: Scientific challenge
  - both fields are related
  - similar models: state machines, Markov chains
  - similar description languages:
    - (stochastic) Petri Nets
    - (stochastic) process algebras
  - same issues: state explosion, compositionality
- Reason #2: Economy

Current situation



Better situation



This would require...

1. A common modelling language

   => process algebras (>> Petri Nets)
   => LOTOS (international standard)

2. A common semantical framework

   => Interactive Markov Chains (IMCs)

3. Efficient software tools

   => CADP toolbox

LOTOS (ISO standard 8807)

- *Language Of Temporal Ordering Specification*
- A formal modelling language for asynchronous systems

- Two orthogonal sub-languages:

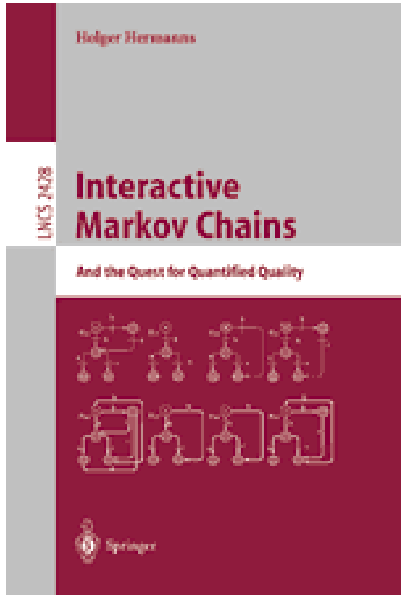
  Data part: abstract data types (ActOne)
  - Constructors and non-constructor operations
  - Equations and pattern-matching

  Behaviour part: process algebra (CCS, CSP, Circal)
  - Concurrent processes (interleaving semantics)
  - Message-passing communication (rendezvous)

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

Interactive Markov Chains (IMCs)

- Defined in H. Hermanns' PhD thesis (LNCS 2428)
- It adds stochastic features to process algebra, still providing:
  - sufficient stochastic expressivity
  - compatibility with process algebra theory
  - useful compositionality results



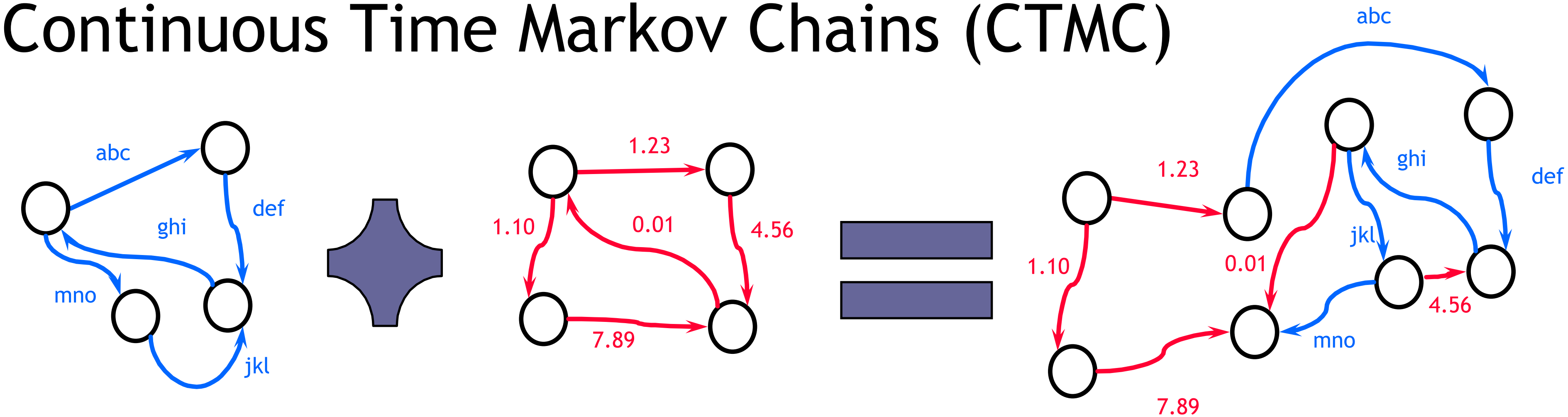
It is not only Hermanns' answer, but really `the' answer

E. Brinksma, U. Herzog

Interactive Markov Chains

An orthogonal extension of

- Labelled Transition Systems (LTS)
- Continuous Time Markov Chains (CTMC)



LTS

CTMC

IMC

labels = typed data
(messages exchanged)

labels = real numbers
λ, μ, ν

both types of labels

Remainder of the talk

- *Motivations*
- Tool support for IMCs within CADP
- CADP tools for generating Markov models
- CADP tools for reducing Markov models
- CADP tools for solving Markov models
- Application 1: the Hubble space telescope
- Application 2: the SCSI-2 bus arbiter
- Conclusion

Tool support for IMCs within CADP

What is CADP?

- *"One of the leading verification toolboxes in academia"*

  H. Hermanns

- *"Among the most popular non-US originating verification tools"*

  R. Cleaveland, D. Pilaud, B. Steffen (May 2003)

- A few figures:
  - license agreement signed by >300 institutions
  - since Jan 1st 2003: CADP installed on > 730 machines
  - 72 published case-studies with CADP
  - 13 software tools connected to CADP

The CADP toolbox

- LOTOS compilers (Caesar and Caesar.adt)
- Simulation, rapid prototyping, test generation, etc.
- Explicit state verification:
  - equivalence checking (bisimulation)
  - model checking (modal mu-calculus)
  - visual checking
- Generic software components (BCG, Open/Caesar)
- Advanced verification techniques:
  - on the fly (boolean equation systems)
  - compositional
  - massively parallel
- Graphical user interface + scripting language (SVL)

A pragmatic approach for IMCs

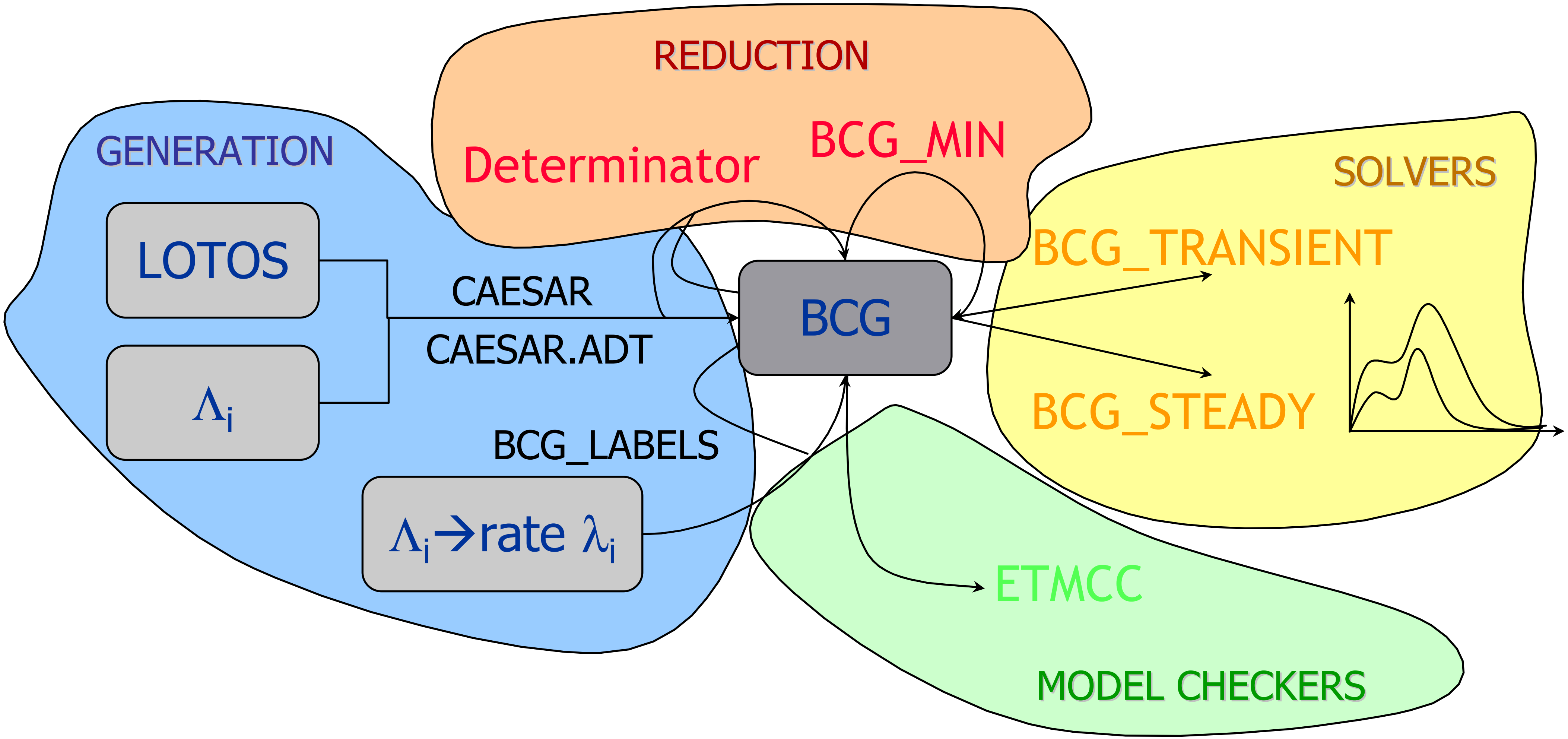
- *Reuse existing CADP tools as much as possible*
- Decision 1: reuse the LOTOS compilers without modification
- Decision 2: reuse the BCG format of CADP as the unique format for
  - Labelled Transition Systems
  - Discrete Time Markov Chains
  - Continuous Time Markov Chains
  - Interactive Markov Chains
  - *mixed models*

Markov models in BCG

- 5 possible types of transition labels
  - ordinary "SEND !21 !true"
  - probabilistic "prob 0.82"
  - stochastic "rate 3.14"
  - mixed probabilistic "SEND !21 !true ; prob 0.82"
  - mixed stochastic "SEND !21 !true ; rate 3.14"

- *also: timed labels (a different story)*
  - *timed:* "wait 10.2"
  - *mixed timed:* "SEND !21 !true ; wait 10.2"

CADP tools for performance evaluation



CADP tools for generating Markov models

(Interactive) Markov Chains in LOTOS

- How to generate an extended BCG from LOTOS?
  ⇔
  How to introduce rates in LOTOS descriptions?

- Two complementary approaches:
  - *Direct insertion*
  - *Compositional insertion*

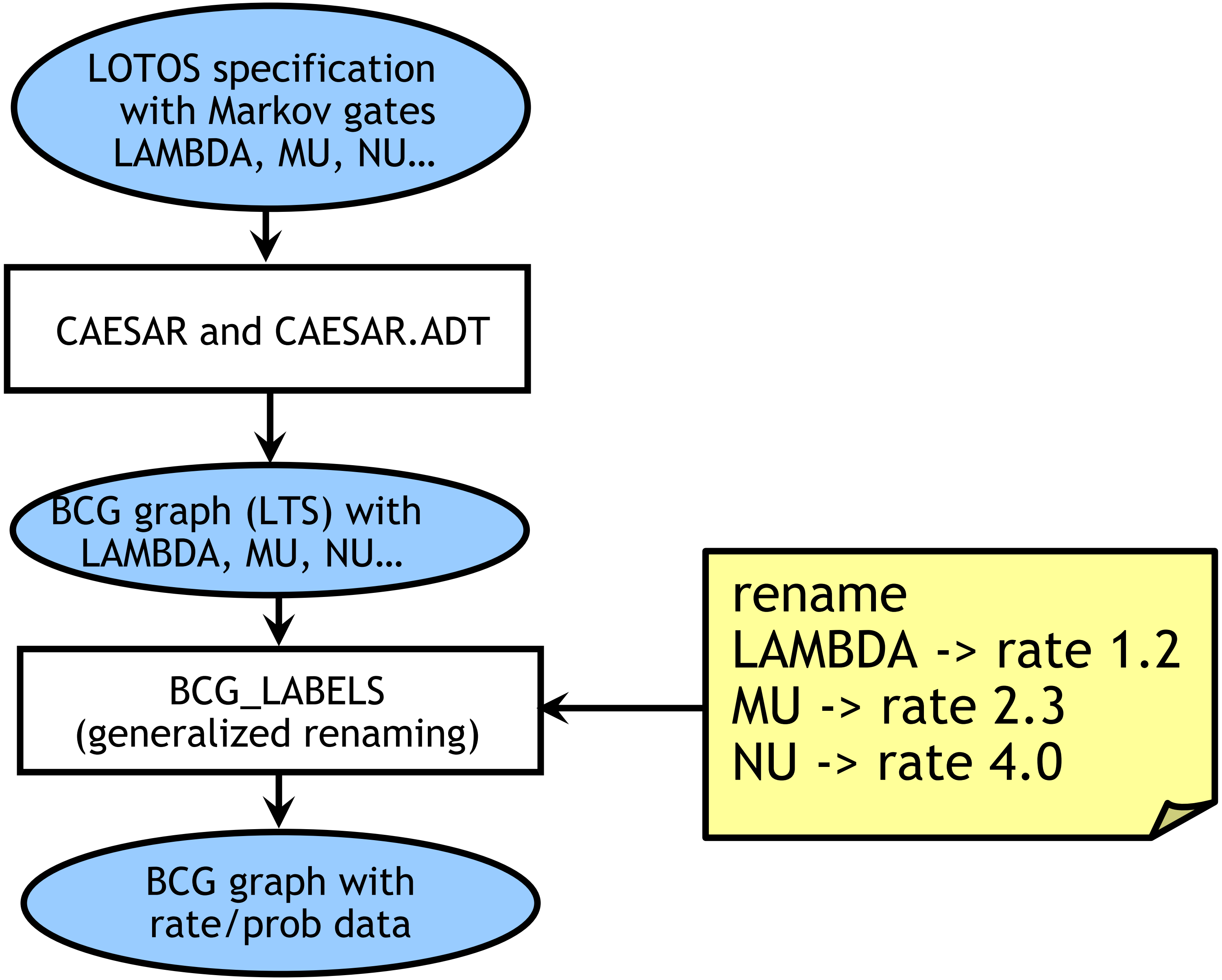
Direct insertion

- Two types of ‘actions’ (gates) in the specification
  - standard actions: SEND, RECV…
  - Markov gates: LAMBDA, MU, NU…
- Insert Markov gates in the LOTOS specification where Markov delays occur
- User-defined (and user-maintained) separation between both types of gates
- No synchronization allowed on Markov gates

Inserting Markov gates

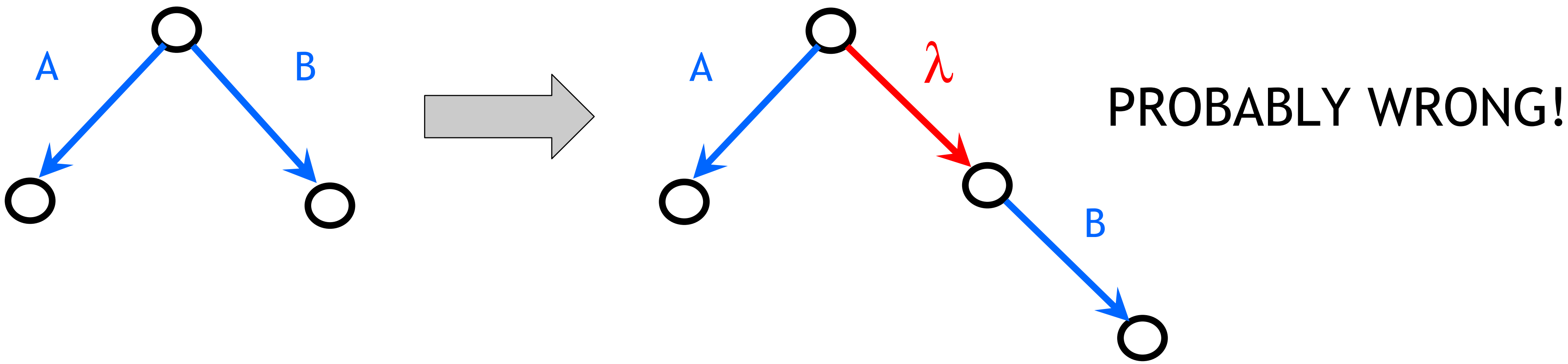
```
process DISK [ARB, CMD, REC, MU] (N:NUM, L:NAT, READY:BOOL):noexit :=
   CMD !N;
      DISK [ARB, CMD, REC, MU] (N, L+1, READY)
   []
   ARB ?W:WIRE [not (READY) and C_PASS (W, N)];
      DISK [ARB, CMD, REC, MU] (N, L, READY)
   []
   [not (READY) and (L > 0)] ->
      MU !N; (* Markov delay inserted 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
```

Direct insertion: Tool trajectory



Direct insertion: A potential risk

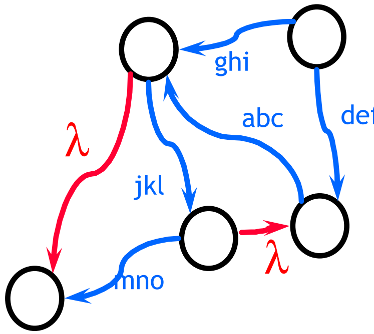
- Inserting Markov delays requires knowledge (at least, educated guesses) about:
  - the *right place* to introduce Markov delays
  - their numerical values
- Introducing Markov gates may corrupt the original functional behaviour!



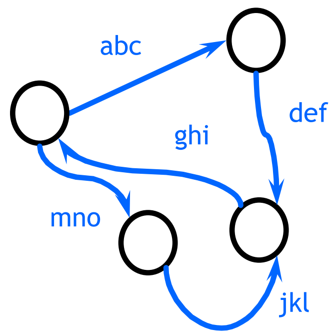
Direct insertion: Proof obligation

- either show that modified specification is branching equivalent to original one, if Markov delays are considered as internal (τ) steps

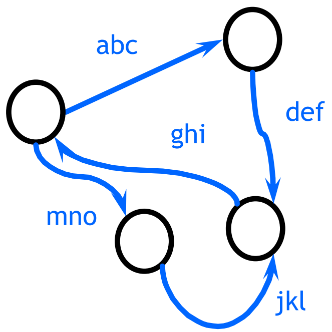
hide λ in (



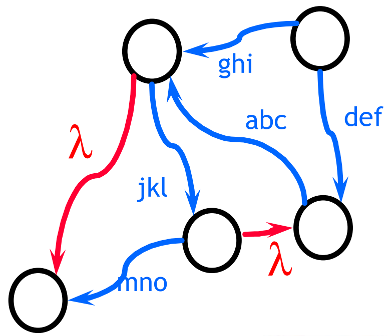
) $\sim_b$



- or repeat model checking on both specifications



and

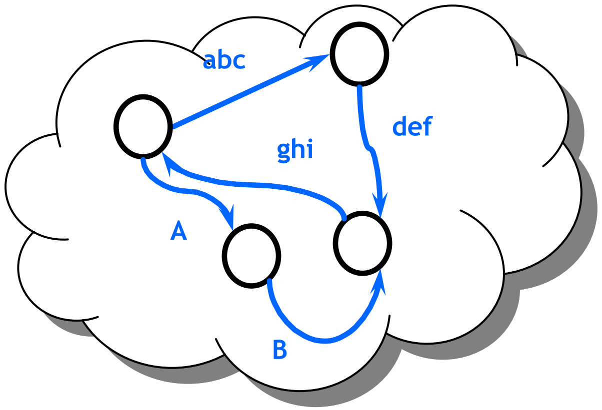


satisfy the same formulas

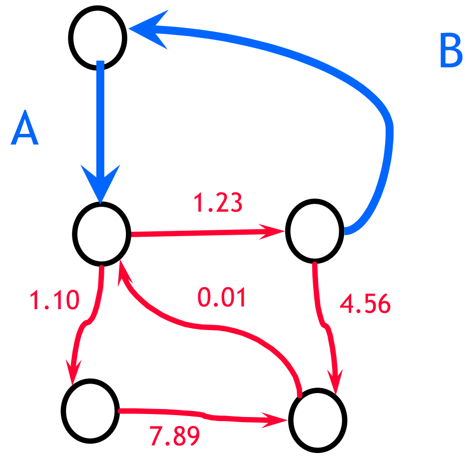
Compositional insertion (1)

- Alternative approach to direct insertion
- Identify visible actions that
  - are to be delayed, or
  - initialize a delay, or
  - may interrupt a delay
- Use the LOTOS 'constraint-oriented' style to insert Markov delays between these actions

Compositional insertion (2)



|[A, B]|



- No proof obligation is needed (proven by Holger Hermanns)
- Another example later (SCSI-2 bus arbiter)

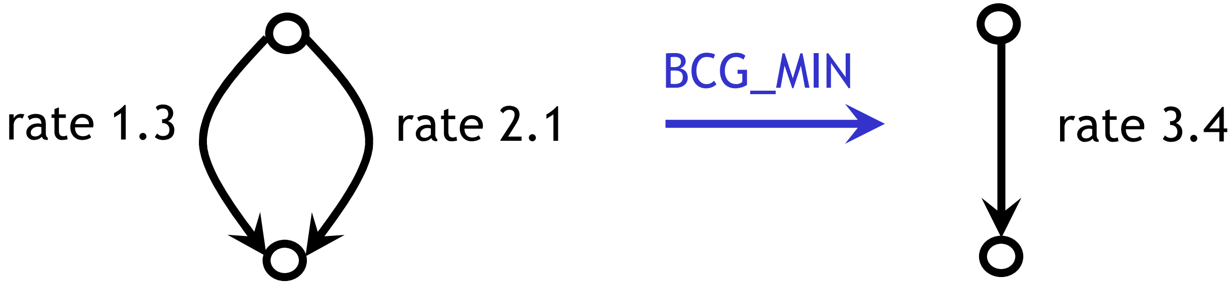
Compositional insertion (3)

- An important constraint must be enforced: synchronization is not allowed on Markov rates
- Why?
- LOTOS semantics :
  "rate λ" || "rate λ" = "rate λ"
- Markov semantics :
  "rate λ" || "rate λ" = "rate $2*\lambda$"
- Solution: use different gate names (LAMBDA, MU, NU...) to avoid unwanted synchronizations

CADP tools for reducing Markov models

The BCG_MIN tool

- BCG_MIN: an efficient (property preserving) minimization tool
- Inputs:
  - BCG graph and its type (LTS, IMC, DTMC, CTMC…)
  - chosen equivalence for minimization
- Output:
  - minimized BCG graph
- For standard LTS:
  - implements strong bisimulation and branching bisimulation
  - truly better than Aldebaran and fc2min
  - up to 8 million states, 43 million transitions
- For probabilistic/stochastic LTS:
  - implement strong and `branching' bisimulation minimization
  - lumpability
  - might translate an IMC into a MC by removing (some) nondeterminism



The DETERMINATOR tool

- Role:
  - On-the-fly generation of a MC starting from either a high level description (e.g., LOTOS) or a low level model (BCG graph)
  - applies local transformations to remove nondeterminism partially
  - implements a determinacy check ("well specified" stochastic process)
  - algorithm from [Ciardo-Zijal-96] [Deavours-Sanders-99]
- Input:
  - on the fly graph (IMC, DTMC, CTMC…)
  - based on CADP's Open/Caesar language-independent technology
- Output:
  - BCG graph (possibly, same as input graph)

- DETERMINATOR before BCG_MIN => significant time savings

CADP tools for solving Markov models

The BCG_STEADY tool

- Numerical solver for Markov chains
- Steady state analysis (equilibrium)
- Inputs:
  - BCG graph with "*action*; **rate** *r*" labels
  - no deadlock allowed
- Outputs:
  - numerical data usable by Excel, Gnuplot...
- Method:
  - BCG graph converted into a sparse matrix
  - computation of a probabilistic vector solution
  - iterative algorithm using *Gauss-Seidel* [Stewart94]

$$\pi_i^{(k+1)} = -\frac{1}{a_{i,i}} \left(\sum_{j<i} \pi_j^{(k+1)} a_{i,j} + \sum_{j>i} \pi_j^{(k)} a_{i,j} \right)$$

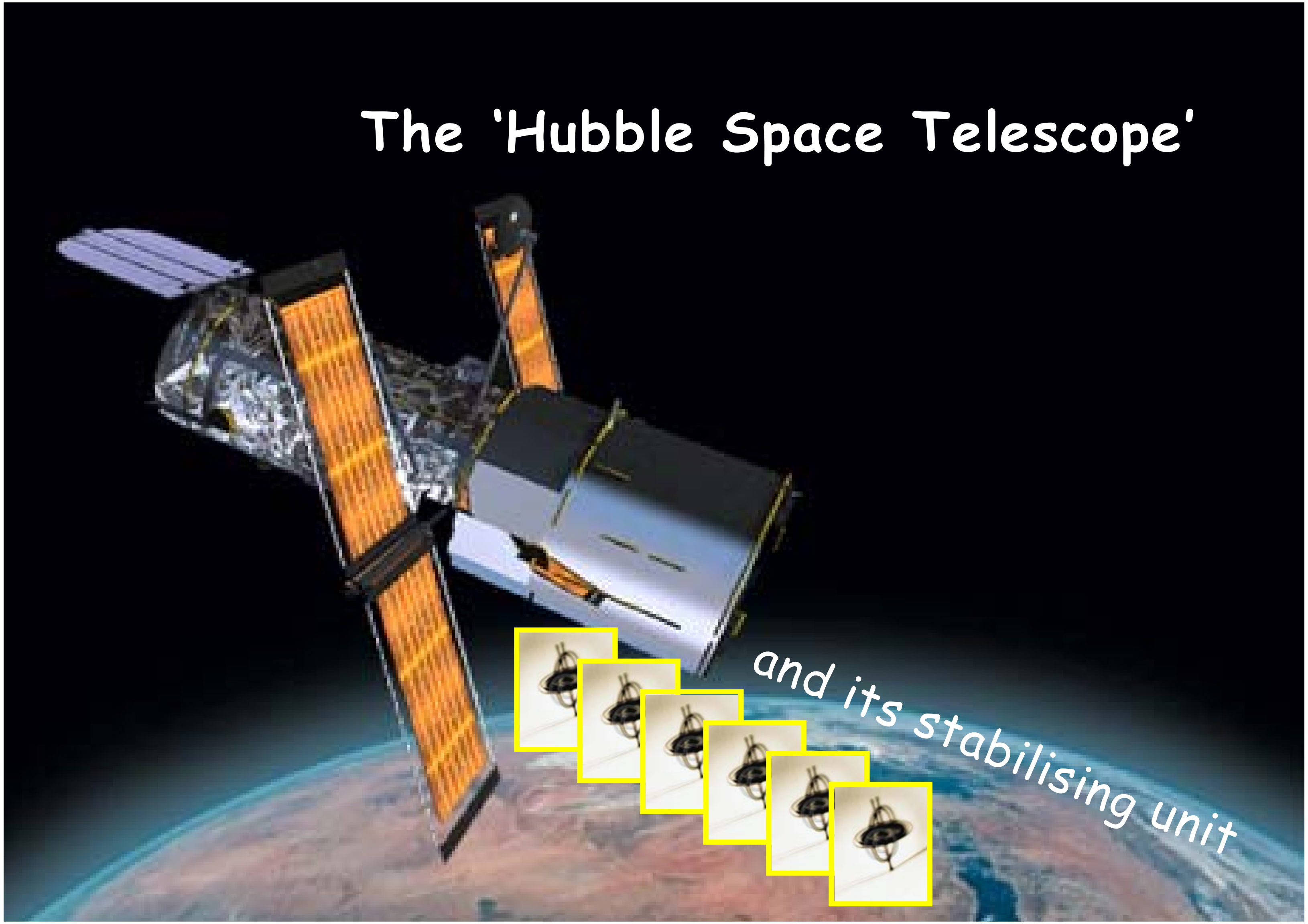
The BCG_TRANSIENT tool

- Numerical solver for Markov chains
- Transient analysis
- Inputs:
  - BCG graph with "*action*; **rate *r***" labels
  - deadlocks permitted
  - list of time instants
- Outputs:
  - numerical data usable by Excel, Gnuplot...
- Method:
  - BCG graph converted into a sparse matrix
  - uniformisation method to compute Poisson probabilities
  - Fox-Glynn algorithm [Stewart94]

$$\underline{\tilde{\pi}}(t) = \sum_{n=0}^{k_{ss}} \psi(\lambda t; n)\underline{\hat{\pi}}(n) + \left(\sum_{n=k_{ss}+1}^{k_{\varepsilon}} \psi(\lambda t; n) \right) \underline{\hat{\pi}}(k_{ss}) \quad with \quad \psi(\lambda t; 0) = e^{-\lambda t}$$

$$and \quad \psi(\lambda t; n+1) = \psi(\lambda t; n) \frac{\lambda t}{n+1}, n \in \mathrm{N}$$

Application 1: The Hubble telescope


The 'Hubble Space Telescope'
and its stabilising unit

A simple Markov model for the Hubble

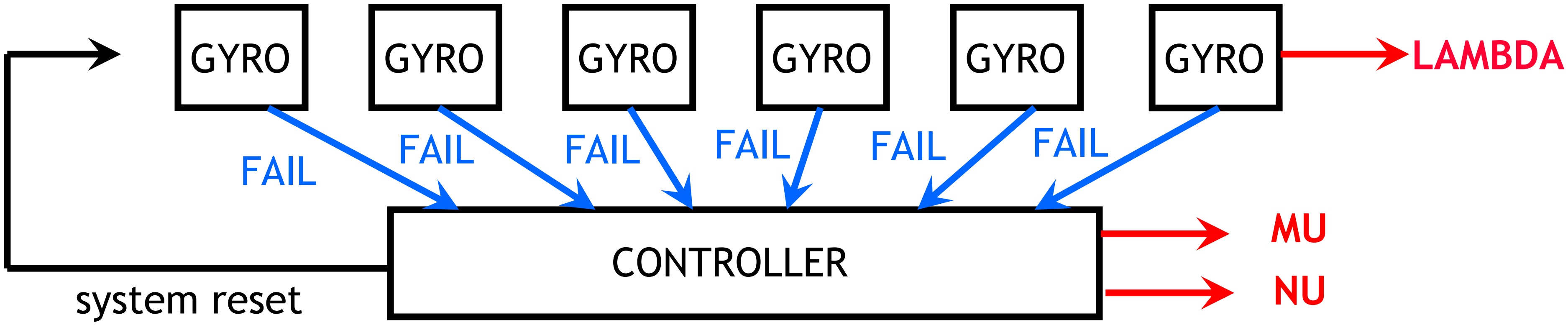
- The Huble telescope has 6 gyroscopes
- As time passes, gyros may fail
- The average lifetime of gyros is 10 years (= 120 months)

  $\lambda = 12 \text{ months} / 120 = 0.1$
- Hubble falls into sleep if only two gyros are left
- Turning on sleep mode requires to halt all equipments, which takes about 3.6 days (= 0.12 month)

  $\mu = 12 \text{ months} / 0.12 = 100$
- When in sleep mode, a shuttle mission must be sent to repair/reset Hubble, which takes about 2 months

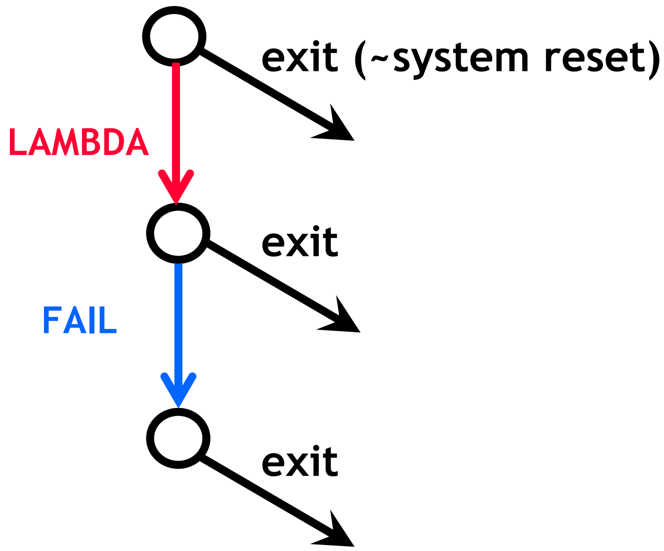
  $\nu = 12 \text{ months} / 2 = 6$
- Without operational gyro, Hubble crashes

Compositional modelling of the Hubble




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

The GYRO process

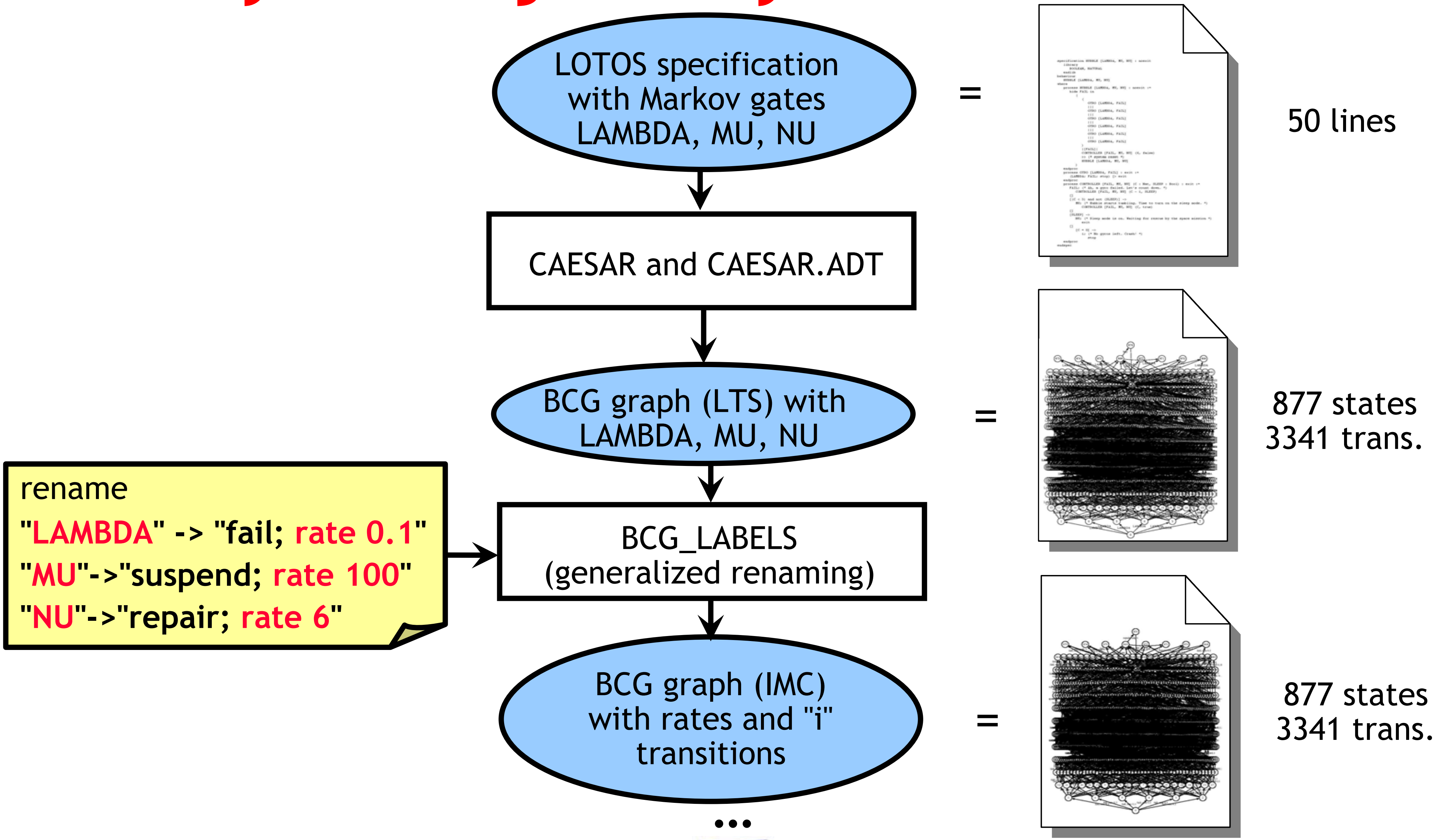



```
process GYRO [LAMBDA, FAIL] : exit :=
  (LAMBDA; FAIL; stop) [> exit
endproc
```

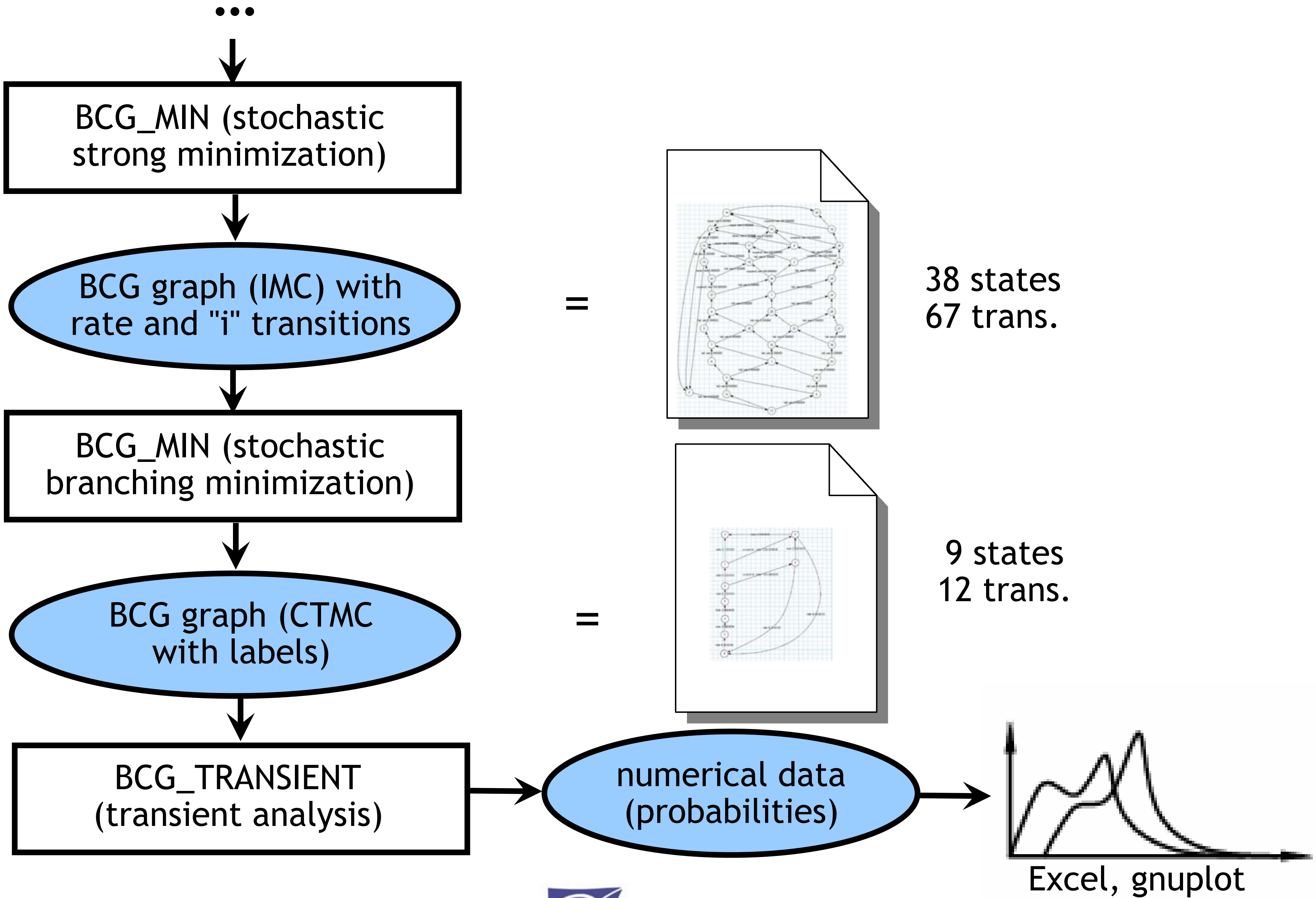
The CONTROLLER process

```
process CONTROLLER [FAIL, MU, NU] (C : Nat, SLEEP : Bool) : exit :=
     FAIL; (* Ah, a gyro failed. Let's count down. *)
           CONTROLLER [FAIL, MU, NU] (C - 1, SLEEP)
     []
     [(C < 3) and not (SLEEP)] ->
          MU; (* Hubble starts tumbling. Time to turn on the sleep mode. *)
                CONTROLLER [FAIL, MU, NU] (C, true)
     []
     [SLEEP] ->
          NU; (* Sleep mode is on. Waiting for the space mission to reset Hubble. *)
                exit
     []
     [C = 0] ->
           i; (* No gyros left. Crash! *)
                stop
endproc
```

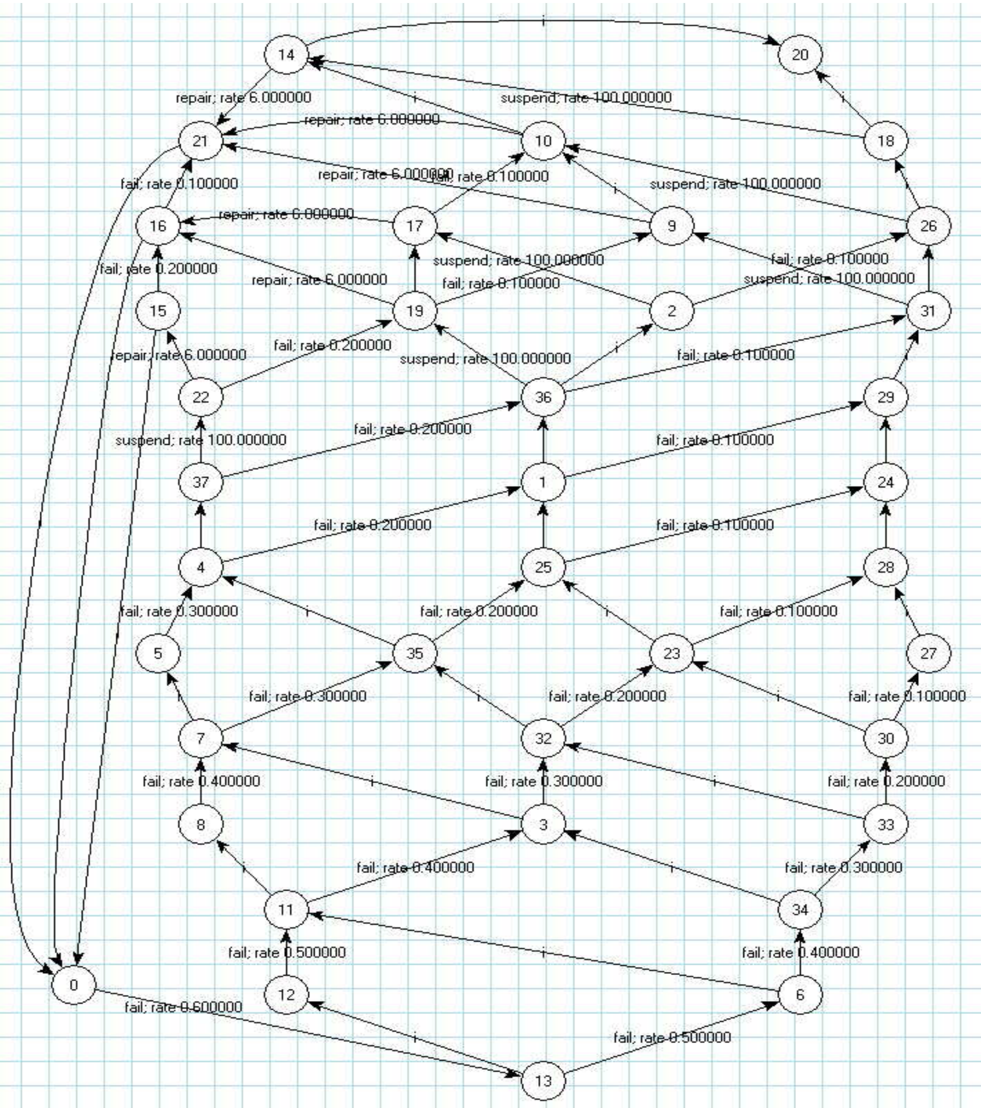
Analysis trajectory for the Hubble



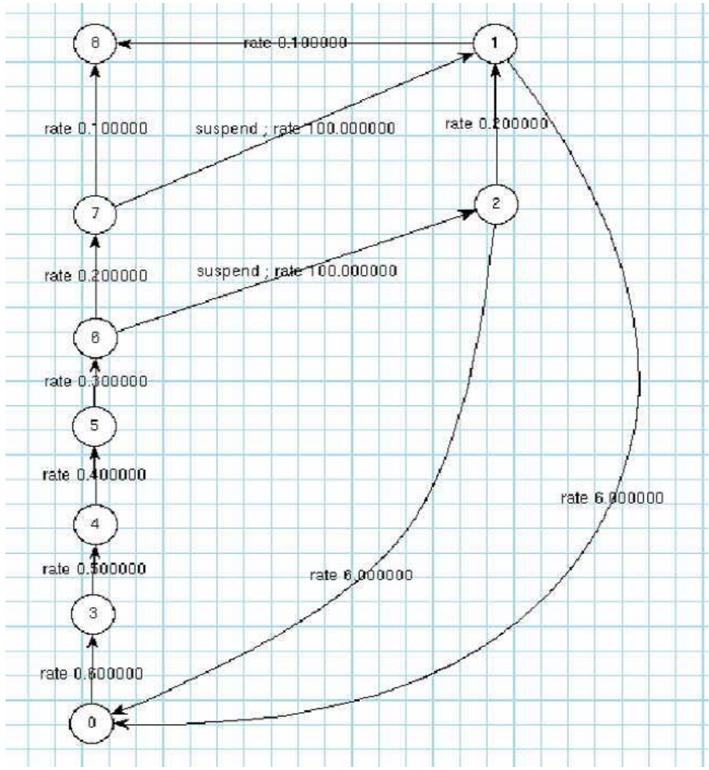
… Analysis trajectory for the Hubble



Minimized IMCs for the Hubble

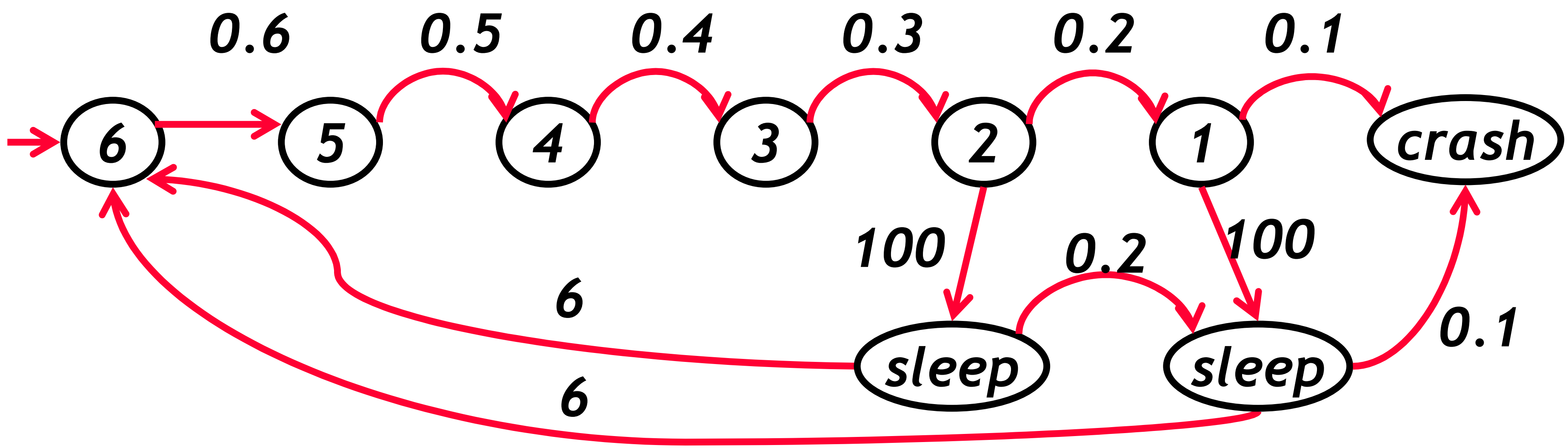


after stochastic **strong** minimization
(38 states, 67 transitions)



after stochastic **branching** minimization
(9 states, 12 transitions)

Visual verification of the final CTMC

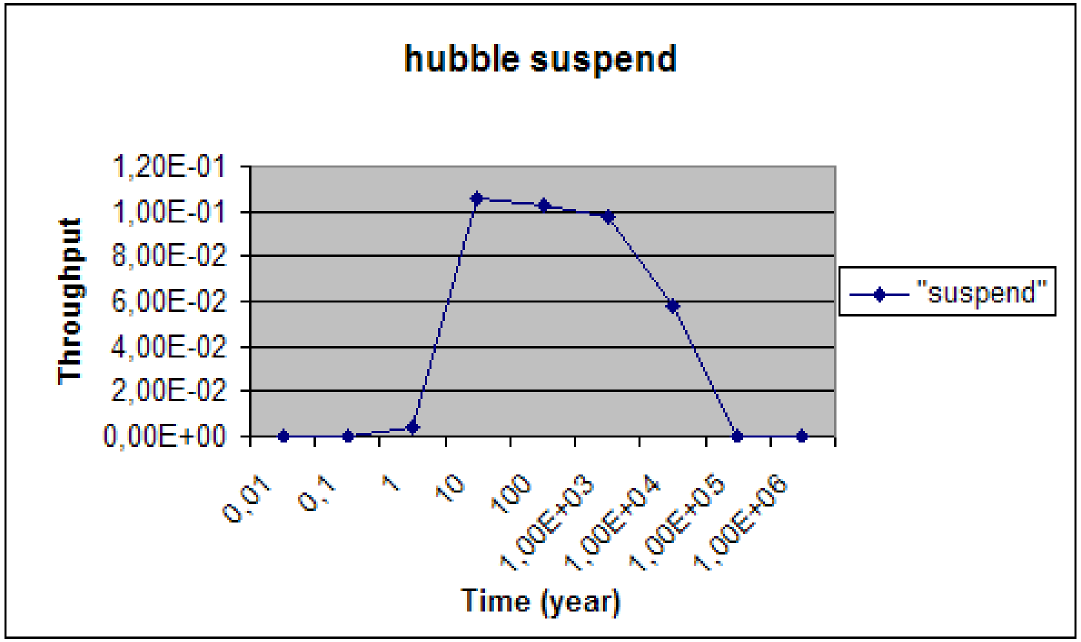


SVL script for the Hubble

```
(* generate the LTS *)
"lts.bcg" = generation of "hubble.lotos";

(* turn the LTS into an IMC *)
"imc.bcg" =  total rename
      "NU" -> "repair; rate 6",  (* to prepare a shuttle mission, for reset takes 1/2 a year *)
      "MU" -> "suspend; rate 100",  (* to suspend the scientific, progtam takes 1/100 of a year *)
      "LAMBDA" -> "fail; rate 0.1"  (* the average lifetime of a gyroscope is 10 years *)
in "lts.bcg";

(* turn the IMC into an CTMC *)
"ctmc.bcg" = branching stochastic reduction with bcg_min of "imc.bcg";

(* look for internal transitions: if absent, "ctmc.bcg" is a Markov chain *)
% bcg_info -hidden "ctmc.bcg"

(* analyse for various time points measured in years *)
% bcg_transient -thr hubble.thr  "ctmc.bcg" .01 .1 1 10 100 1e3 1e4 1e5 1e6
```

Analysis of the Hubble using BCG_TRANSIENT

time	"repair"	"fail"	"suspend"
0.01	1.52E-11	0.5994	1.24E-09
0.1	5.45E-07	0.59403	4.34E-06
1	0.00248872	0.543138	0.00373419
10	0.105761	0.414947	0.105725
100	0.102729	0.414615	0.102786
1.00E+03	0.0974923	0.393478	0.097546
1.00E+04	0.0577739	0.233175	0.0578058
1.00E+05	0.00031195	0.00125902	0.00031212
1.00E+06	6.03E-27	2.43E-26	6.04E-27



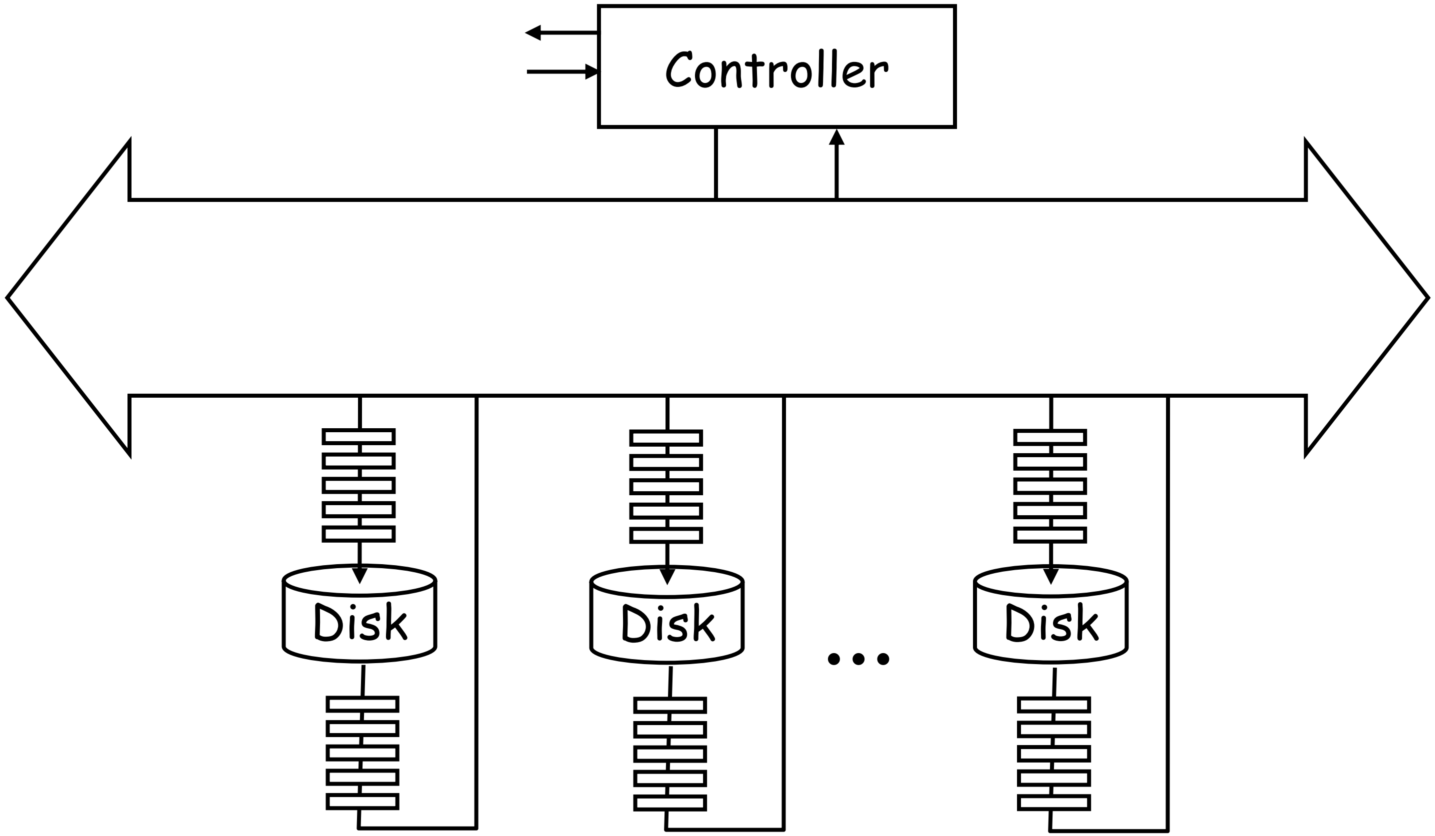
Application 2: The SCSI-2 bus arbiter

Case study

SCSI-2: Small Computer System Interface

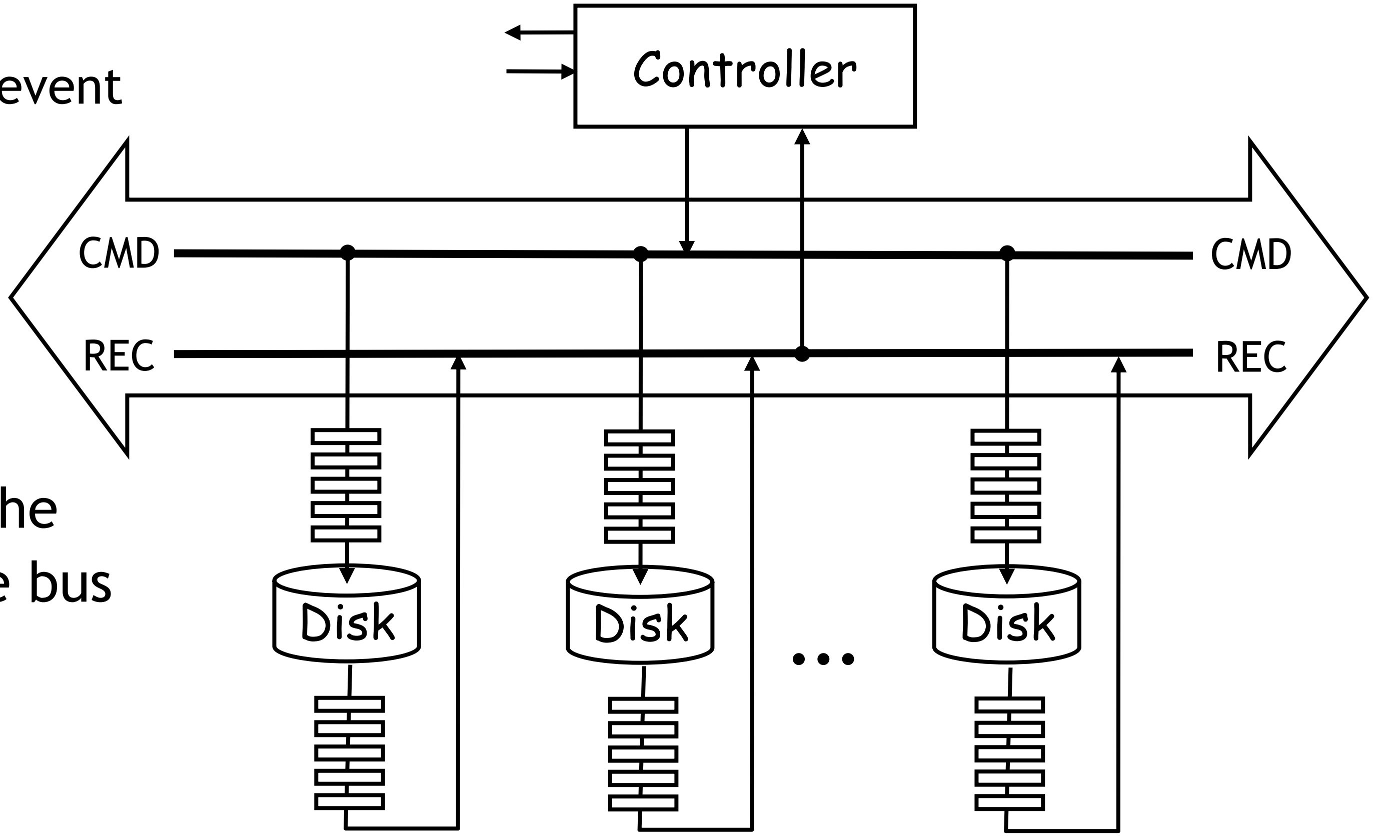
- brought to our attention by Massimo Zendri (Bull SA, Italy)
- designed to provide fast access to multiple storage devices, via a shared bus
- up to 7 devices (disks) and 1 controller
- under study: SCSI-2 bus arbitration protocol
- ‘starvation problem’ discovered by Bull engineers

The SCSI-2 architecture



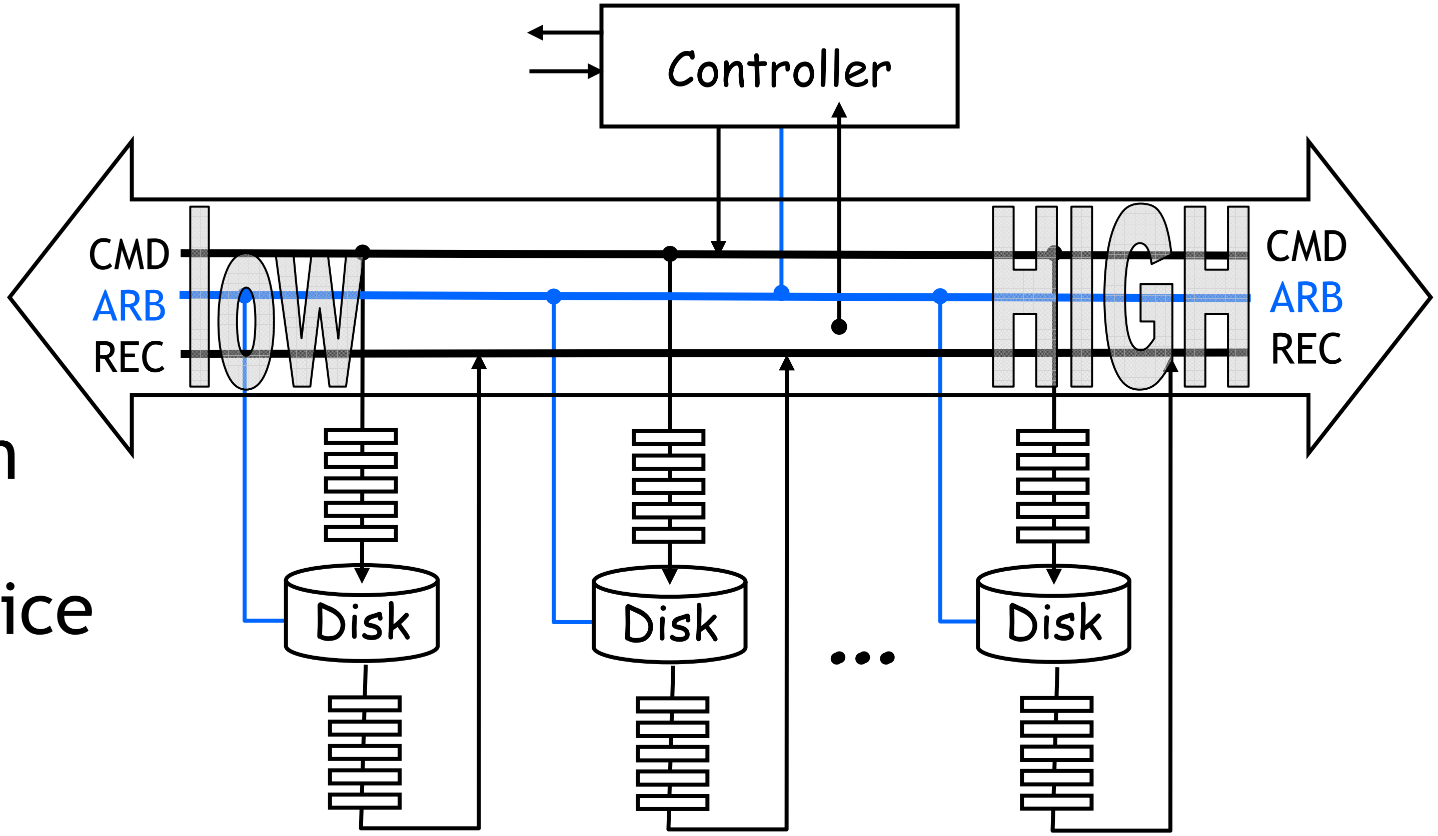
SCSI-2 bus usage

- Controller
  - handles (OS level) requests
  - passes read/write requests to the designated disk (CMD)
  - passes results back to the OS (REC)
  - provides flow control to prevent disk flooding,
- Disks
  - process incoming CMDs,
  - send back results by REC,
- Disks and Controller share the bus, but mutually exclusive bus access is granted by a distributed bus arbitration mechanism.



SCSI 2 bus arbitration

- Prioritized, based on static IDs on bus
- Realized through a mesh of dedicated wires
- Any bus access is preceded by a scan ensuring that no higher priority device requires the bus



Starvation and how it was fixed

- The Bull engineers observed ‘starvation’ of applications for some specific configurations, depending on the position of the controller on the bus
- They observed that this problem was absent if the controller was in the highest position, and the OS was put on the lowest priority disk
- Model checking with CADP revealed the starvation problem and its cause: a livelock preventing lower priority disks to get the bus [Garavel & Mateescu]
- (Problem solved in SCSI-3 standard)

Specifying the SCSI-2 in LOTOS

- Capturing the SCSI-2 bus arbitration priority mechanism (distributed, virtually synchronous) is nontrivial

- Only process algebras with n-party rendezvous (LOTOS, CSP) can do it properly

- Languages with only binary communication => combinatorial explosion

Specifying the SCSI-2 in LOTOS

- Use of a key LOTOS feature: *value negotiation*
- W: a tuple of 8 booleans (wires)
- Each process *i* states its own constraints:
  - *C_PASS (i) : Wi = 0*
  - *C_WIN (i) : Wi = 1 and no j>i such that Wj = 1*
  - *C_LOSS (i) : Wi = 1 and exists j>i such that Wj = 1*
- Parallel composition of 8 processes => Intersection of the 8 corresponding constraints

(For details, see Garavel-Hermanns paper at FME'02)

Parallel composition of 7 disks and 1 controller

8-party rendezvous

```
(
DISK [ARB, CMD, REC, MU] (0, 0, false)
|[ARB]|
DISK [ARB, CMD, REC, MU] (1, 0, false)
|[ARB]|
   ...
|[ARB]|
DISK [ARB, CMD, REC, MU] (6, 0, false)
)
|[ARB, CMD, REC]|
CONTROLLER [ARB, CMD, REC, LAMBDA] (7, 7, ZERO)
```

The DISK process

```
process DISK [ARB, CMD, REC, MU] (N:NUM, L:NAT, READY:BOOL):noexit :=
   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
```

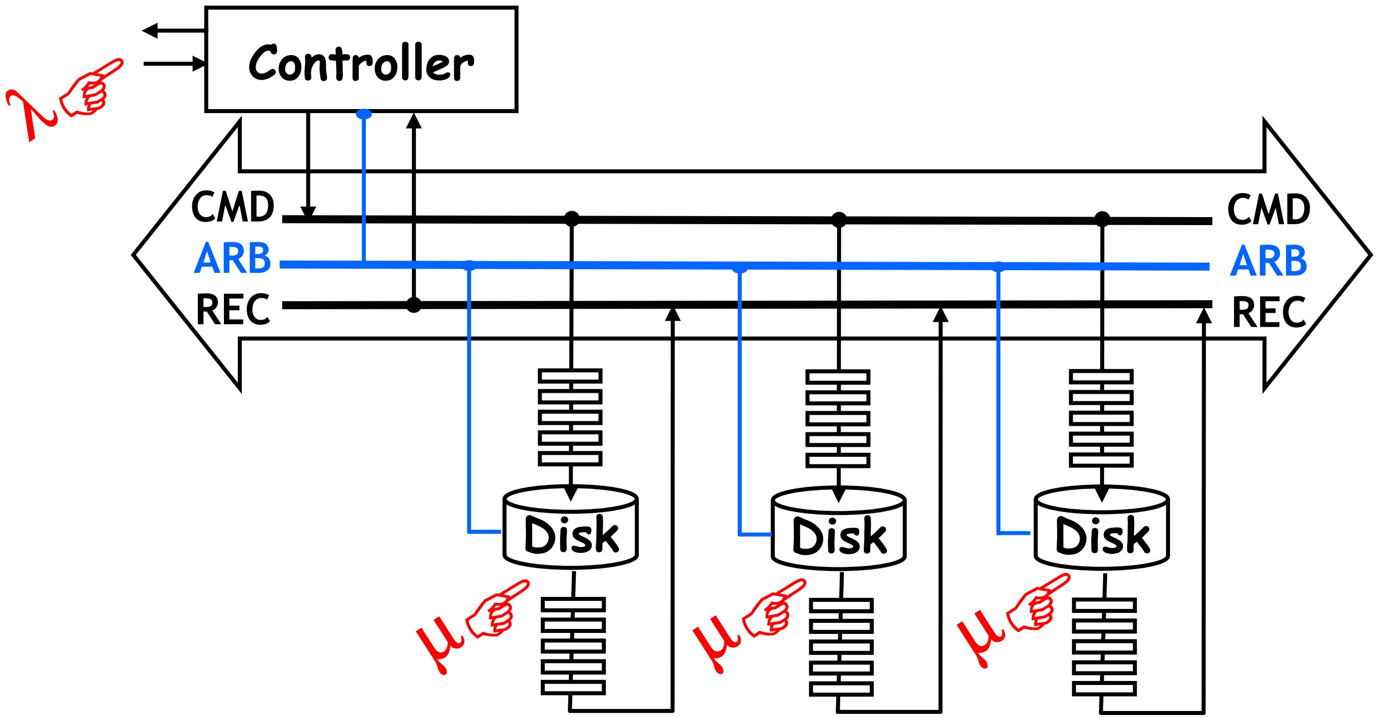
an 8-tuple of bits

'pattern matcher'

Direct insertion of Markov delays

Two Markov delays are inserted directly:

- λ: load (i.e., stress) of the controller
- μ: disk servicing time



The CONTROLLER process with a Markov delay

```
process CONTROLLER [ARB, CMD, REC, LAMBDA] (NC:NUM, PENDING:NUM,
                                            T:TABLE) : noexit :=
   ARB ?W:WIRE [(PENDING == NC) and C_PASS (W, NC)];
      CONTROLLER [ARB, CMD, REC, LAMBDA] (NC, PENDING, T)
   []
   (
   choice N:NUM []
      [(PENDING == NC) and (N <> NC)] ->
         [VAL (T, N) < 8] ->
            LAMBDA !N; (* Markov delay inserted here *)
               CONTROLLER [ARB, CMD, REC, LAMBDA] (NC, N, T)
   )
   []
   ARB ?W:WIRE [(PENDING <> NC) and C_LOSS (W, NC)];
      CONTROLLER [ARB, CMD, REC, LAMBDA] (NC, PENDING, T)
   []
   ARB ?W:WIRE [(PENDING <> NC) and C_WIN (W, NC)];
      CMD !PENDING;
         CONTROLLER [ARB, CMD, REC, LAMBDA] (NC, NC, INCR (T, PENDING))
   []
   REC ?N:NUM [N <> NC];
      CONTROLLER [ARB, CMD, REC, LAMBDA] (NC, PENDING, DECR (T, N))
endproc
```

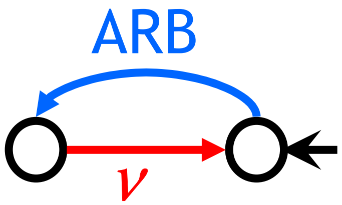
The DISK process with a Markov delay

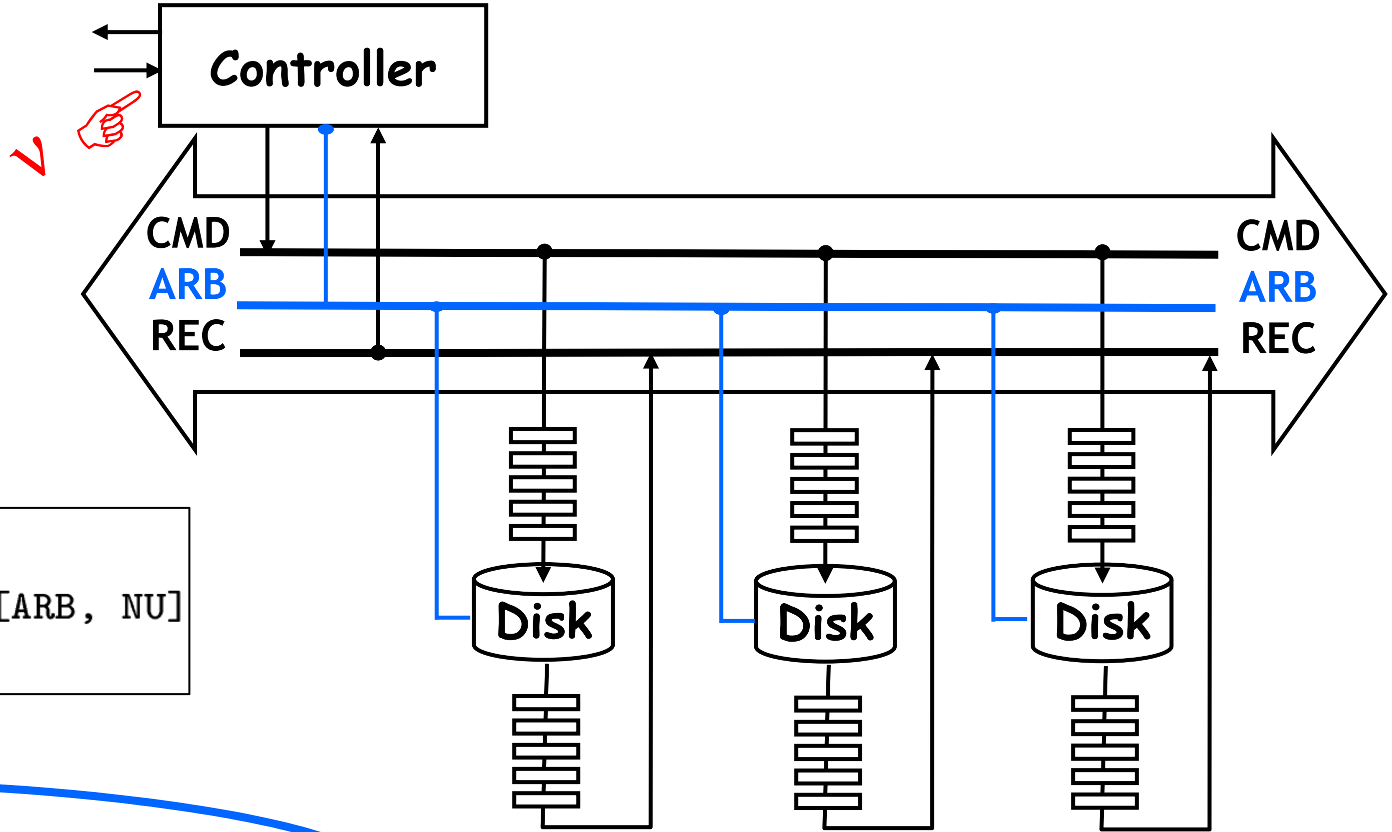
```
process DISK [ARB, CMD, REC, MU] (N:NUM, L:NAT, READY:BOOL):noexit :=
   CMD !N;
      DISK [ARB, CMD, REC, MU] (N, L+1, READY)
   []
   ARB ?W:WIRE [not (READY) and C_PASS (W, N)];
      DISK [ARB, CMD, REC, MU] (N, L, READY)
   []
   [not (READY) and (L > 0)] ->
      MU !N; (* Markov delay inserted 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
```

Compositional insertion of Markov delays

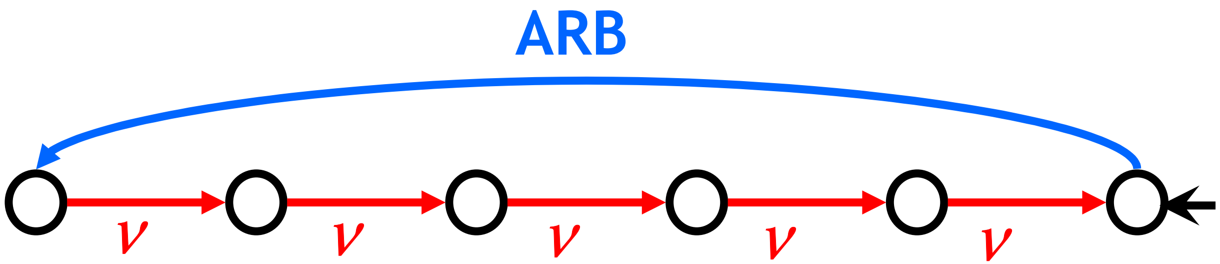
- *bus delay* ν : to be inserted between any two consecutive bus arbitrations ARB

```
process BUS [ARB, NU]:noexit :=
  ARB; NU; BUS [ARB, NU]
endproc
```





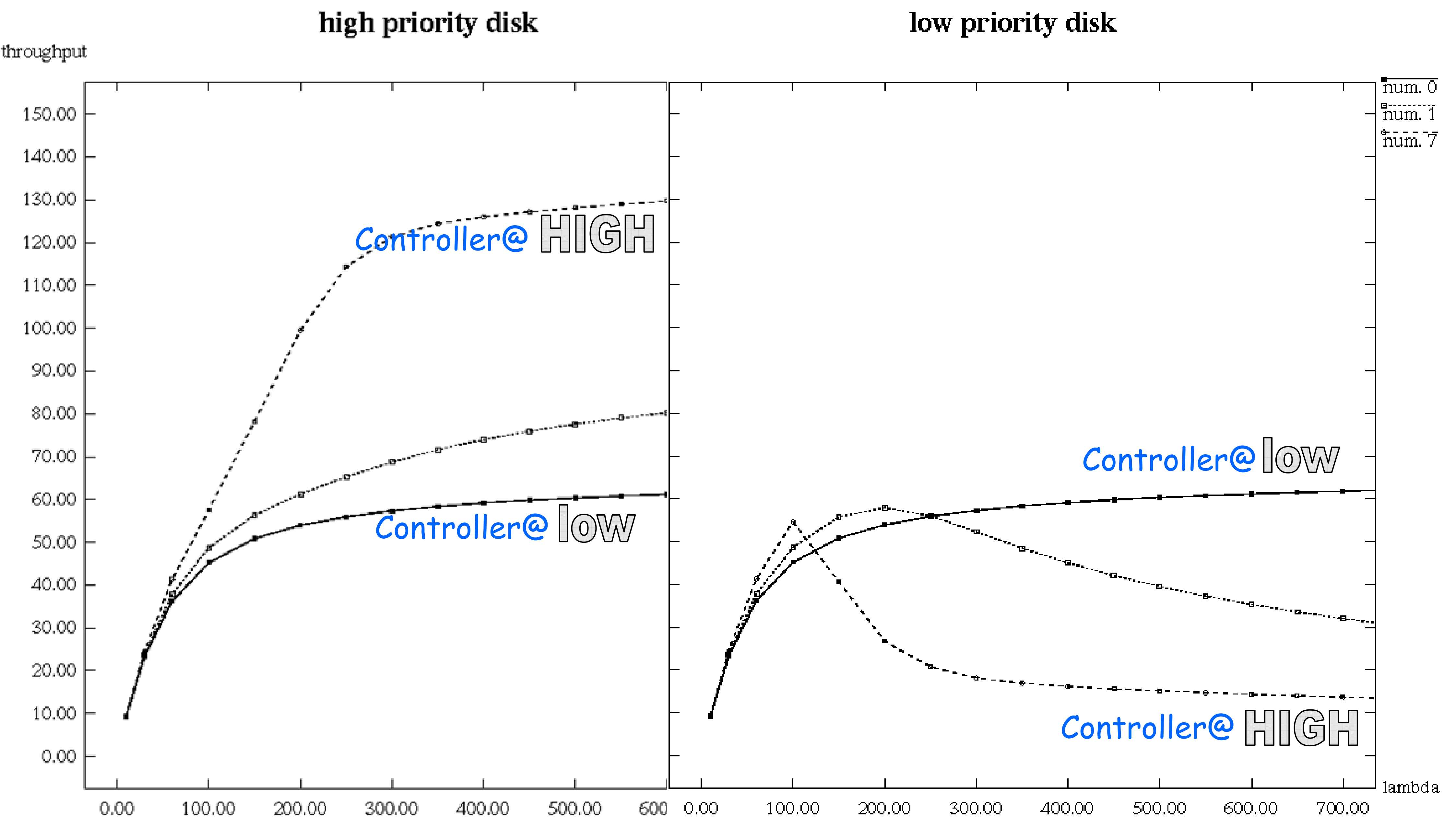

```
process BUS_5 [ARB, NU]:noexit :=
  ARB; NU; NU; NU; NU; NU; BUS_5 [ARB, NU]
endproc
```



SVL script for the SCSI-2

```
"model_1.bcg" = branching reduction of
                    total rename "ARB !.*" -> ARB  in
                      hide CMD, REC in
                        "SCSI.lotos" ;
"model_2.bcg" =  generation of
                    hide all but LAMBDA, MU, NU in
                      ("model_1.bcg" |[ARB]| "erlang.lotos":BUS1 [ARB, NU]) ;
% DISK_SPEED=400
% for BUS_SPEED in 400 4000 40000  (* from 2.5 ms down to 250 µs *)
% do
  % for LOAD in 10 25 50 100 200 400 800 1600  (* from 100 ms down to 625 µs *)
  % do
      "model_3.bcg" = branching stochastic reduction of
                        total rename
                           "NU" -> "BUS; rate $BUS_SPEED",
                           "MU !$DISK_L" -> "DISK_L; rate $DISK_SPEED",
                           "MU !$DISK_M" -> "DISK_M; rate $DISK_SPEED",
                           "MU !$DISK_H" -> "DISK_H; rate $DISK_SPEED",
                           "LAMBDA !.*" -> "rate $LOAD"
                        in "model_2.bcg" ;
      % bcg_steady -thr -append "$BUS_SPEED.thr" "model_3.bcg" LOAD=$LOAD
  % done
% done
```

Influence of the controller position



Summary and findings

- The SCSI-2 was analyzed both for functional and performance aspects
- The ‘Bull fix’ (*putting the OS on the lowest disk and put the controller in highest priority position*) is explained
- Performance study suggests a better solution: *put the controller in lowest priority position*

Conclusion

Conclusion

- Three scientific goals:
  - Combine functional verification and performance evaluation
  - Broaden the CADP toolkit to performance analysis
  - Tackle large models compositionally
- A pragmatic approach:
  - use LOTOS "as is" (no syntax extension)
  - reuse many existing CADP tools (caesar, bcg_labels, SVL)
  - new tools: bcg_min, determinator, bcg_steady, bcg_transient
- Part of next version of CADP

  `http://www.inrialpes.fr/vasy/cadp`
- Future work
  - direct analysis of IMC
  - model checking of Markov chains

Bibliography

- [CZ96] G. Ciardo et R. Zijal. *Well-defined stochastic Petri nets*. MASCOTS'96
- [DS99] D.D. Deavours et W.H. Sanders. *An efficient well-specified check*. PNPM'99
- [GH02] H. Garavel et H. Hermanns. *On Combining Functional Verification and Performance Evaluation using CADP*. FME'02
- [GL01] H. Garavel et F. Lang. *SVL: A Scripting Language for Compositional Verification*. FORTE/PSTV'01
- [Her98] H. Hermanns. *Interactive Markov Chains and the Quest for Quantified Quality*.
- [Her01] H. Hermanns. *Construction and Verification of Performance and Reliability Models*. EATCS'01
- [HJ03] H. Hermanns et C. Joubert. *A Set of Performance and Dependability Analysis Components for CADP*. TACAS'03
- [Kun86] K.S. Kundert. *Sparse Matrix Techniques*. CASD'86
- [Ste94] W.J. Stewart. *Introduction to the numerical solution of Markov chains*.

More information?

http://www.inrialpes.fr/vasy

http://depend.cs.uni-sb.de/