TIMO SCHNEIDER <TIMOS@INF.ETHZ.CH>

DPHPC Recitation Session
SPIN Tutorial

Last week:

- **Roofline model**
- **Balance principle**
- **Basic idea: Models for performance expectation!**

This week:

- **You have heard a lot about locks**
  - **They are complicated**
  - **They are difficult to optimize**
  - **Over-optimization quickly leads to incorrect locks**
  - **So how do we make sure locks (or other parallel primitives) are correct in practice?**

Reasoning about correctness

- We have the necessary tools for proving correctness!
- Are they practical? Why or why not?

- Example:
  - Recent bachelor thesis defense talk:
    - Implemented hierarchical R/W looks for distributed systems, based on MCS lock
    - Question:
      - How did ensure correctness?
      - How would you do it?

SPIN

- **One tool that can aid us in proving properties of parallel code is SPIN**
- **Free software, "large" user base, lots of documentation on spinroot.com**
- **SPIN is based on model checking**
- **There are other approaches**
  - **Abstract interpretation**
  - **Profiling/Tracing (does this really prove something?)**
- **General idea:**
  - **Represent process as a state-machine**
  - **When processes run in parallel, generate all possible interleaving of states (state space explosion!)**
  - **Formulate invariants for the combined state machine**

SPIN

- **Need a way to describe these state machines**
- **If that description is "too far off" from our code, we risk specifying the wrong state machine!**

- **SPIN = Simple Promela Interpreter**
- **PROMELA = PROcess MEta LAnguage**

Hello World

```
/* A "Hello World" Promela model for SPIN. */

active proctype Hello() {
        printf("Hello process, my pid is: %d\n", _pid);
}

init {
        int lastpid;
        printf("init process, my pid is: %d\n", _pid);
        lastpid = run Hello();
        printf("last pid was: %d\n", lastpid);
}
```

Execute in random simulation mode: spin -n2 hello.pr

PROMELA Semantics

- The body of a process consists of a sequence of statements. A statement is either
  - executable: the statement can be executed immediately
  - blocked: the statement cannot be executed.
- An assignment is always executable.
- An expression is also a statement; it is executable if it evaluates to non-zero.
  - 2 < 3 always executable
  - x < 27 only executable if value of x is smaller 27
  - 3 + x executable if x is not equal to –3
- The assert-statement is always executable.
- If <expr> in assert evaluates to zero, SPIN will exit with an error, as the <expr> “has been violated”

PROMELA Example: Mutual Exclusion?

```
bit flag;       /* signal entering/leaving the section */
byte mutex;  /* # procs in the critical section. */

proctype P(int i) {
        flag != 1;
        flag = 1;
        mutex++;
        printf("MSC: P(%d) has entered section.\n", i);
        mutex--;
        flag = 0;
}

proctype monitor() {
        assert(mutex != 2);
}

init {
 run P(0); run P(1); run monitor();
}
```

PROMELA Example: Mutual Exclusion?

```
bit x, y;        /* signal entering/leaving the section */
byte mutex;  /* # of procs in the critical section. */

active proctype A() {
        x = 1;
        y == 0;
        mutex++; mutex--;
        x = 0;
}

active proctype monitor() {
        assert(mutex != 2);
}

active proctype B() {
        y = 1;
        x == 0;
        mutex++; mutex--;
        Y = 0;
}
```

PROMELA Example: Mutual Exclusion?

- **Show how these things can be run in practice and what we can observe (DEMO)**
  - **Random simulation mode – this is like software testing**
  - **Guided simulation mode (-i)**
  - **Verification mode (spin -a lock.pr; gcc pan.c; ...)**
    - **This is why people use SPIN**
    - **Generates a verifier in C code, so that compiler can optimize it**
    - **Then exhaustively searches all possible states**
    - **Can be slow/eat all your memory**

PROMELA Semantics: if

```
if
:: choice1 -> stat1.1; stat1.2; stat1.3; …
:: choice2 -> stat2.1; stat2.2; stat2.3; …
:: …
:: choicen -> statn.1; statn.2; statn.3; …
fi;
```

- **If there is at least one choice (guard) executable, the if statement is executable and SPIN non-deterministically chooses one of the executable choices.**
- **The “else” choice is executable iff no other choices are**
- **If no choice is executable, the if-statement is blocked**

PROMELA Semantics: do

```
do
:: choice1 -> stat1.1; stat1.2; stat1.3; …
:: choice2 -> stat2.1; stat2.2; stat2.3; …
:: …
:: choicen -> statn.1; statn.2; statn.3; …
od;
```

- **With respect to the choices, a do-statement behaves in the same way as an if-statement.**
- **However, instead of ending the statement at the end of the choosen list of statements, a do-statement repeats the choice selection.**
- **The (always executable) break statement exits a do-loop statement and transfers control to the end of the loop**

PROMELA Semantics: Communication

- **Communication between processes is via channels:**
  - **message passing**
  - **rendez-vous synchronisation (handshake)**
- **Both are defined as channels**

```
chan <name> = [<dim>] of {<t1>,<t2>, … <tn>};

Example:
mtype {MSG, ACK};
chan toS = [2] of {mtype, bit};
```

PROMELA Semantics: Communication

- **channel = FIFO-buffer (for dim>0)**
- **! Sending - putting a message into a channel**

  ch ! <expr1>, <expr2>, … <exprn>;

- **Values of <expri> must correspond with the types of the channel declaration.**
- **A send-statement is executable if the channel is not full.**
- **? Receiving - getting a message out of a channel**

  ch ? <var1>, <var2>, … <varn>;

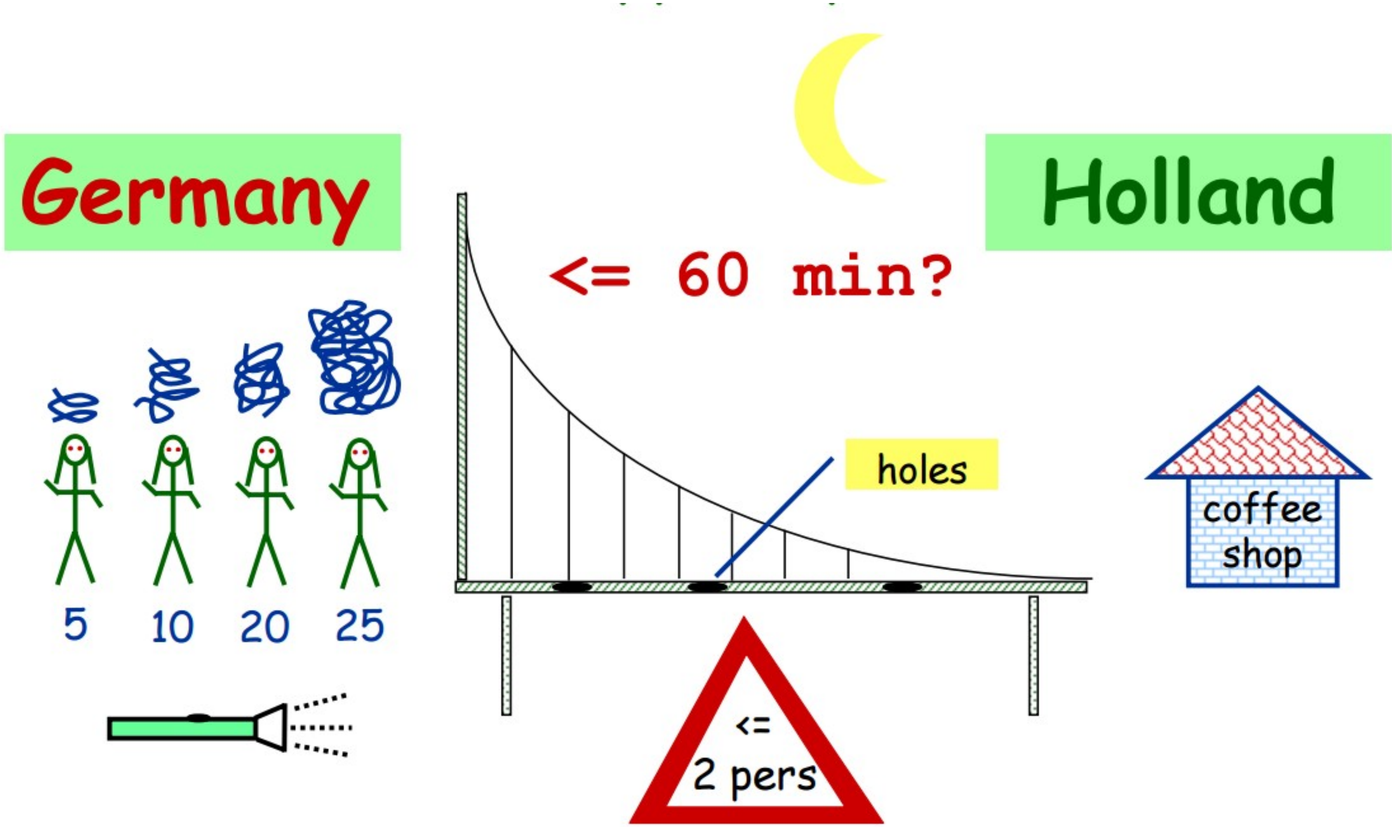
- **If the channel is not empty, the message is fetched from the channel and the individual parts of the message are stored into the <vari>s.**

  ch ? <const1>, <const2>, … <constn>;

- **If the channel is not empty and the message at the front of the channel evaluates to the individual <consti>, the statement is executable and the message is removed from the channel.**

Homework

- **Use SPIN to solve the following riddle:**



Homework

- **Take the "better than Dijkstra" lock from the slides of lecture 10 and show that it does not work using SPIN.**