Automated Verification (CS6315)
Homework 3

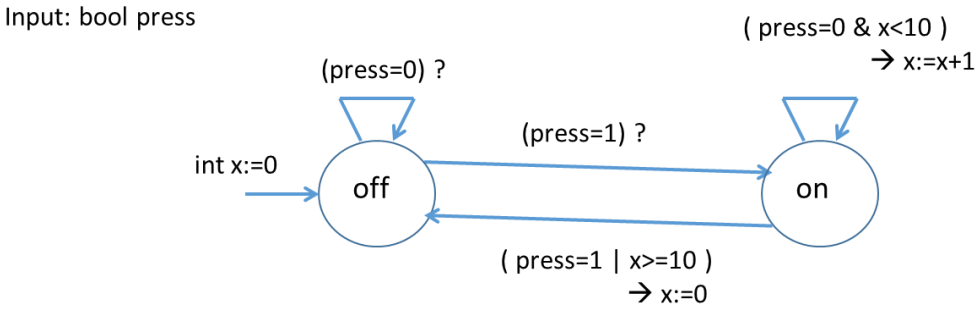
Due on Brightspace

Formal Models and Model Checking with Reachability

The corresponding chapters in the Alur textbook is primarily chapter 3.

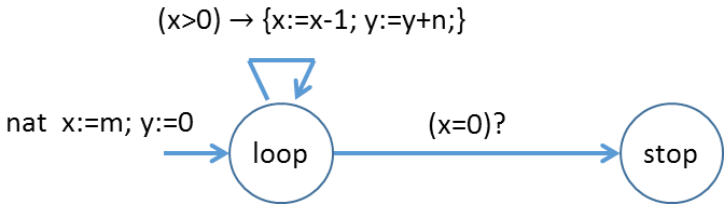
1. Alur, Exercise 3.10

Consider the reactive component `Switch` of figure 2.2 (and below). How many reachable states does it have? Draw or explain the reachable subgraph for the corresponding transition system. Hint: first, think about what is the overall state space of the system?



2. Alur, Exercise 3.13

Consider the transition system `Mult(m,n)` described in exercise 3.1 (and below). Describe this transition system symbolically using initialization and transition formulas. Hint: see Alur 3.4.1 for how this was done for other examples. This is similar to a problem from homework 2, but focusing on the symbolic transition system representation.



3. Alur, Exercise 3.14

Consider the description of the component `Switch` given as an extended state machine in figure 2.2 (and above in problem 1). Give the initialization and reaction formulas corresponding to Switch. Obtain the transition formula for the corresponding transition system in as simplified form as possible.

4. Alur, Exercise 3.16

Consider the symbolic image computation for a transition system with two real-valued variables x and y and a transition description given by the formula `x′=x+1 ∧ y′=x`. Suppose the region `A` is described by the formula `0 <= x <= 4 ∧ y <= 7`. Compute the formula describing the post-image (successors / post-states) of `A`.

5. Trying out Model Checking Tools

For this question, we will try some of the examples available with nuXmv. In future assignments, you may work to create and modify more sophisticated examples, but for now, we just want to make sure you can run existing examples before creating your own.

References: https://nuxmv.fbk.eu/
Manual: https://es.fbk.eu/tools/nuxmv/downloads/nuxmv-user-manual.pdf

This is described for Windows, if you want to run on Linux/Mac, see links above and follow installation instructions, it should be straightforward.

Some of the next commands are listed for the non-interactive modes of nuXmv, but of course you are welcome to use the interactive mode (-int) as we have primarily done in class, although if you do this, you'll need to look through the commands (help once in interactive mode).

a) Download the executable and examples for nuXmv in the file cs6315_hw03.zip from Brightspace for homework 3, then unzip the files. Alternatively, you may download the latest binaries from the nuXmv website above, but this archive on Brightspace also contains a specific example you'll look at (the counter/press example discussed in class).

b) Open a command prompt (press Windows + R, then type `cmd`, or go to a run prompt and type cmd).

c) Change directory to where you unzipped the examples, e.g.:

```
cd /D C:\eecs6315_hw03\
```

d) Execute nuXmv with the help option to see what's available:

```
nuXmv.exe -h
```

e) Execute nuXmv on the example discussed in lectures (note that you may have to use a different path depending on where you put the files):

```
nuXmv.exe C:\eecs6315_hw03\examples\counter.smv
```

f) Look through the lists of verified specifications and counterexample traces listed to get some understanding of what is going on and how to use the tool. To make it easier to navigate, you can pipe the output to a text file, e.g., with:

```
nuXmv.exe C:\eecs6315_hw03\examples\counter.smv > result.txt
```

g) Execute nuXmv on the example discussed in lecture with the following options, and ***write the number of total states and the number of states reachable (this answer is where most of the grading for this problem will come from and is just to illustrate you tried the tool)***. This usage relates directly back to several of the other questions in this assignment from the textbook; this gives an automated way now to answer how much of the state-space is reachable:

```
nuXmv.exe -r -ii -ils -is -ofm flat_out C:\eecs6315_hw03\examples\counter.smv
```

The option `-r` prints the reachable state information, `-ii` disables verifying INVARSPECs, `-ils` disables verifying LTLSPECs, `-is` disables verifying SPECs (CTLSPECs), and `-ofm` outputs a flattened model (the composition of the different modules/components describing the system). Take a look at the flat_out file (it should be in C:\eecs6315_hw03\ or whatever directory contains nuXmv.exe). Also, take a look at some of the other examples to get a feel for how the models and specifications are constructed. In future assignments, we will have you create models and write specifications, so certainly feel free to start modifying these to experiment with how things work.

Also, you can perform some basic correctness checking on the model to ensure it's well defined. This can be accomplished with the –ctt options, which ensures the transition relation is total:

```
nuXmv.exe -ctt examples\counter.smv
```

h) Next, open the model file counter.smv, and change the type definition of x in lines 6 to 9.

Then, execute the reachable state computation again, and ***write the number of states reachable in each of these scenarios***, where x is in 0..100, 0..1000, 0..10000, and 0..100000:

```
nuXmv.exe -r C:\eecs6315_hw03\examples\counter.smv
```

i) Finally, using the x range of 0..100, execute nuXmv in interactive mode and print the reachable states as follows:

```
nuXmv.exe -int C:\eecs6315_hw03\examples\counter.smv
```

This will open the nuXmv interactive terminal. The go command is a shorthand for the other commands we've used (flatten_hierarchy, encode_variables, build_model). You can use tab completion and type help to see commands. For this problem, you need to execute the following, but you can execute most commands with a –h option and it will show all usage details and options, etc. available. ***For the answer to this problem, include the output of the last command***, `print_reachable_states -v`.

```
nuXmv > go
nuXmv > print_reachable_states -h
usage: print_reachable_states [-h] [-v] [-d] [-f] [-o filename]
   -h           Prints the command usage.
   -v           Prints the list of reachable states.
   -d           Prints the list of reachable states with defines (Requires -v).
   -f           Prints the formula representing the reachable states.
   -o filename  Prints the result on the specified filename instead of on standard output
nuXmv > print_reachable_states -v
```

```
nuXmv > quit
```