Model Checking with Spin

Presenter : Liang Ge

Outline

- Setting Up
- What is Spin?
- Why Spin?
- Data Structures in Spin
- Verifying Models
- Final Example : Needham-Schroeder Protocol

Outline

- Setting Up
- What is Spin?
- Why Spin?
- Data Structures in Spin
- Verifying Models
- Final Example : Needham-Schroeder Protocol

Setting Up

- Available at http://spinroot.com/spin/Man/README.html
- Available on Unix/Linux, Windows, Macs (Preferably on Unix/Linux)
- Spin(Currently 6.01)

  need GCC
- iSpin(Currently 1.04)

  need Tcl/Tk

Outline

- Setting Up
- What is Spin?
- Why Spin?
- Data Structures in Spin
- Verifying Models
- Final Example : Needham-Schroeder Protocol

What is Spin?

- Pan (Protocol Analyzer)

  First edition of Spin in 1980

- Spin (Simple Promela Interpreter)

  - Promela (Process Meta Language)

    modelling language for describing concurrent systems

  - Run models and try to find errors

  - An efficient software verification system

Outline

- Setting Up
- What is Spin?
- Why Spin?
- Data Structures in Spin
- Verifying Models
- Final Example : Needham-Schroeder Protocol

Why Spin?

- How to verify a system?

  Examine all possible states of a system, see if there are any errors in the states

- Consider the states of a running system

  system with 1 process --- simple

  system with 2 processes

  …

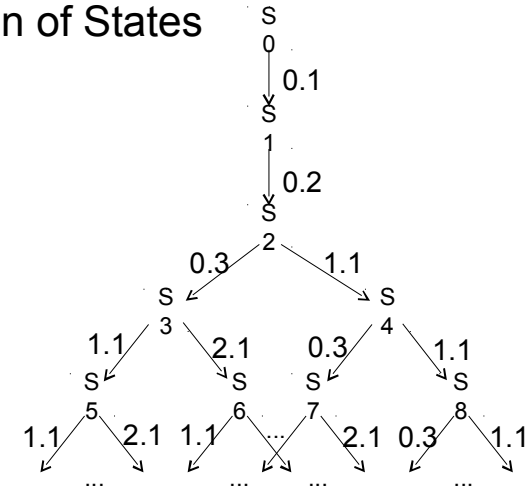
  system with 10 processes --- complicated

Why Spin?

- Tree Representation of States

```
bool x;
proctype Thread i () {
    do :: x = ! x          i.1
    od
}
init {
    x = false;             0.1
    run Thread1();         0.2
    run Thread2()          0.3
}
```

--- Model1



--- State Tree for Model1

Why Spin?

- Reason :

  different schedules (ordering of execution)
- Result :

  state explosion
- Spin can deal with this

  How? See next part

Outline

- Setting Up
- What is Spin?
- Why Spin?
- Data Structures in Spin
- Verifying Models
- Final Example : Needham-Schroeder Protocol

Data Structures in Spin

- State Vector

  "holds the value of all variables as well as program counters (current position of execution) for each process."

- Depth-first Stack

  "holds the states (or transitions) encountered down a certain path in the computation tree."

- Seen State Set

  "holds the state vectors for all the states that have been checked already (seen) in the depth-first search."

Data Structures in Spin

- State Vector

  values of program counter for each process

  0.3, 1.1, -, false

  values of program variables

- Depth-first Stack

  keep track so that Spin can traverse the tree

Data Structures in Spin

- Seen State Set
  - try to draw a new tree representation for model1, this time replace those labels S*i* with state vectors.
  - states start to repeat themselves. (same state vector)
  - infinite → finite

- These data structures make verification possible

Outline

- Setting Up
- What is Spin?
- Why Spin?
- Data Structures in Spin
- Verifying Models
- Final Example : Needham-Schroeder Protocol

Verifying Models

- Example Model

```
byte x, t1, t2;
init {
    x = 1;                          0.1
    run Thread1();          0.2
    run Thread2();          0.3
    assert(x != 3)                  0.4
}
```

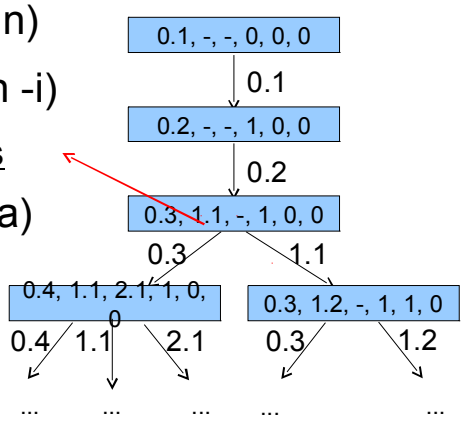
```
proctype Thread1() {
    do :: t1 = x;               1.1
          t2 = x;               1.2
          x = t1 + t2           1.3
    od
}
proctype Thread2() {
    do :: t1 = x;               2.1
          t2 = x;               2.2
          x = t1 + t2           2.3
    od
}
```

--- Model2

Verifying Models

- Spin as a simulator
- - random simulation (spin)
- - guided simulation (spin -i)

  user make choices at forks
- Spin as a verifier (spin -a)
- - exhausive search
- - this is what makes
- spin useful



--- State Tree for Model2

Verifying Models

- Verify : 4 steps
- 1) spin -a *model* - generate verifier
- 2) gcc pan.c -o pan - compile verifier
- 3) ./pan- run verifier, write trace file if error found
- 4) spin -t *model* - simulate program using trace file (recreate error)

Verifying Models

- Types of errors
- - assertion violation detection

- - deadlock detection

  Spin outputs “invalid end state”
- - dead code detection (error?)

  Spin outputs “unreached ...”

Verifying Models

- Examples for deadlock & deadcode

- Verify models in files:

  deadlock

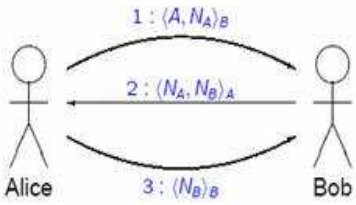
  deadcode

Outline

- Setting Up
- What is Spin?
- Why Spin?
- Data Structures in Spin
- Verifying Models
- Final Example : Needham-Schroeder Protocol

Final Example

- Use Spin to verify the Needham-Schroeder protocol (based on public key cipher)



- secret represented by pair $\langle N_A, N_B \rangle$ of "nonces"
- messages can be intercepted
- assume secure encryption and uncompromised keys

- Try to find a counter example