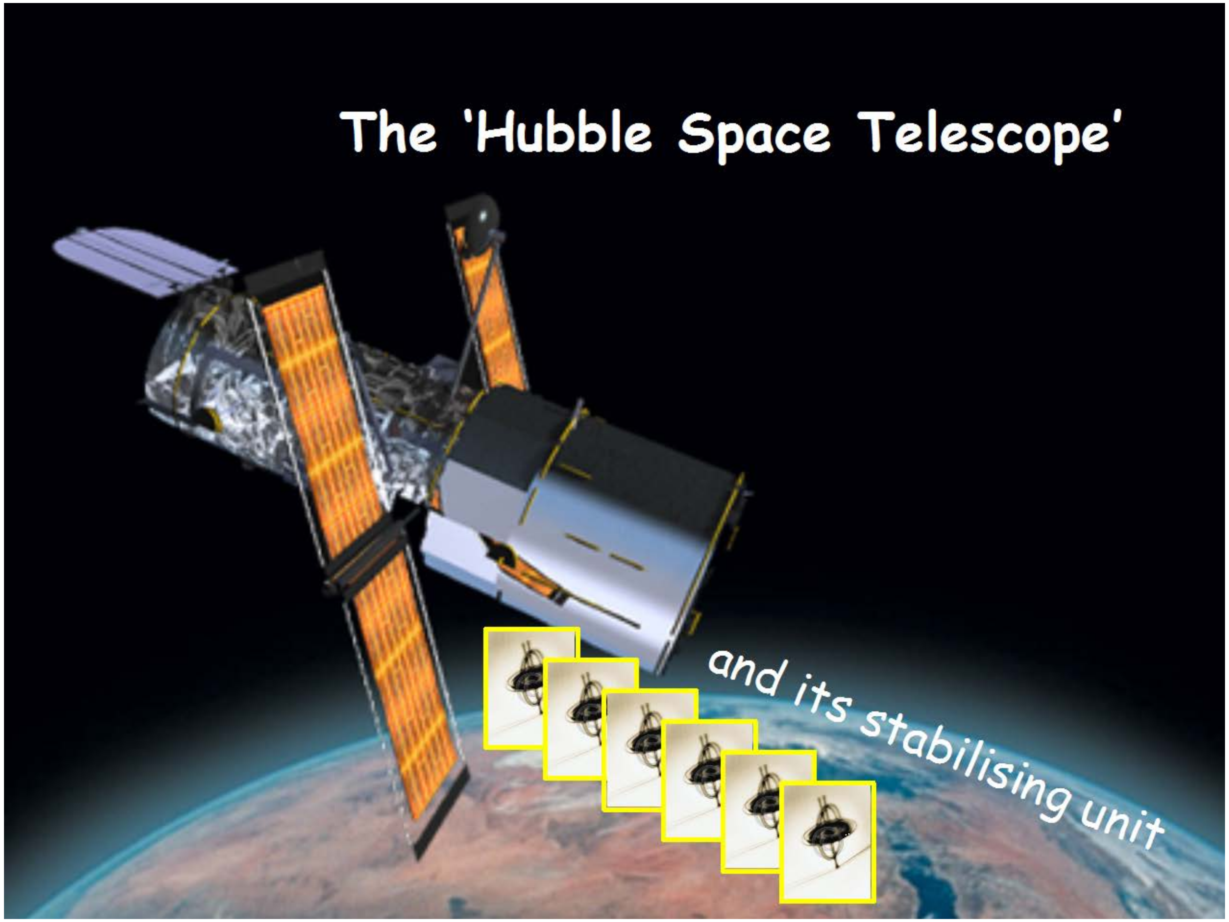
A complete application of Interactive Markov Chains: The Hubble space telescope

Hubert Garavel (based on H. Hermanns' work)

Univ. Grenoble Alpes – MOSIG – November 2022

The Hubble Space Telescope



Overview

- Hubble needs three gyroscopes to stabilize
- It has six gyroscopes (i.e., three spare)
- Each gyroscope may fail as time passes
- Goal: estimate the lifetime of Hubble

From: Holger Hermanns. Construction and Verification of Performance and Reliability Models. Bulletin of the European Association of Theoretical Computer Science 74:135-154, 2001

- Initially: Hubble specification written in LOTOS
- Here: updated, simpler version written in LNT

Methodology

- Model Hubble as a parallel composition of IMCs
- Assign plausible values to rate parameters
- Generate the global state space (IMC)
- Hide all visible actions
  the IMC contains only τ- and rate-transitions
- Minimize the IMC for branching stochastic equiv.
  hopefully, all τ's are removed, so one gets a CTMC
- Perform transient analysis on this CTMC
  to predict the lifetime of Hubble

Modelling details

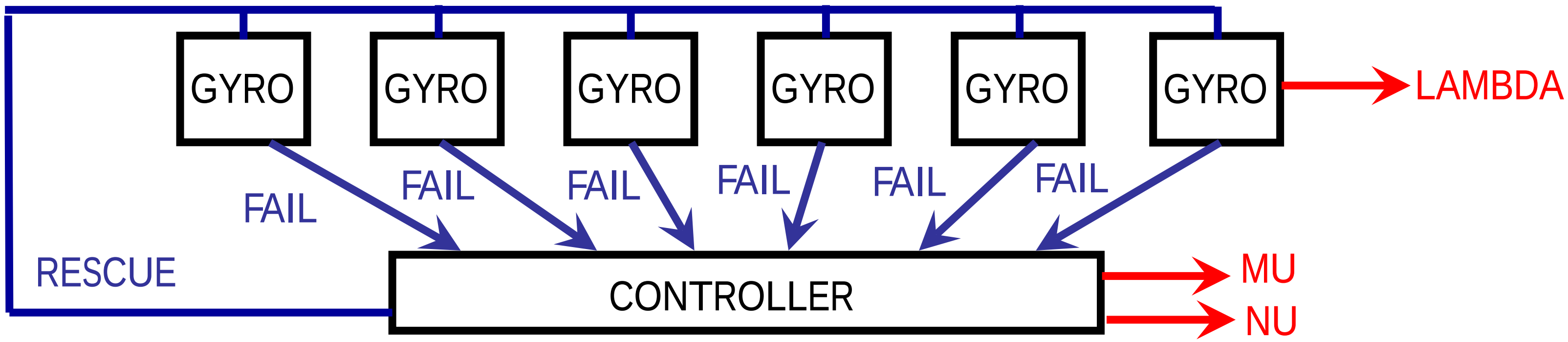
- The Huble telescope has 6 gyroscopes
- As time passes, the gyros may fail
- The average lifetime of gyros is 10 years (= 120 months)

  λ = *12 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)

  μ = *12 months / 0.12 = 100*
- When in sleep mode, a shuttle mission must be sent to repair/reset Hubble, which takes about 2 months

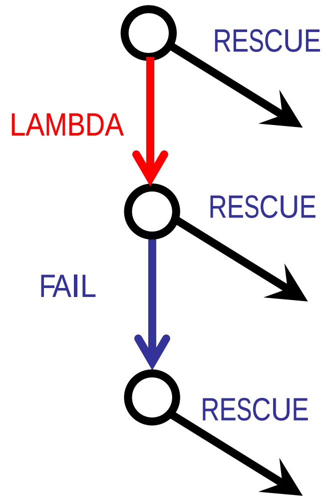
  ν =*12 months / 2 = 6*
- Without operational gyro, Hubble crashes

Component architecture of the Hubble




```
process HUBBLE [LAMBDA, MU, NU : delay] is
   hide  FAIL, RESCUE : none  in
      loop
         par FAIL, RESCUE  in
            CONTROLLER [FAIL, RESCUE, MU, NU]  ||
            par
               GYRO [FAIL, RESCUE, LAMBDA]  ||  GYRO [FAIL, RESCUE, LAMBDA]  ||
               GYRO [FAIL, RESCUE, LAMBDA]  ||  GYRO [FAIL, RESCUE, LAMBDA]  ||
               GYRO [FAIL, RESCUE, LAMBDA]  ||  GYRO [FAIL, RESCUE, LAMBDA]
            end par
         end par     -- reset after rescue by the space mission: gyros replaced, controller restarted
      end loop
   end hide
end process
```

The GYRO process

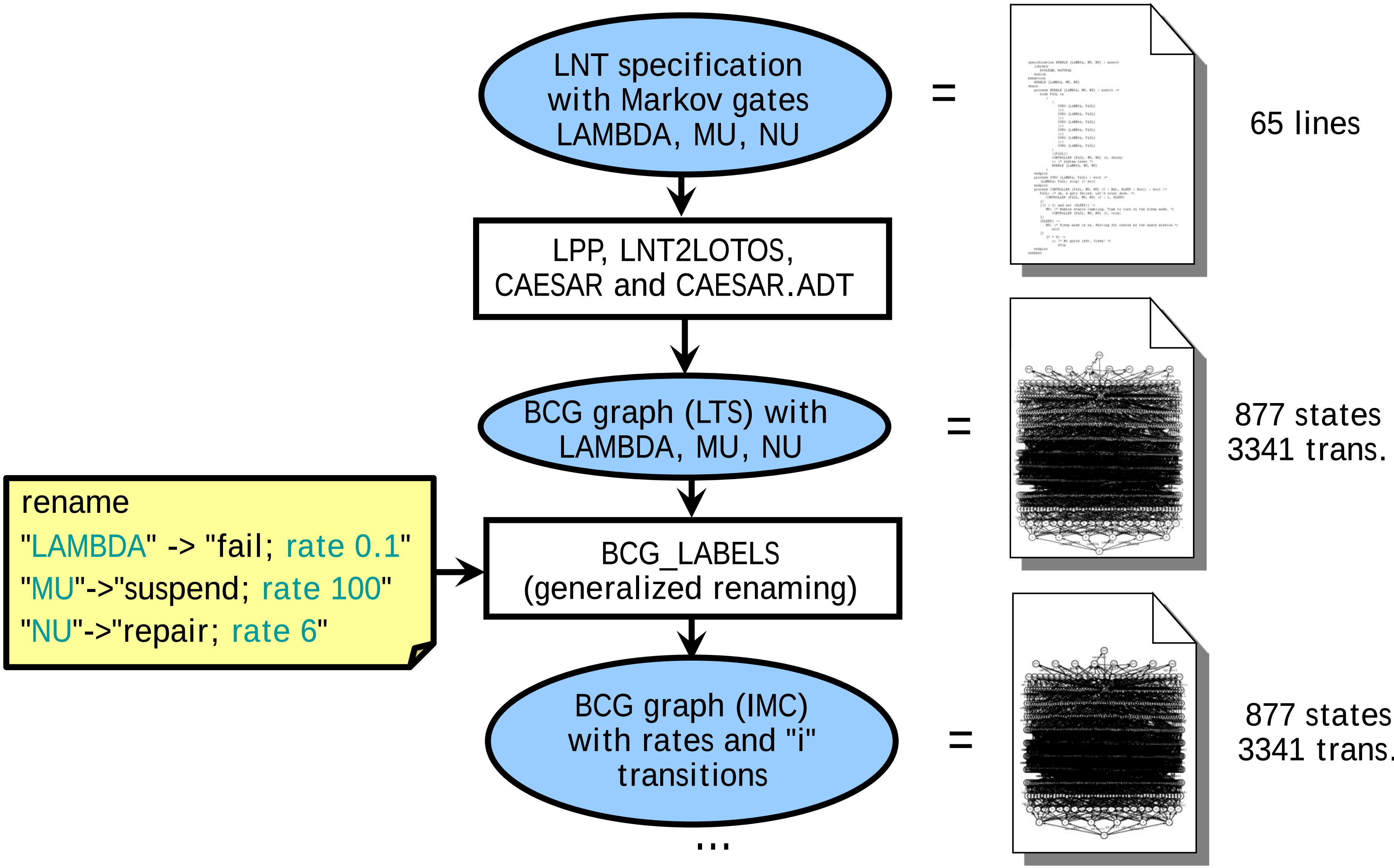



```
process  GYRO [FAIL, RESCUE : none, LAMBDA : delay]  is
  disrupt
    LAMBDA;
    FAIL;
    stop
  by
    RESCUE – The space mission arrives
end process
```

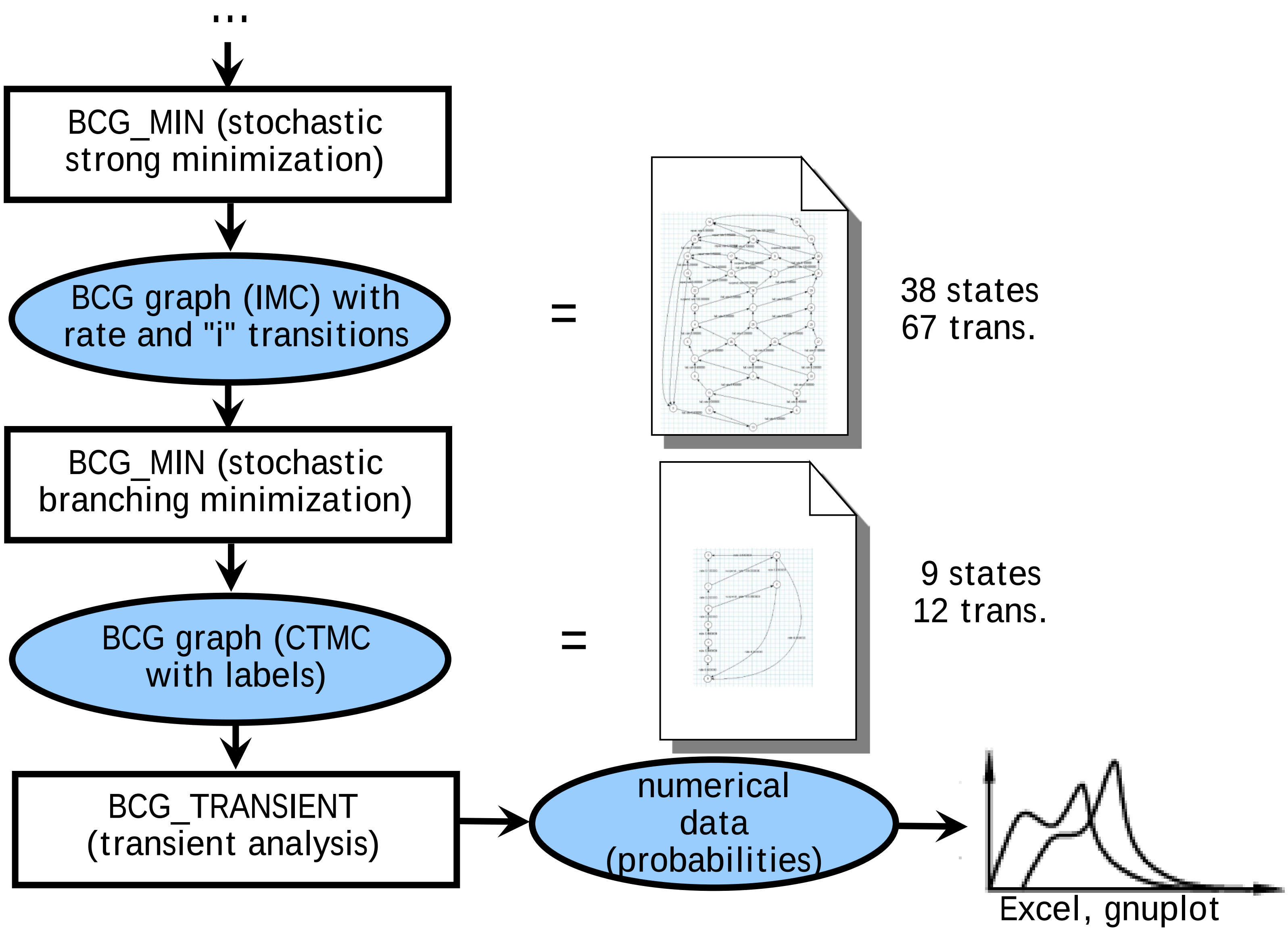
The CONTROLLER process

```
process CONTROLLER [FAIL, RESCUE: none, MU, NU : delay] is
    var C : nat, SLEEP : bool in
        C := 6; SLEEP := false;  -- Initially, there are six functioning gyros and sleep mode is off
        loop L in
            select
                FAIL; C := C - 1 -- Ah, a gyro failed. Let's count down
            [] only if (C < 3) and not (SLEEP) then
                   MU;  SLEEP := true  -- Hubble starts tumbling. Time to turn on the sleep mode
               end if
            [] only if SLEEP then
                   NU; RESCUE; break L   -- Sleep mode is on. Waiting for the space mission
               end if
            [] only if C == 0 then
                   i;  stop    -- No gyros left. Crash!
               end if
            end select
        end loop
    end var
end process
```

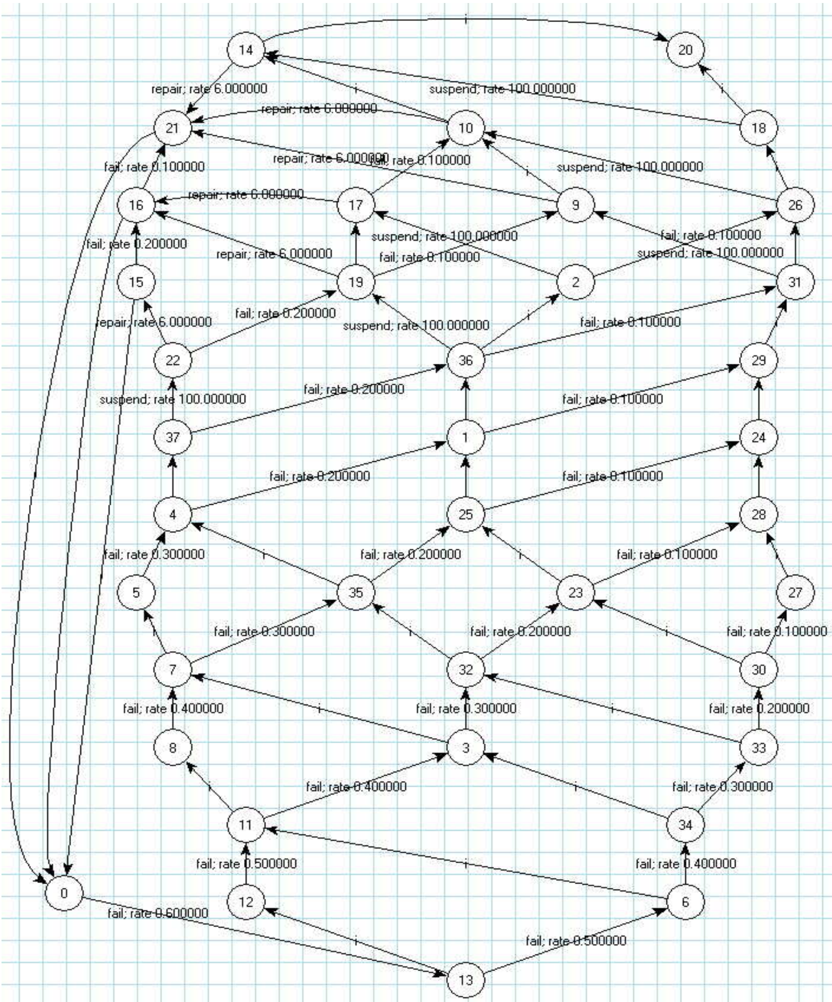
Analysis trajectory for the Hubble (1)



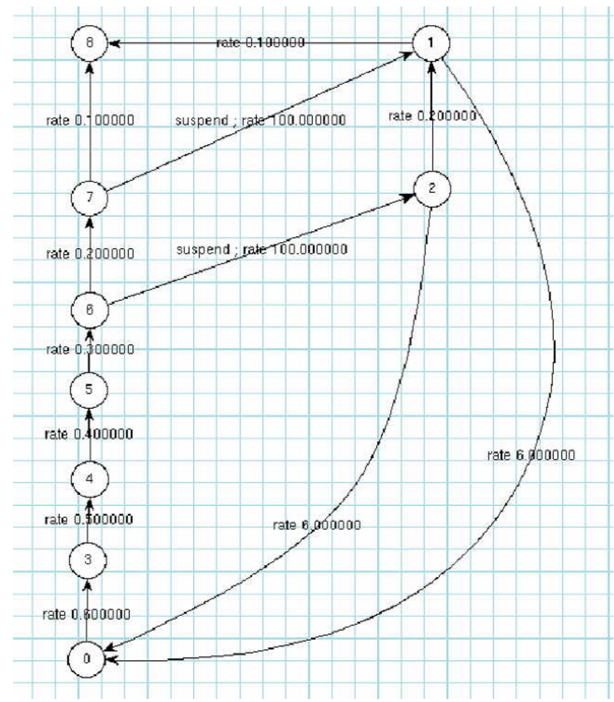
Analysis trajectory for the Hubble (2)



Minimized IMCs for the Hubble

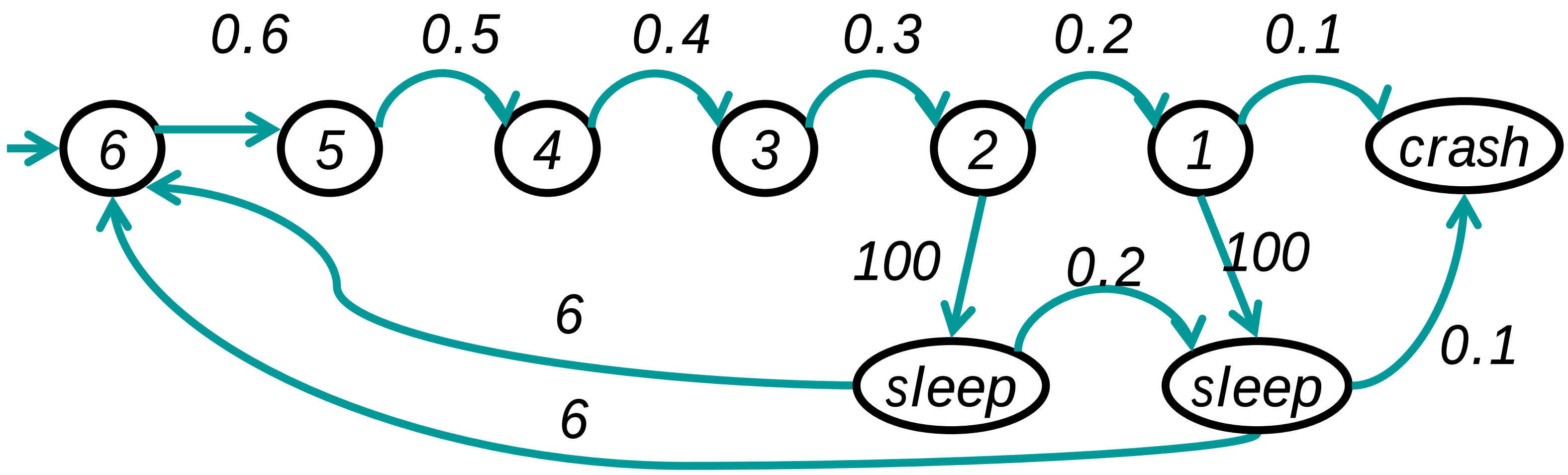


after stochastic strong minimization
(38 states, 67 transitions)



after stochastic branching minimization
(9 states, 12 transitions)

Visual verification of the final CTMC



Analysis of the Hubble using BCG_TRANSIENT

- Transitions of the CTMC are labelled with **tags** (otherwise, only rate values could be observed):
  - **repair**: the space shuttle mission resets the Hubble
  - **suspend**: the Hubble has entered sleep mode
  - **fail**: the Hubble crashes
- **Throughtput**: average number of times a transition labelled a given tag is fired per time unit

time	throughput of "repair"	throughput of "fail"	throughput of "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