A Practical Demonstration of the Model Checker SPIN [a]

Nathalie Cauchi

Computer Aided Formal Verification

November, 2018

[a] The slides are based on Giuseppe Perelli and Dieky Aszkiya's presentation

What is SPIN

SPIN is a general tool for:

- verifying the correctness of concurrent software models
- in a rigorous and mostly automated fashion.

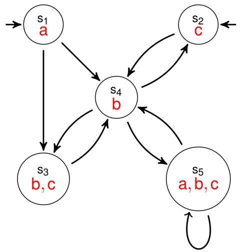
It has been applied to:

- flood control and the verification of the control barriers in the Netherlands
- verification of medical device transmission protocols.

www.spinroot.com

Today we will use the tool to encode transition systems and LTL formulas to be model checked via backward induction.

Transition Systems in SPIN




```
byte state = 1;
bool a = true, b = false, c = false;
active proctype P()
{
do
:: atomic{ state==1 -> state=3; a=false; b=true; c=true }
:: atomic{ state==1 -> state=4; a=false; b=true; c=false }
:: atomic{ state==4 -> state=2; a=false; b=false; c=true }
:: atomic{ state==4 -> state=3; a=false; b=true; c=true }
:: atomic{ state==4 -> state=5; a=true; b=true; c=true }
:: atomic{ state==2 -> state=4; a=false; b=true; c=false }
:: atomic{ state==3 -> state=4; a=false; b=true; c=false }
:: atomic{ state==5 -> state=4; a=false; b=true; c=false }
:: atomic{ state==5 -> state=5; a=true; b=true; c=true }
od
}
```

Execution

- The SPIN code is saved in a text file with extension .pml (e.g. example.pml);
- SPIN can only handle a single initial state in a verification process;
- Since the transition system above has two initial states, then we have to run the verification twice, once for each state, changing the initialization of the variable state;
  - If a property is satisfied by using all the initial states, then the property is satisfied by the transition system;
  - If a property is not satisfied by using some initial states, then the property is not satisfied by the transition system;

Encoding LTL Formulas

Syntax

$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{F}\varphi \mid \mathsf{G}\varphi \mid \varphi \mathsf{U} \varphi$

Operator	Math	SPIN
negation	$\neg$	!
conjuction	$\wedge$	&&
disjunction	$\vee$	||
implication	$\rightarrow$	->
equivalence	$\leftrightarrow$	<->
next	X	X
until	U	U
eventually	F (or $\diamond$)	<>
globally	G or $\Box$	[]

Examples

LTL	SPIN
$\diamond\Box c$	<> [] c
$\Box\diamond c$	[] <> c
$(X\neg c) \rightarrow XXc$	(X ! c) -> (X X c)
$\Box a$	[] a
$aU(b \vee c)$	a U (b || c)
$(XXb)U(b \wedge c)$	(X X b) U (b && c)

Preparing a SPIN file TS1.pml

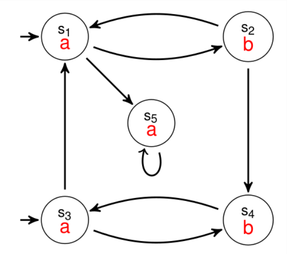
- Attach to file TS1.pml the following code:

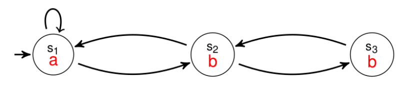
  - ltl F1 {<> [] (c || b)}
  - ltl F1 {<> [] c || b}
  - ltl F1 {<> [] c}

Verification using SPIN

1. Use SPIN with parameter `-a` to the promela file containing both the model and the specifications: `spin -a TS1.pml`.
   This generates a C file called pan.c
2. Compile the C file using GCC: `gcc -o pan pan.c`.
3. Execute the binary file: `./pan -a -N F1`.
   This checks the specification F1 against the model. To check another specification, just replace F1 with either F2 or F3.
4. If the output says error: 0 then the property is satisfied, otherwise the property is not satisfied.
5. In the case a property is not satisfied, we can generate a counterexample: `spin -t -p TS1.pml`

Exercise 1





1. Consider the two transition systems above;
2. Encode them in two separated files, e.g., `TS2.pml` and `TS3.pml`
3. Using SPIN, prove that they are not LTL -equivalent, i,e., there exist two formulas φ_2 and φ_3 such that,
    - TS2 $\models \varphi_2$
    - TS3 $\not\models \varphi_2$
    - TS3 $\models \varphi_3$
    - TS2 $\not\models \varphi_3$

Exercise 2

1. Compare TS2 and TS3 with the following transition system