A Practical Demonstration of the Model Checkers SPIN & NuSMV [a]

Nathalie Cauchi

AIMS: Systems verification

January, 2019

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

Part I: SPIN

What is SPIN

SPIN is a general tool for:

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

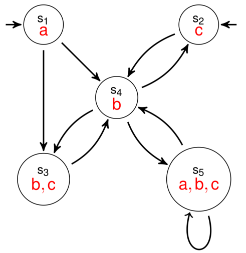
It has been applied to:

- 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`
$(X\,\neg c) \rightarrow X\,X\,c$	`(X ! c) -> (X X c)`
$\Box a$	`[] a`
$a\,U\,(b \vee c)$	`a U (b || c)`
$(X\,X\,b)\,U\,(b \wedge c)$	`(X X b) U (b && c)`

Preparing a SPIN file TS1.pml

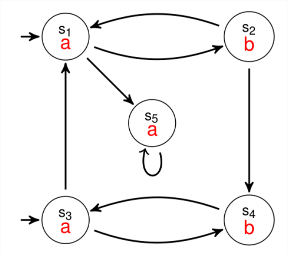
- Attach to file TS1.pml the following code:

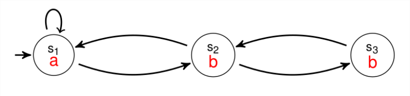
  - ltl F1 $\{<>\ []\ (c \mid\mid b)\}$
  - ltl F1 $\{<>\ []\ c \mid\mid b\}$
  - ltl F1 $\{<>\ []\ c\}$

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 $\not\models \varphi_3$

Part II: NuSMV

What is NuSMV

NuSMV: a symbolic model checker

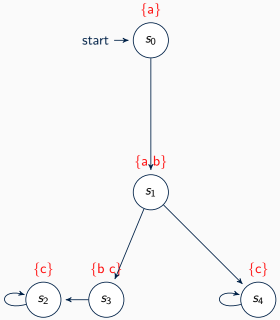
- the first model checker based on BDDs
- open architecture for model checking, used:
  - for verification of industrial designs
  - as a core for custom verification tools [1]

Application

- We will perform two tasks:

  1. We will first use the tool to encode transition systems and LTL and CTL formulas to be model checked.
  2. We will use the tool to perform bounded model checking.

Transition systems in NuSMV




```
MODULE main
VAR
state :{s0,s1,s2,s3,s4};
ASSIGN
init(state) := {s0};
next(state) := case
state=s0 :  s1;
state=s1 :  {s3, s4};
state=s2 :  s2;
state=s3 :  s2;
state=s4 :  s4;
esac;
DEFINE
a := state=s0 | state=s1;
b := state=s1 | state=s3;
c := state=s2 | state=s3 | state=s4;
```

Remark

- The NuSMV code is saved in a text file with extension `.smv`

```
TS1.smv
```

- Unlike SPIN, NuSMV can handle multiple initial states in the verification process. Hence, we only need to run the verification once.
- Can model check both `LTL` and `CTL` properties.

NuSMV specification for LTL and CTL formulae

- An LTL formula consists of atomic proposition(s), boolean operator(s) and temporal operator(s)
- A CTL formula consists of atomic proposition(s), boolean operator(s), temporal operators and path quantifier(s)

operator	math	NuSMV
not	$\neg$	`!`
and	$\wedge$	`&`
or	$\vee$	`|`
implies	$\rightarrow$	`->`
equivalent	$\leftrightarrow$	`<->`
always	$\Box$	`G`
eventually	$\diamond$	`F`
until	U	`U`
next	$\bigcirc$	`X`
for all	$\forall$	`A`
exist	$\exists$	`E`

Examples

- Some examples of the translation of LTL /CTL formula from mathematical notations to NuSMV commands

LTL/CTL formula	NuSMV
$\Diamond\Box c$	`FG c`
$\Box \Diamond c$	`GF c`
$(\bigcirc\neg c) \rightarrow (\bigcirc \bigcirc c)$	`(X ! c) -> (X X c)`
$\Box a$	`G a`
$a\ U\ \Box(b \vee c)$	`a U (G (b | c))`
$(\bigcirc \bigcirc b)\ U\ (b \vee c)$	`(X X b) U (b | c)`
$\exists \Diamond \forall\Box c$	`EF AG c`
$\forall\Box\exists \Diamond \neg c$	`AG EF !c`

Preparing a NuSMV file TS1.smv

- Attach to the file `TS1.smv` the following code:

```
LTLSPEC F G a
CTLSPEC EF AG c
```

Verification using NuSMV

- To verify the transition system against the given specification(s), execute the NuSMV with the parameter name of the smv file:

```
NuSMV TS1.smv
```

- NuSMV automatically generates a counter-example when a specification is not satisfied

Exercise 1

- Verify the transition system used in example (`TS1.smv`) against the following properties:
  - $\Diamond\Box\neg b$
  - $\exists \Diamond (a \wedge b \wedge \forall \bigcirc b)$
  - $\forall\Box(b \rightarrow \forall \bigcirc c)$
  - $\forall\Box(a \leftrightarrow \neg c)$
- In each case, explain why the property was satisfied or not.

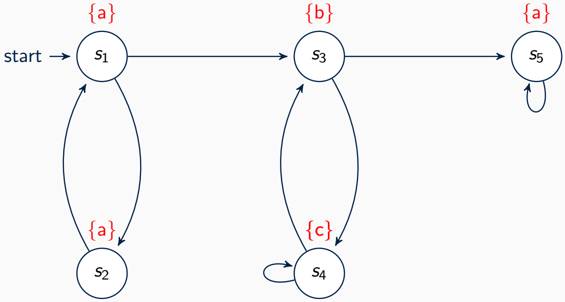
Bounded Model Checking

Recall:

- employs a SAT solver for model checker
- focuses on counterexample generation (up to a certain length)

We will now perform bounded model checking on a transition system.

Bounded Model Checking: Exercise



- Consider the above transition system
- Encode the transition system (e.g. `TS3.smv`)

Bounded Model Checking: Exercise

- Verify the transition system (e.g. `TS3.smv`) against the following properties using bounded model checking

  - $\square \diamond a$
  - $\diamond\square(a \rightarrow (b \rightarrow \diamond c))$
  - $\square(a \wedge (\bigcirc c \rightarrow \diamond a))$

- To do bounded model checking:

```
NuSMV -bmc -bmc_length 2 TS3.smv
```

- Run bounded model checking with different maximum counterexample length and comment on result

The End

Thank you!

nathalie.cauchi@cs.ox.ac.uk