Software Quality FS 2011

Exercise 1 - Discussion

Cédric Jeanneret

Requirements Engineering Research Group

Department of Informatics

University of Zurich

http://www.ifi.uzh.ch/rerg/people/jeanneret

University of Zurich^UZH

Grading
What are these funny symbols?

Exercise 1 had 7 parts

- Chameleons Colony (5), Petrinet and Lamport's Bakery

--, -, ~, +, ++

- Ordinal scale

Average: **median**

- The median of 3 assignments is the second-ranked grade

LTL Properties
Safety? Liveness?

Safety

*Something bad **never** happens*

– Violations always have a finite witness

– Checked on finite executions

□¬ all chameleons are of the same color

Liveness

*Something good **keeps** happening*

– Violations never have a finite witness (but may have a finite cycle)

– Checked on infinite executions

□◊ some chameleons transmute

LTL Properties

Safety? Liveness?

Pr1: Absence of deadlock

□ (t1enabled V t2enabled V t3enabled V t4enabled)

Pr2: T4 can be fired at least once

◊ t4enabled

Pr3: T3 can be fired an infinite number of time

□◊ t3enabled

Pr4: As soon as P4 receives a token, it never gets empty again

□¬p4 V (¬p4 U □ p4)

LTL Properties

To negate or not to negate?

SPIN looks for a trace satisfying a given property

When investigating whether...

... *P* holds, let SPIN search for a **counterexample**

```
spin –a –f "!P"  ...
spin –a –f "![]P"  ...
```
or
```
spin –a –f "<>!P" ...
```

... *P* can be true, let SPIN search for an **example**

```
spin –a –f "P"  ...
spin –a –f "<>P"  ...
```

LTL Properties

Never claims

(never claims generated from LTL formulae are stutter-invariant)

pan: claim violated! (**at depth 43**)
pan: wrote Colony.pml.trail

[…]

State-vector 28 byte, **depth reached 43**, errors: 1
        22 states, stored
         0 states, matched
        22 transitions (= stored+matched)
         0 atomic steps

LTL Properties

Never claims

Automaton for the mutation process

```
./pan –d
proctype mutations
     state  23 -(tr   7)-> state  23  [id   6 tp   2] [D---G] line 13 => D_STEP
     state  23 -(tr   8)-> state  23  [id  13 tp   2] [D---G] line 13 => D_STEP
     state  23 -(tr   9)-> state  23  [id  20 tp   2] [D---G] line 13 => D_STEP
     state  23 -(tr   2)-> state  23  [id  21 tp   2] [----G] line 13 => else
state 23 line 13 is a loopstate
```

proctype :never:

```
     state   5 -(tr   3)-> state   7  [id  31 tp   2] [----G] line 48 =>
            (!((((nRed&&nBlue)||(nRed&&nGreen))||(nGreen&&nBlue))))
     state   5 -(tr   1)-> state   5  [id  33 tp   2] [----G] line 48 => (1)
     state   7 -(tr   1)-> state   8  [id  37 tp   2] [-a--L] line 52 => (1)
     state   8 -(tr   4)-> state   0  [id  38 tp 3500] [--e-L] line 53 => -end-     [(257,9)]
state 5 line 48 is a loopstate
```

Automaton for the verification of the property

Lamport's Bakery
Hints

Define 2 arrays as global variables

- bit choosing[N] and byte number[N]

Define 1 inline "procedure"

- To compute the maximum number in the number[] array

Define 1 process (given number of iterations)

- Sequence: enter CS, do something in CS, exit CS
- Local variable: _pid

Do not use atomic or d_step blocks