Formal Methods in Software Engineering

Paul Wild

Tutorial session 1

Tuesday 22nd October, 2019

Tutorial procedure

Homework

- 50% of your grade comes from homework exercises
- exercise sheets will appear regularly
- submission by e-mail, usually until before the next tutorial

Class

- homework presentation, comparison of solutions, discussion of problems
- we will experiment with the tools during class
- active participation required

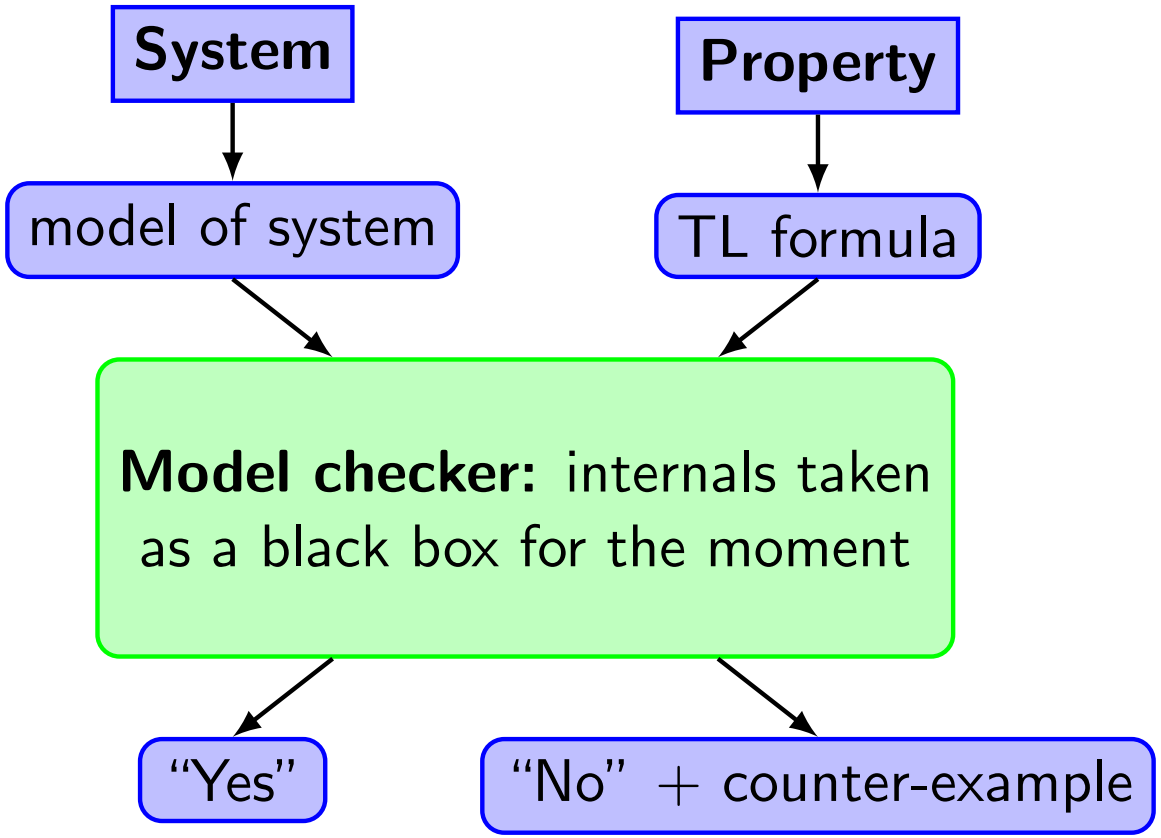
Model Checking and Temporal Logics

Model-based verification techniques

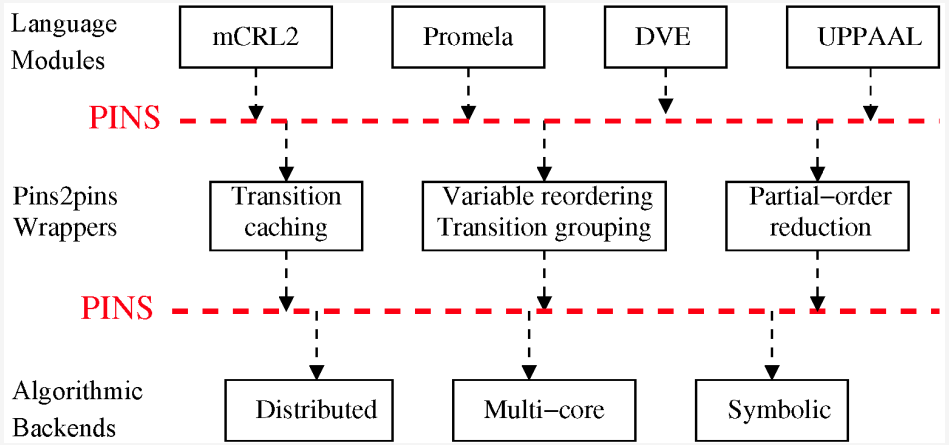
- describe a *system* and its behaviour in a mathematically precise and unambiguous manner
- use algorithms to explore all possible states of the models
- **model checking:** exhaustive search
- **simulation:** experiments with a restrictive set of scenarios
- **testing:** experiments on a "real" implementation of the model

Model Checking and Temporal Logics

Model Checking



Overview of LTSmin tools



The tools of LTSmin are made up of three layers:

- *Language front-ends* that support different specification languages for models and translate them to the PINS interface.
- *PINS2PINS wrappers* that transform the models, enabling various optimizations and model checking.
- *Algorithmic backends* offering various algorithms for reachability and model-checking (both explicit-state and symbolic).

Basic syntax of Promela

Manuals

Promela is the modelling language of the Spin model checker.
Documentation can be found here:

- `http://spinroot.com/spin/Man/`

Basic syntax of Promela

```
#define __instances_A 1
#define __instances_B 1

byte state = 1;

proctype A()
{ byte tmp;
  (state==1) -> tmp = state; tmp = tmp+1; state = tmp
}

proctype B()
{ byte tmp;
  (state==1) -> tmp = state; tmp = tmp-1; state = tmp
}

init
{ atomic { run A(); run B() }
}
```

LTL model checking for Promela models

- Compile the model into PINS format:

```
$ spins example.prm
```

- Check an LTL formula and store a counterexample:

```
prom2lts-seq \
  --ltl="!␣<>(state==0␣&&␣<>(state==1))" \
  --trace=trace.gcf example.prm.spins
```

- Print a trace containing a counterexample:

```
ltsmin-printtrace trace.gcf | grep action
```

Dining Philosophers

Setting

- n philosophers are sitting around a circular dining table and there are n forks, one between each pair of adjacent philosophers.
- Whenever a philosopher is hungry, they first grab the fork to their left, then the fork to their right, and then they eat.
- When they are done they release both forks.

Modelling in Promela

We will now try to model this in Promela.

- The philosophers will be modelled as processes (`proctype`).
- The forks will be modelled as channels (`chan`).

What are some properties of this system that we may want to check?