# 15. The SPIN Model Checker


Computer-Aided Verification

Dave Parker

University of Birmingham

2017/18

# Module syllabus

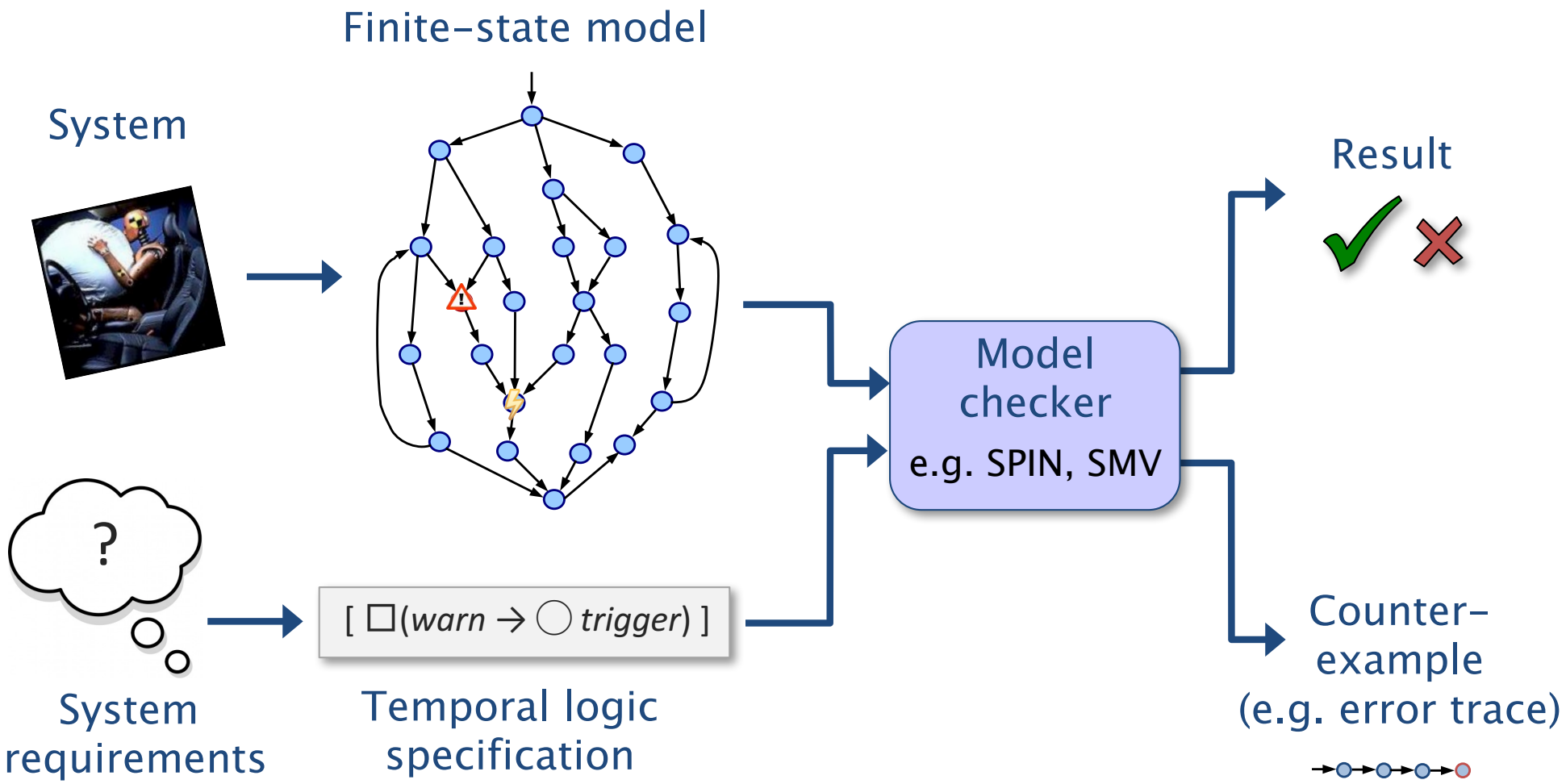
- Modelling sequential and parallel systems
  - labelled transitions systems, parallel composition
- Temporal logic
  - LTL, CTL and CTL*, etc.
- Model checking
  - CTL model checking algorithms
  - automata-theoretic model checking (LTL)
- Verification tools: SPIN
- Advanced verification techniques
  - bounded model checking via propositional satisfiability
  - (symbolic execution), (symbolic model checking)
- Quantitative verification
  - (real-time systems), probabilistic systems

# Overview

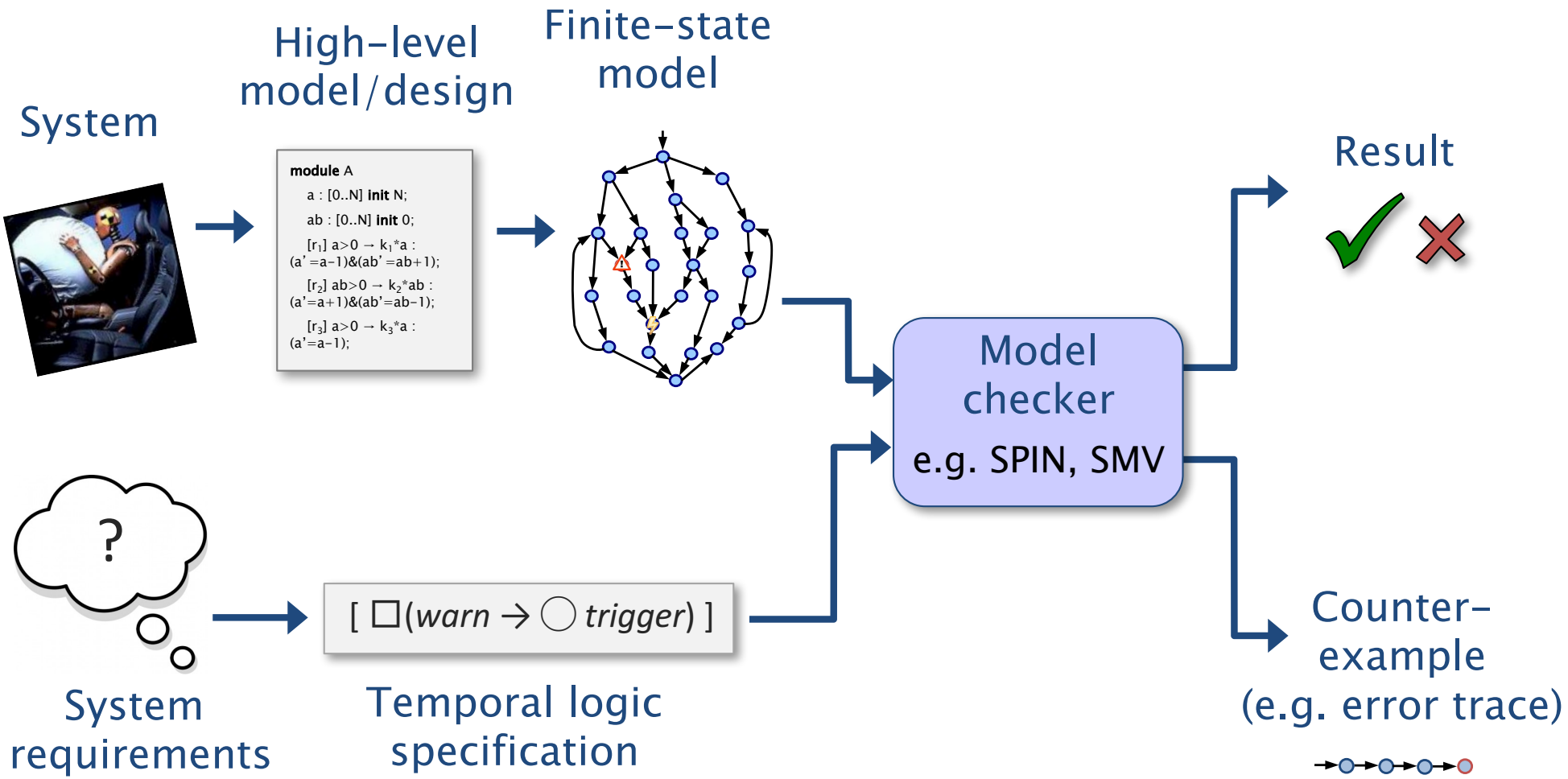
- The SPIN model checker
  - overview
  - ProMeLa
  - demo
  - examples

- Background reading & reference:
  - basic manual: http://spinroot.com/spin/Man/Manual.html
  - tool options: http://spinroot.com/spin/Man/Spin.html

# Verification via model checking



# Verification via model checking



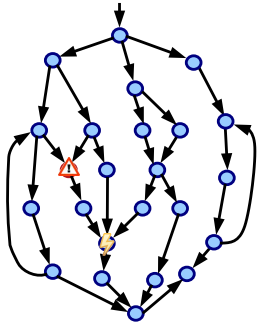
# Verification via model checking

System

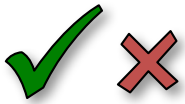
High-level model/design

```
module A
    a : [0..N] init N;
    ab : [0..N] init 0;
    [r1] a>0 → k1*a :
(a' =a-1)&(ab' =ab+1);
    [r2] ab>0 → k2*ab :
(a'=a+1)&(ab'=ab-1);
    [r3] a>0 → k3*a :
(a'=a-1);
```

Finite-state model



Result



Model checker

e.g. SPIN, SMV

?

System requirements

[ □(*warn* → ◯ *trigger*) ]

Temporal logic specification

Counter-example (e.g. error trace)

# High-level model description

- Example concurrent program:

```
process Inc = while true do if x < 200 then x := x + 1 fi od

process Dec = while true do if x > 0 then x := x − 1 fi od

process Reset = while true do if x = 200 then x := 0 fi od
```

- ProMeLa/SPIN model (fragment):

```
int x=0;
proctype Inc () {
      do :: true −> if :: (x < 200) −> x = x + 1 fi od
}
proctype Dec() {
      do :: true −> if :: (x > 0) −> x = x − 1 fi od
}
…
```

# SPIN

- SPIN model checker
  - prominent, widely used verification tool
  - open source, freely available (http://spinroot.com/)
  - developed by Gerald Holzmann at Bell Labs
  - many success stories: NASA mission software (Deep Space 1, Mars Exploration Rovers), Toyota control software investigation
- Key features
  - custom modelling language: ProMeLa
  - on-the-fly model checking for safety, liveness, LTL
    - verification vs falsification (bug hunting)
    - state storage using hash table of lists of states
  - simulator (random, interactive, guided)
  - separate user interface (iSpin)

# ProMeLa

- ProMeLa: Process Meta Language
  - modelling language for the SPIN tool
- Key ingredients
  - processes (one or more, in parallel)
  - typed data variables (can be shared, for communication)
  - channels (synchronous/asynchronous communication)
- Language notation
  - guarded commands (nondeterministic choice)
  - plus imperative-style control flow (and embedded C)
- Semantics
  - labelled transition systems (LTSs), via program graphs

# ProMeLa – Guarded commands

- Guarded commands

```
:: guard -> statement
```

  - execute statement if guard is true
- Conditionals

```
if
:: guard -> statement1
:: !guard -> statement2
fi
```

  - if (guard) then $statement_1$ else $statement_2$

Nondeterminism

- Loops

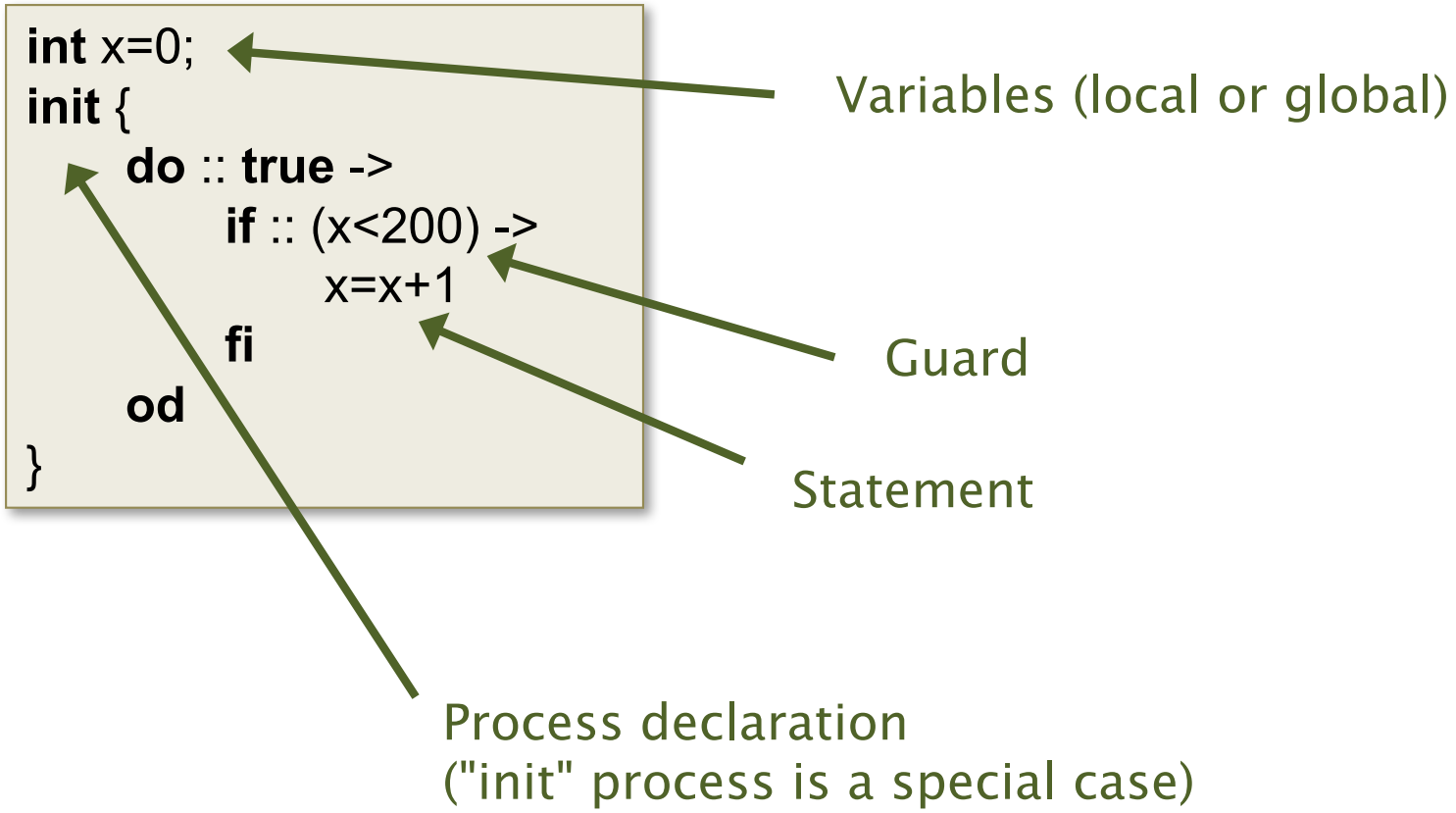
```
do :: guard -> statement od
```

  - while (guard) do statement

```
do :: guard1 -> statement1
:: guard2 -> statement2
od
```

# ProMeLa – Example

- One process, one variable (integer x)



# Demo

# ProMeLa – Concurrent program

- Earlier example:

```
int x = 0;

proctype Inc() {
      do :: true -> if :: (x < 200) -> x = x + 1 fi od
}
proctype Dec() {
      do :: true -> if :: (x > 0) -> x = x - 1 fi od
}
proctype Reset() {
      do :: true -> if :: (x==200) -> x=0 fi od
}

init {
      run Inc() ; run Dec() ; run Reset()
}
```

# ProMeLa – Concurrent program

- Add a "monitor" process that checks whether $0 \leq x \leq 200$

```
int x = 0;
proctype Inc() {
      do :: true -> if :: (x < 200) -> x = x + 1 fi od
}
proctype Dec() {
      do :: true -> if :: (x > 0) -> x = x - 1 fi od
}
proctype Reset() {
      do :: true -> if :: (x==200) -> x=0 fi od
}
proctype Check () {
      assert (x >= 0 && x <= 200)
}
init {
      run Inc() ; run Dec() ; run Reset() ; run Check()
}
```