Bounded Model Checking (in NuSMV)*

Alessandra Giordani
agiordani@disi.unitn.it
http://disi.unitn.it/~agiordani

Formal Methods Lab Class, May 9, 2014

Università degli Studi di Trento

*These slides are derived from those by Stefano Tonetta, Alberto Griggio, Silvia Tomasi, Thi Thieu Hoa Le for FM lab 2011/13

Bounded Model Checking

Key ideas:

- looks for counter-example paths of increasing length k (called *bound*) (i.e. path consisting of $k + 1$ states)
  - oriented to finding bugs: *is there a bad behaviour?*
- for each k, builds a boolean formula that is satisfiable iff there is a counter-example of length k
  - can be expressed using $k \cdot |\mathbf{s}|$ variables
- satisfiability of the boolean formulas is checked using a *SAT procedure*
  - can manage complex formulas on several 100K variables
  - returns satisfying assignment (i.e., a counter-example)

Bounded Model Checking

An example: the modulo 8 counter

```
MODULE main
VAR
  b0   : boolean;
  b1   : boolean;
  b2   : boolean;
ASSIGN
  init(b0) := FALSE;
  init(b1) := FALSE;
  init(b2) := FALSE;
  next(b0) := !b0;
  next(b1) := (!b0 & b1) | (b0 & !b1);
  next(b2) := ((b0 & b1) & !b2) | (!(b0 & b1) & b2);
DEFINE  out  := toint(b0) + 2*toint(b1) + 4*toint(b2);
```

Simulating the model

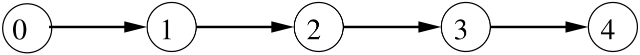
Initializing command: `go_bmc`
Picking initial state command: `bmc_pick_state`
Simulating command: `bmc_simulate`

```
NuSMV > bmc_simulate -k 3 -p
-> State 6.1 <-
    b0 = FALSE
    b1 = FALSE
    b2 = FALSE
    out = 0
-> State 6.2 <-
    b0 = TRUE
    out = 1
-> State 6.3 <-
    b0 = FALSE
    b1 = TRUE
    out = 2
-> State 6.4 <-
    b0 = TRUE
    out = 3
```

Checking LTL specifications

The following specification is false:

```
LTLSPEC G (out = 3 -> X out = 5)
```



- It is an example of *safety* property (→ “nothing bad ever happens”)
  - the counterexample is a *finite* trace (of length 4)
  - there are no counterexamples of length up to 3
- LTL properties can be checked via the `check_ltlspec_bmc` and `check_ltlspec_bmc_onepb` commands

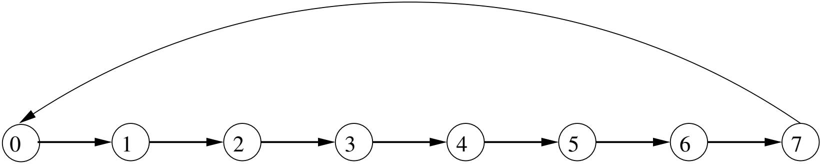
Checking LTL specifications

```
NuSMV > check_ltlspec_bmc -p "G (out = 3 -> X out = 5)"
-- no counterexample found with bound 0 for specification ...
-- no counterexample found with bound 1 for specification ...
-- no counterexample found with bound 2 for specification ...
-- no counterexample found with bound 3 for specification ...
-- specification  G (out = 3 ->  X out = 5)   is false
-- as demonstrated by the following execution sequence
-> State 1.1 <-
    ...
    out = 0
-> State 1.2 <-
    ...
-> State 1.4 <-
    ...
    out = 3
-> State 1.5 <-
    ...
    out = 4
```

Checking LTL specifications

The following specification is false:

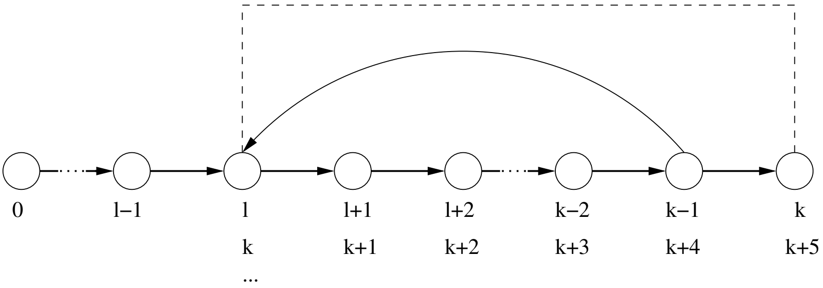
```
LTLSPEC ! G ( F (out =2))      --> F ( G ! (out =2))
```



- It is an example of *liveness* property (→ "something desirable will eventually happen")
  - the counterexample is an *infinite* trace (with a *loop* of length 8)
  - since the state where out = 2 is entered infinitely often,
    the property is false

Bounded Model Checking: counterexamples

The general form of counterexamples found by BMC is the following:



- The counterexample is composed of
  - a *prefix* part (times from `0` to `l-1`)
  - a *loop* part (indefinitely from `l` to `k-1`)
  - as the loop is always backward, it is called *loopback*

Length and loopback condition

- `check_ltlspec_bmc` looks for counterexamples of length up to k.
- `check_ltlspec_bmc_onepb` looks for counterexamples of length k.
- To set the loopback conditions use: `-l bmc_loopback`.
  - `bmc_loopback >=0` : loop to a precise time point
  - `bmc_loopback < 0` : loop length
  - `bmc_loopback = 'X'`: no loopback
  - `bmc_loopback = '*'`: all possible loopbacks
- To set the bounded length use: `-k bmc_length`.
- Default values: `bmc_length = 10`, `bmc_loopback = '*'`
- Default values can be changed using:
  - `set bmc_length k` sets the length to `k`
  - `set bmc_loopback l` sets the loopback to `l`

Checking LTL specifications

Let us consider again the specification ! G (F (out =2))

```
NuSMV > check_ltlspec_bmc_onepb -k 9 -l 0 -p "! G ( F (out =2))"
-- no counterexample found with bound 9 and loop at 0 for specification ...
```

Checking LTL specifications

Let us consider again the specification `!  G ( F (out =2))`

```
NuSMV > check_ltlspec_bmc_onepb -k 9 -l 0 -p "! G ( F (out =2))"
-- no counterexample found with bound 9 and loop at 0 for specification ...


NuSMV > check_ltlspec_bmc_onepb -k 8 -l 1 -p "! G ( F (out =2))"
-- no counterexample found with bound 8 and loop at 1 for specification ...
```

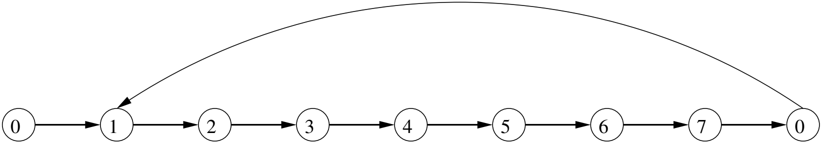
Checking LTL specifications

Let us consider again the specification `! G ( F (out =2))`

```
NuSMV > check_ltlspec_bmc_onepb -k 9 -l 0 -p "! G ( F (out =2))"
-- no counterexample found with bound 9 and loop at 0 for specification ...


NuSMV > check_ltlspec_bmc_onepb -k 8 -l 1 -p "! G ( F (out =2))"
-- no counterexample found with bound 8 and loop at 1 for specification ...


NuSMV > check_ltlspec_bmc_onepb -k 9 -l 1 -p "! G ( F (out =2))"
-- specification ! G  F out = 2   is false
-- as demonstrated by the following execution sequence
...
```



Checking LTL specifications

Let us consider again the specification `!G ( F (out =2))`

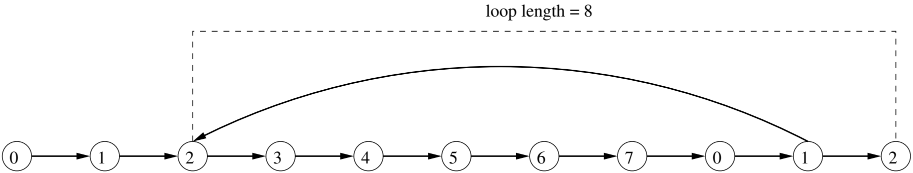
```
NuSMV > check_ltlspec_bmc_onepb -k 9 -l X -p "! G ( F (out =2))"
-- no counterexample found with bound 9 and no loop for specification ...
```

Checking LTL specifications

Let us consider again the specification `!G ( F (out =2))`

```
NuSMV > check_ltlspec_bmc_onepb -k 9 -l X -p "! G ( F (out =2))"
-- no counterexample found with bound 9 and no loop for specification ...
```

```
NuSMV > check_ltlspec_bmc_onepb -k 10 -l -8 -p "! G ( F (out =2))"
-- specification ! G  F out = 2   is false
-- as demonstrated by the following execution sequence
...
```



Checking invariants

- Bounded model checking can be used also for checking invariants
- Invariants are checked via the `check_invar_bmc` command
- Invariants are checked via an inductive reasoning, i.e.
  BMC tries to prove that:
  - the property holds in every initial state
  - the property holds in every state reachable from any state where it holds

Checking invariants

- Consider the following example:

```
MODULE main
VAR
  out : 0..15;
ASSIGN
  init(out) := 0;
TRANS
  case
     out = 7 :  next(out) = 0;
     TRUE    :  next(out) = ((out + 1) mod 16);
  esac
INVARSPEC out in 0..10
INVARSPEC out in 0..7
```

Checking invariants

```
NuSMV > check_invar_bmc
-- cannot prove the invariant out in (0 .. 10) : the induction fails
-- as demonstrated by the following execution sequence
-> State 1.1 <-
    out = 10
-> State 1.2 <-
    out = 11
-- invariant out in (0 .. 7)   is true
```

- The invariant `out in 0..10` is true. However, the induction fails because a state in which out=11 can be reached from a state in which out=10
- If an invariant cannot be proved by inductive reasoning, it does not necessarily mean that the formula is false
- The stronger invariant `out in 0..7` is proved true by BMC, therefore also the invariant `out in 0..10` is true