Introduction to Model Checking

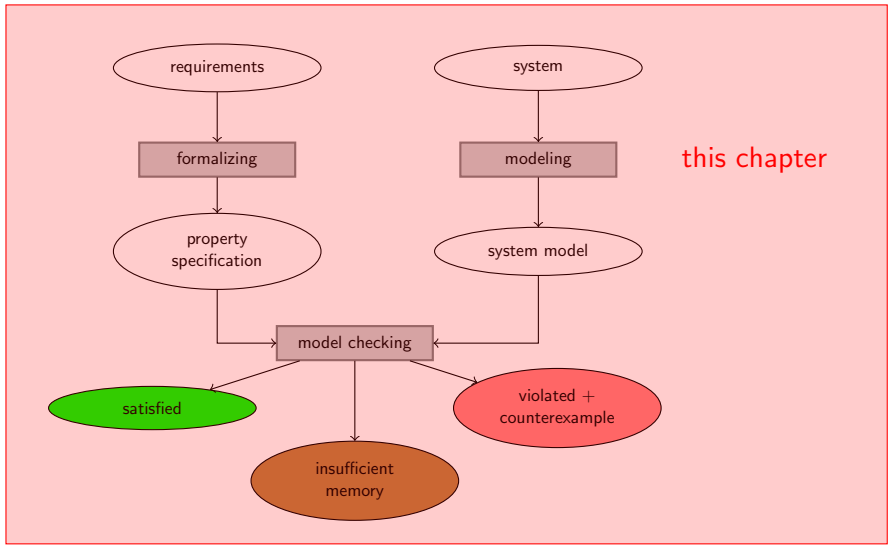
René Thiemann

Institute of Computer Science
University of Innsbruck

WS 2011/2012

Outline

Model checking overview



Spin

- developed by Bell Labs
- achieved ACM system software award
- features
  - LTL model checker
  - Promela as specification language
  - ...
- freely available (`http://spinroot.com`)
- literature

  The SPIN MODEL CHECKER
  Primer and Reference Manual
  Addison-Wesley

LTL in Spin

- differences in syntax:

$$\begin{aligned} \wedge &\equiv \&\& \\ \vee &\equiv \| \\ \neg &\equiv ! \\ \Rightarrow &\equiv -> \\ G &\equiv [] \\ F &\equiv <> \end{aligned}$$

- X operator per default not available
  (if enabled and used, turn off partial order reduction!)
- atomic propositions via #define-directive
  - `#define ap1 (x > 0)`
  - `#define ap2 (a[1] == y && x > 2)`
- NBAs are generated from LTL formula
  (or can be entered instead of LTL formula)

Promela (process meta language)

- is specification language used in Spin
- is similar to imperative programming languages
- has some specialities
  - `if` and `do` with several conditions

```
if :: x > 0 -> stmt1
   :: y > 0 -> stmt2
   :: z < 5 -> stmt3
fi
```

non-deterministic choice of satisfied condition

no condition is satisfied $\Rightarrow$ { exit of loop `do` / wait `if` }

  - processes; communication via channels or shared variables
  - keyword `atomic` to execute statements without interruption
  - ...

Peterson's mutual exclusion algorithm

```
byte x, b1, b2, crit1, crit2; // global vars

proctype one()
{
  do :: true ->
     b1 = 1;
     x = 2;
     if :: (b2 == 0 || x == 1) ->
        crit1++;
     fi;
     crit1--;
     b1 = 0;
  od
} /* two similar */

init {
  run one(); run two();
}
```


Keywords

- `proctype` is used to declare processes
- `init` is the initial process
- `run proc_name()` starts a process
- global variables are declared outside `proctype` declarations
- local variables are declared within `proctype` declaration
- variables are initialized with 0

Using Spin

- load or enter program
- `Run / LTL Property manager`
- load or specify requirements
  - define atomic proposition (`Symbol Definitions`)
  - enter `Formula`
  - `Generate` Büchi automata
- `Run Verification`
  - often: `With Weak Fairness` (schedule all processes if possible)
  - increase memory under `Set Advanced Options` if out-of-memory error
  - `Run`
- In case of non valid verification result:
  `Run Guided Simulation` and step through trace to spot error.

Example

DEMO

Peterson's mutual exclusion algorithm using arrays

```
#define N 2
bool x, b[N];
byte crit;
active [N] proctype one_or_two()
{
  do :: true ->
     b[_pid] = 1;
     x = 1 - _pid;
     if :: (!b[1 - _pid] || x == _pid) ->
        crit++;
     fi;
     crit--;
     b[_pid] = 0;
  od
}
```


Keywords (continued)

- `active [n] proctype name() { ...}` starts n processes during initialization
- `_pid` is the number of the process from $0, \ldots, n-1$
- `#define` is often used to define nr of components / clients / ...

Summary

- Spin can be used to analyze real programs
- communication via shared variables demonstrated
- communication via channels not yet shown (next Chapter)