Dependable Systems - Lab Report

NAME Name

Version 1.1
2024-07-02

Inhaltsverzeichnis

1 Exercise 2a - Fault-Tolerant Computer System

The Model should have these properties:

1. A fault-tolerant computer system consist of two main CPUs.
2. The fault-tolerant computer system tolerates the failure of any one of the main CPUs. I.e., as long as only one main CPU fails, the system remains operational.
3. Failure rate = 1/1000; Repair rate = 1/10

I modeled the states as follows:

s=0	The system is fully functional.
s=1	One CPU has failed, the system is still running.
s=2	Both CPUs have failed, the system is down.

I implemented this in 2a_computerSystem.pm:

```
ctmc

//definition of the failure rate
const double rate_failure = 1/1000;
const double rate_repair = 1/10;
module PC
        //definition of states
        s: [0..2] init 0;
        //guard -> rate: action;
        [] s=0 -> rate_failure*2: (s'=1);
        [] s=1 -> rate_repair: (s'=0);
        [] s=1 -> rate_failure: (s'=2);
        [] s=2 -> rate_repair*2: (s'=1);
endmodule

//definiton of the reward system
rewards
        s=0: 1;
        s=1: 1;
endrewards
```

To calculate the MTTF and the availability of the system I used query2a.pct1:

```
//mttf
R=? [ F(s=2) ]
//Result: 51494.764692749566 (+/- 0.5132659280373123 estimated; rel err
    9.967341944366238E-6)

//availabilty
S=? [ !(s=2) ]
//Result: 0.9999019703872098
```

I got the results with this command: *./prism 2a_computerSystem.pm query2a.pct1 -sor*

1.1 Plotting reliability

I used the script bellow to get some values for the reliability of the system and plotted it with latex.

```
#!/bin/bash

outfile=result_2a_v2.csv
#ensure the output file is empty
echo "Time,Reliability" > $outfile

for t in $(seq 0 2000 50000)
do
        #create query
        echo "P=? [ !(F[0, $t] s=2) ]" > help.pctl
        result=$(./prism 2a_computerSystem.pm help.pctl | grep -oP '(?<=
            Result: )[^ ]+')
        #save the result to the CSV file
        echo "$t,$result" >> $outfile
done
```

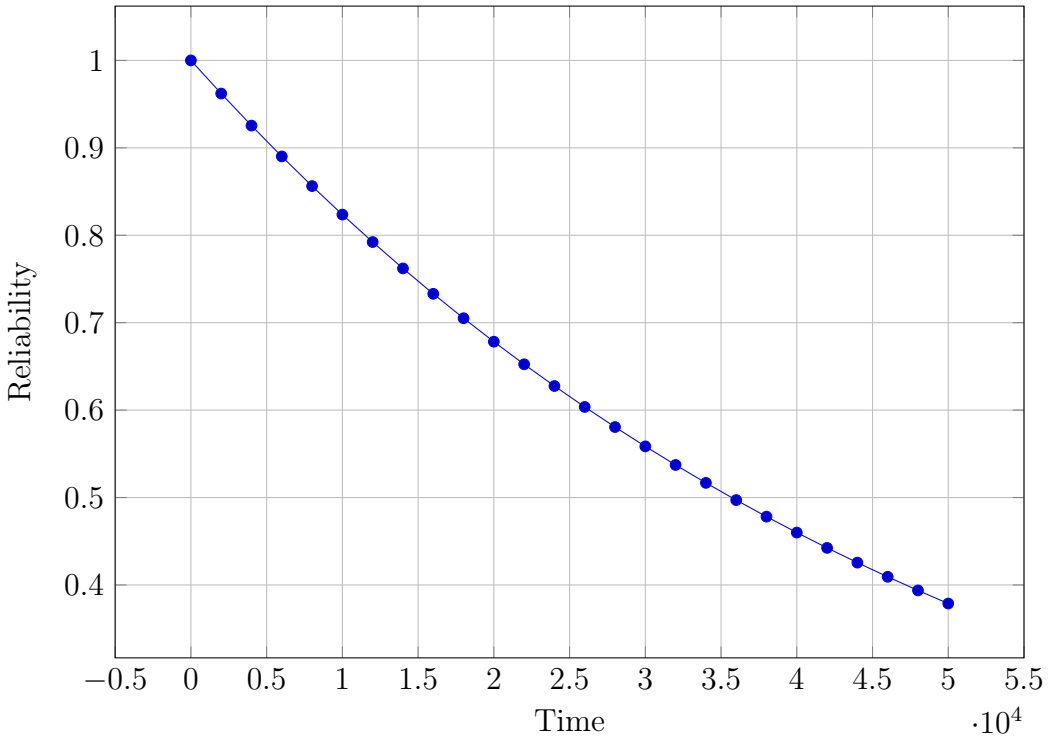

Abbildung 1: Reliability over Time

2 Exercise 2b - Fault-Tolerant Computer System

The Model should have these properties:

1. A fault-tolerant computer system consist of two main CPUs.

2. The fault-tolerant computer system tolerates the failure of any one of the main CPUs. I.e., as long as only one main CPU fails, the system remains operational.

3. Failure rate = 100 FIT; Repair rate = none

4. Assumption Coverage = 0.7

I modeled the states as follows:

s=0	The system is fully functional.
s=1	One CPU has failed (with our assumed fail behavior), the system is still running.
s=2	The system is not operational.

Our new failure rate:
1 FIT = $10^{-9} \frac{\text{failures}}{\text{hour}}$
100 FIT = $100 \cdot 10^{-9}$

I implemented this in 2b.pm:

```
ctmc

//definition of the failure rate
const double rate_failure = 100 * 1e-9;
const double coverage = 0.7;
module PC
        //definition of states
        s: [0..2] init 0;
        //guard -> rate: action;
        [] s=0 -> rate_failure*2*coverage: (s'=1);      //good failure
        [] s=0 -> rate_failure*2*(1-coverage): (s'=2);  //bad failure
        [] s=1 -> rate_failure: (s'=2);
endmodule

//definiton of the reward system
rewards
        s=0: 1;
        s=1: 1;
endrewards
```

To calculate the MTTF and the availability of the system I used query2b.pctl:

```
//mttf
R=? [ F(s=2) ]
//Result: 1.2E7 (exact floating point)

//availabilty
S=? [ !(s=2) ]
//Result: 0.0
```

I got the results with this command: *./prism 2b.pm query2b.pctl*

2.1 Plotting reliability

I used the script bellow to get some values for the reliability of the system and plotted it with latex.

```
#!/bin/bash

outfile=result_2b.csv
#ensure the output file is empty
echo "Time,Reliability" > $outfile

for t in $(seq 0 200000 5000000)
do
        #create query
        echo "P=? [ !(F[0, $t] s=2) ]" > help.pctl
        result=$(./prism 2b.pm help.pctl | grep -oP '(?<=Result: )[^ ]+')
        #save the result to the CSV file
        echo "$t,$result" >> $outfile
done
```

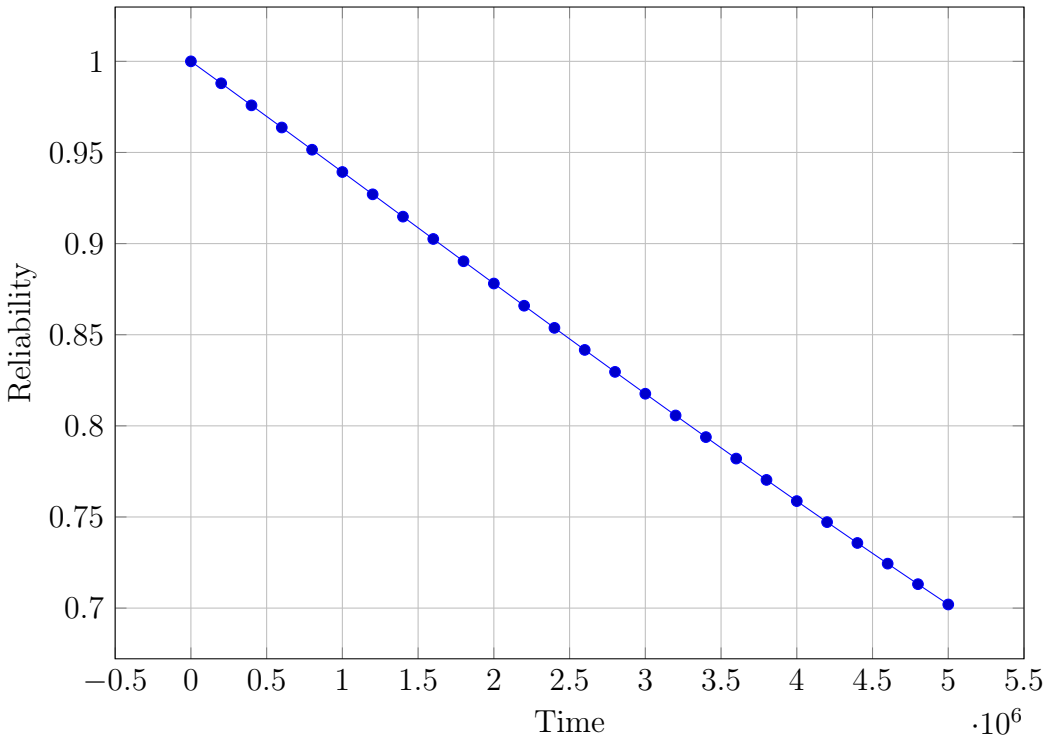

Abbildung 2: Reliability over Time

3 Exercise 3 - Your turn!

My model describes a person and their relationship status:

1. in a relationship (good state):

(a) normal fights decrease the quality of the relationship (relationship component failure)

(b) there can be fights that destroy the relationship (not covered relationship component failure)

(c) talking can make the relationship better (relationship repair)

(d) if the relationship quality is to bad it is broken (relationship failure)

2. if not in a relationship, they can find a (new) relationship (repair)

3. can get depressed and not be able to find a new relationship (full system failure)

I modeled this with those states:

$s > 0$	in a relationship, how good the relationship is depends on the value of s
$s = 0$	currently alone
$s = -1$	depressed and therefore not able to find a new relationship

An implementation of this system can be seen in 3.pm:

```
ctmc

//definitions of rates
const double fights = 1/96;      //fight rate (1/(4 days))
const double c_fights = 0.999;   //coverage for fights
const double talk = 1/24;        //relationship talk rate
const int states = 8;            //quality levels of a relationship
const double new = 1/9000;       //rate of finding a new relationship (~1 y)
const int start_new = floor(states/2);
const double dep = 1/35000;      //rate of depression if alone (~1/(4y))
module Person
        //definition of states
        s: [-1..states] init start_new;
        //guard -> rate: action;
        [] s > 0 -> fights*c_fights: (s'= s-1);          //good failure
        [] s > 0 -> fights*(1-c_fights): (s'= 0);        //bad failure
        [] s > 0 & s < states -> talk : (s'= s+1);
        [] s = 0 -> new : (s'= start_new);
        [] s = 0 -> dep : (s'= -1);
        [] s = -1 -> 1 : (s'= -1);
endmodule

//definiton of the reward system
rewards
        s > 0: 1;
endrewards
```

To calculate the MTTF and the availability of the system I used query3.pctl:

```
//mttf
R=? [ F(s=0) ]
//Result: 92377.90895729311 (+/- 0.9232281743199776 estimated; rel err
    9.994036287905064E-6)
// (around 10.5 years)

```

```
//availabilty
S=? [ !(s=0 & s=-1) ]
//Result: 1.0
```

I used this command to get the results: *./prism 3.pm query3.pctl -maxiters 100000 -sor*

3.1 Plotting reliability

I used the script bellow to get some values for the reliability of the system and plotted it with latex.

```
#!/bin/bash

outfile=result_3.csv
#ensure the output file is empty
echo "Time,Reliability" > $outfile

for t in $(seq 0 5000 100000)
do
        #create query
        echo "P=? [ !(F[0, $t] s=2) ]" > help.pctl
        result=$(./prism 3.pm help.pctl | grep -oP '(?<=Result: )[^ ]+')
        #save the result to the CSV file
        echo "$t,$result" >> $outfile
done
```

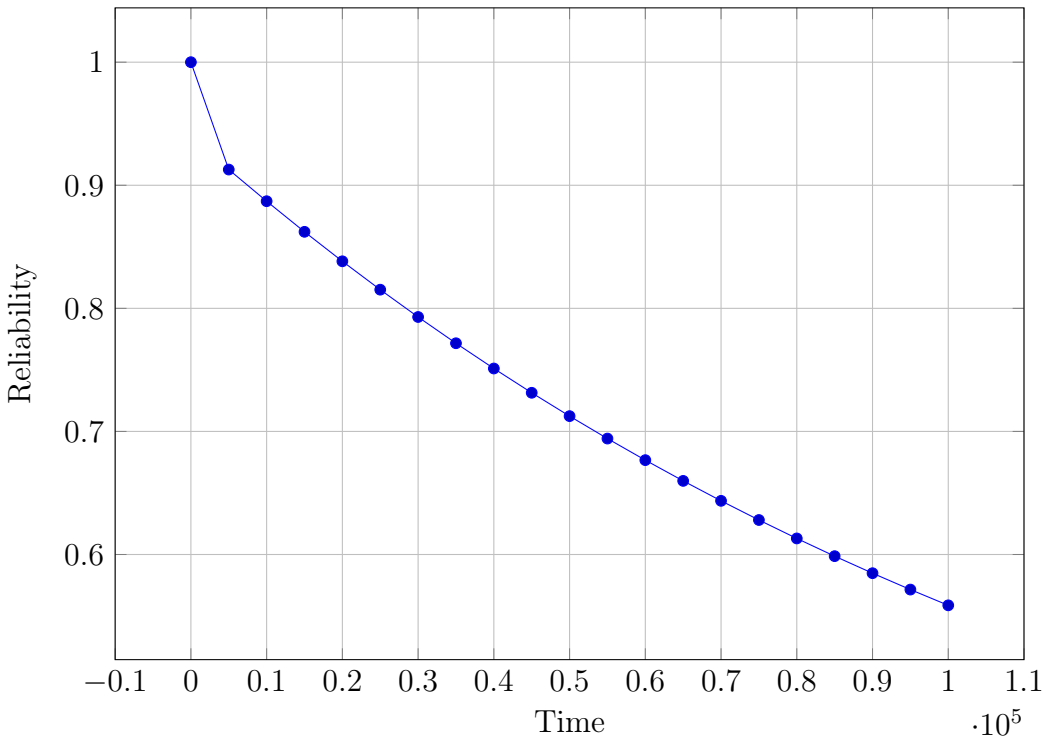

Abbildung 3: Reliability over Time