SMV and nuXmV Quickstart

IA169: Model Checking

Martin Jonáš

Faculty of Informatics, Masaryk University

SMV

SMV

- a language for describing transition systems
- we use its extension that is used in a model checker NUXMV

Variables

Variable definitons

```
VAR
  name1 : boolean;
  name2 : {ready, waiting, stopped};
  name3 : 1..8;
```

Possible types

- `boolean` (values `TRUE` and `FALSE`)
- enumerative: e.g., { `ready, waiting, stopped` }
- bounded interval: e.g, `1..8`
- C integers: `integer`
- real numbers: `real`
- bit-vectors: `unsigned word[8]` or `signed word[16]` or ...

And more.

Operations

Boolean

- `&`, `|`, `xor`, `->`, `<->` (all infix)
- `!` (prefix)

Arithmetic

- `+`, `-`, `*`, `/`, `mod` (all infix)

Comparisons

- `=`, `!=`, `<`, `>`, `<=`, `>=` (all infix)

And more.

Initial values and successor values

```
ASSIGN
  init(var1) := TRUE;
  next(var1) := !var1;
  init(var2) := 0;
  next(var2) := var2 + var3;
  init(var3) := {ready, wating};
```

For each variable, specify its initial value and its new value based on the current values of the variables. Can specify a set of variables: nondeterminism.

If an initial value or the next value are not specified, they can be arbitrary (are unconstrained).

Initial and successor conditions

Alternatively

```
INIT
  var1 <-> TRUE;
INIT
  var2 >= 10;
INIT
  var3 != waiting;

TRANS
  var1 = !next(var1);
TRANS
  next(var4) >= var4 + var5;
```

Specify a set of formulas that has to hold in the initial states.

Specify a set of formulas that have to hold for transition (use `var` and `next(var)`).

Case expression

```
ASSIGN
  next(var1) :=
    case
      var2 = waiting : 10;
      var2 = ready : 12;
      TRUE : 42;
    esac;
```

Returns a body of the first condition that is true.

Can contain more complex formulas:

```
ASSIGN
  next(var1) :=
    case
      var2 = waiting & request & var3 > 12 : !var1;
      TRUE : var1;
    esac;
```

Fairness constrains

```
1 FAIRNESS (state = ready);
```

Restricts the runs to those where the formula `state = ready` holds infinitely often.

Equivalent to

```
1 JUSTICE (state = ready);
```

nuXmv

nuXmv

- a symbolic model checker for transition systems
- supports reachability, LTL, and CTL properties
- also supports infinite-state systems (will not be needed in this course)
- free only for non-commercial and academic usage

Running nuXmv

- `./nuXmv -it` for interactive usage
- `./nuXmv -source filename.cmd` for batch usage

Useful commands

- `read_model model.smv` to read a SMV model from a given file
- `go` to set up all internal data structures
- `pick_state -v -r` picks a random (`-r`) initial state and prints it (`-v`)
- `simulate -v -r -k 10` runs 10 steps (`-k`) of a simulation from the picked state, prints all the states (`-v`), chooses the successors randomly (`-r`)
- `reset` resets everything (if you want to use a new model)
- `check_ltlspec -p "ltl_formula"` checks whether the given LTL formula holds in the model
- `check_invar -p "property"` checks whether the given property holds in all reachable states of the model
- `help` lists all commands
- `help [command]` shows help for the given command
- `exit`