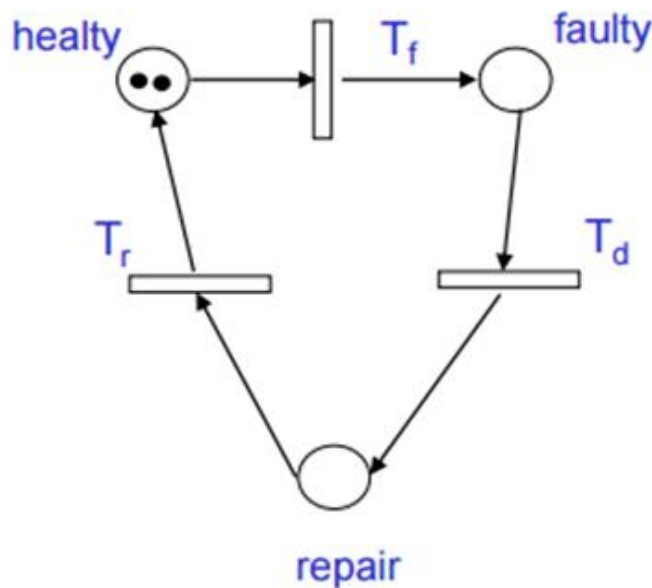


Stochastic Activity Networks in Moebius

Failure-Detection-Repair

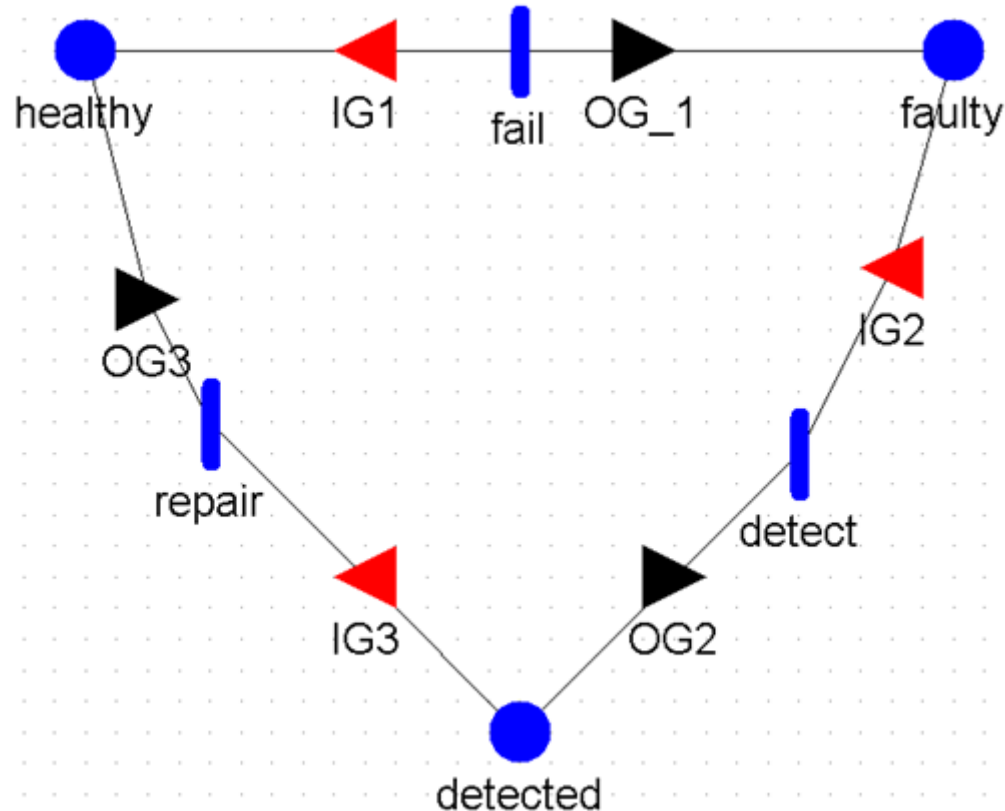


- Two identical CPUs
- **Failure of the CPU:**
exponentially distributed
with parameter λ
- **Fault detection:**
exponentially distributed
with parameter δ
- **CPU repair:**
exponentially distributed
with parameter μ



Evaluate
Availability of
the system
during steady
state,
considering that
the system is
working if at
least one CPU is
healthy

Atomic Model



Remember to edit three global variables
(lambda, mu and delta)

Set the initial state for places)

(healthy = 2, faulty = 0, detected = 0)

Set the rate of each event as the
number of tokens in the input place times
the rate of the event

Set the input enabling function, stating
that the activities are enabled if there is
at least one token in the input place

Set the output function, so that it
decreases the number of tokens in the
input place and increases the ones of
the output place by one

Fail example



Input Predicate

```
healthy->Mark() > 0
```

Output Function

```
faulty->Mark()++;  
healthy->Mark()--;
```

Name:

fail

Time distribution function:

Exponential



1

Rate

```
return lambda*healthy->Mark();
```

Case quantity:

1

Case 1

1

Reward model



- Create a performance variable called **availability**.
- Express its **reward function** according to the condition of correctness of the system.
- Set a steady-state **Time** option with default configurations.

Available State Variables (double click to insert)

```
ex1_san->healthy  
ex1_san->faulty  
ex1_san->detected
```

Reward Function

```
if( ex1_san->healthy->Mark() > 0) return 1;  
else return 0;
```

Study model



The dialog box is titled "Incremental Range". It contains the following fields and options:

- Study:** rstudy (with a "View Values" button next to it)
- Variable:** lambda
- Type:** double
- Initial:** 0.001
- Final:** 0.005
- Increment:** 0.002
- Options:** Three radio buttons are present: "Additive" (selected), "Multiplicative", and "Exponential".
- Buttons:** "OK" and "Cancel" buttons at the bottom.

Set a **range study** model where all the three rates vary from 0.01 to 0.05 with a step of 0.02.

All the possible combinations lead to **27 experiments**

Transformer and Solver



Again use the State Space Generator
(NOT Symbolic) as transformer

Then, for simplicity,
select the Direct Steady State Solver.

The SSG Info dialog box contains the following fields and controls:

- Study Name:
- Experiment List:
 - Experiment_1
 - Experiment_2
 - Experiment_3
 - Experiment_4
 - Experiment_5
 - Experiment_6
- Run Name:
- Build Type:
- Trace Level:
- Hash Value:
- ☐ Flag Absorbing States
- ☐ Place Comments in Output
-

The Direct Steady State Solver dialog box contains the following fields and controls:

- Input / Output tabs
- Direct Steady State Solver computes the *steady state* measures defined in the reward model.
- State Space Name:
- Stopping Criterion:
- Rows:
- Stability:
- Tolerance:
- Verbosity:
- Output File Name:
- Debug File Name:
- State Vector File Name:
- ☐ Plot Complementary Distribution
- ☐ Run In The Background
- ☐ Place Comments in Output
-

- Experiment 1:
lambda = 0.001, **delta** = 0.001, **mu** = 0.001 → Availability = 0.5
- Experiment 14:
lambda = 0.003, **delta** = 0.003, **mu** = 0.003 → Availability = 0.5
- Experiment 4:
lambda = 0.003, **delta** = 0.001, **mu** = 0.001 → Availability = 0.224
- Experiment 3:
lambda = 0.001, **delta** = 0.005, **mu** = 0.001 → Availability = 0.582
- Experiment 19:
lambda = 0.001, **delta** = 0.001, **mu** = 0.005 → Availability = 0.696

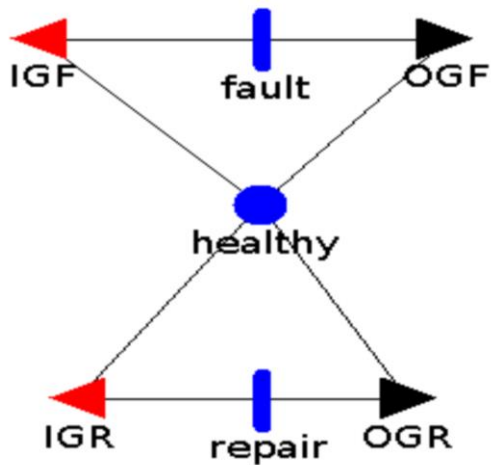
Exercise



- Evaluate steady state availability of the **TMR with repair** and **ideal voter**
- **Study** the availability with different repair rate values w.r.t. the same failure rate

Exercise

- Evaluate steady state availability of the **TMR with repair** and **ideal voter**
- **Study** the availability with different repair rate values w.r.t. the same failure rate



Reward Function

```
if (TMRrep->healthy->Mark() > 1) return 1;  
else return 0;
```

William H. Sanders and John F. Meyer, ``Stochastic Activity Networks: formal definitions and concepts'', in Lectures on formal methods and performance analysis: first EEF/Euro summer school on trends in computer science, 2002.

https://www.mobius.illinois.edu/wiki/index.php/Möbius_Documentation

Thanks to prof. Andrea Domenici for previous version of the slides.