Embedded Systems

Prof. Sedighi

Amirkabir University of Technology
(Tehran Polytechnic)

Department of Computer Engineering

Reza Adinepour ID: 402131055

Homework 10

Chapter 15 - Reachability Analysis and Model Checking

June 1, 2024

Embedded Systems

Homework 10

Reza Adinepour ID: 402131055

Department of Computer Engineering

Question 1

Consider the system M modeled by the hierarchical state machine of Figure 13.2, which models an interrupt-driven program.

Model M in the modeling language of a verification tool (such as SPIN). You will have to construct an environment model that asserts the interrupt. Use the verifica tion tool to check whether M satisfies ϕ, the property stated in Exercise 5:

$$\phi : \text{The main program eventually reaches program location C.}$$

Explain the output you obtain from the verification tool.

Soloution

To solve this exercise, various verification tools such as `PAT`, `Spin`, etc., can be used. According to the book's suggestion, Spin is used. This powerful tool has a large number of auxiliary packages that make it easier to use in different environments. Two examples of these tools are: `iSpin`[a], which is based on `TCL/TK`, and `jSpin`[b], which is developed using Java.

In the Spin environment, a language called Promela is used, and the user guide for it is available on the Spin[c] website. For this section, two processes are used: one for the `ISR` and one for the program inside the while loop. By running both simultaneously, it is assumed that an interrupt occurs in each section. The problem condition is checked with `timerCount == 0`, and if this happens, the program exits the loop related to the while section.

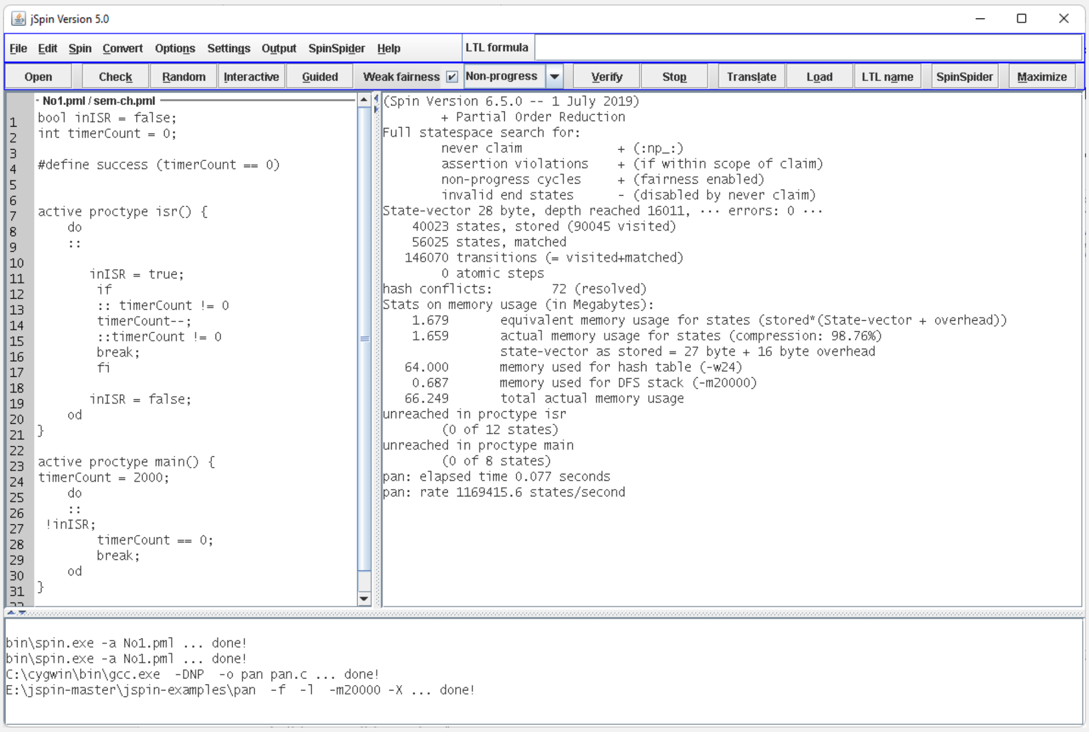
The results obtained from Verify and Check are as follows:

[a] http://spinroot.com/spin/Man/3_SpinGUI.html

[b] https://github.com/motib/jspin

[c] http://spinroot.com/spin/Man/Quick.html

Soloution


jSpin Version 5.0
File Edit Spin Convert Options Settings Output SpinSpider Help
LTL formula
Open Check Random Interactive Guided Weak fairness ☑ Non-progress Verify Stop Translate Load LTL name SpinSpider Maximize
No1.pml / sem-ch.pml

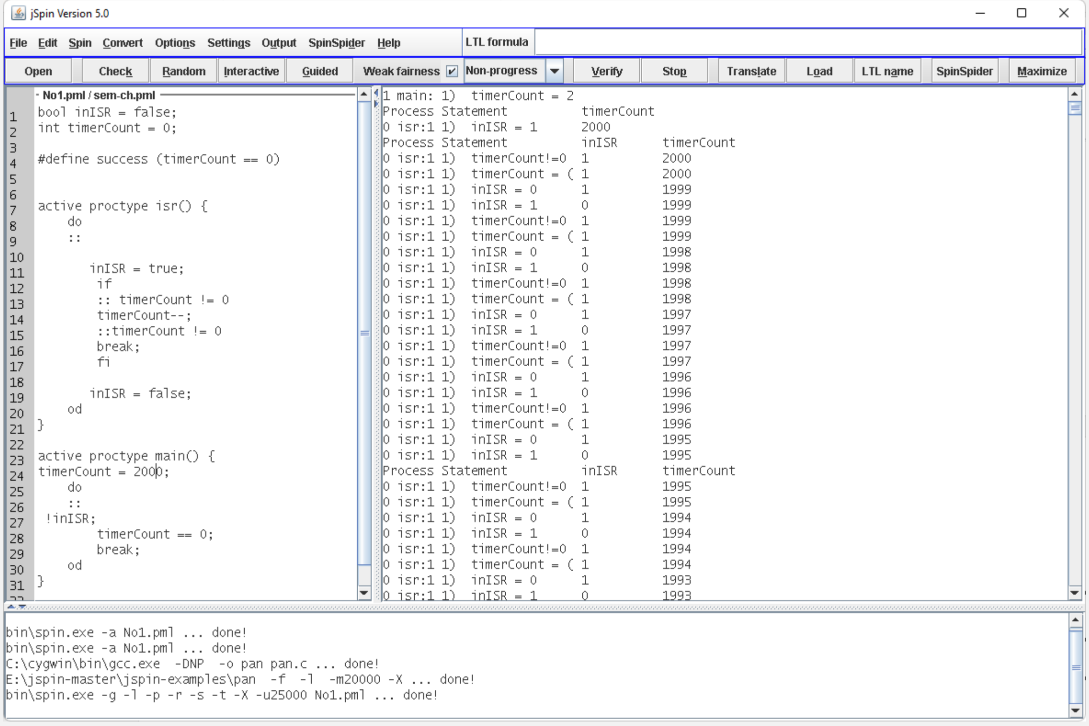
```
bool inISR = false;
int timerCount = 0;

#define success (timerCount == 0)


active proctype isr() {
    do
    ::

       inISR = true;
        if
        :: timerCount != 0
        timerCount--;
        ::timerCount != 0
        break;
        fi

       inISR = false;
    od
}

active proctype main() {
timerCount = 2000;
    do
    ::
 !inISR;
        timerCount == 0;
        break;
    od
}
```

```
(Spin Version 6.5.0 -- 1 July 2019)
        + Partial Order Reduction
Full statespace search for:
        never claim             + (:np_:)
        assertion violations    + (if within scope of claim)
        non-progress cycles     + (fairness enabled)
        invalid end states      - (disabled by never claim)
State-vector 28 byte, depth reached 16011, ··· errors: 0 ···
    40023 states, stored (90045 visited)
    56025 states, matched
   146070 transitions (= visited+matched)
        0 atomic steps
hash conflicts:        72 (resolved)
Stats on memory usage (in Megabytes):
    1.679       equivalent memory usage for states (stored*(State-vector + overhead))
    1.659       actual memory usage for states (compression: 98.76%)
                state-vector as stored = 27 byte + 16 byte overhead
   64.000       memory used for hash table (-w24)
    0.687       memory used for DFS stack (-m20000)
   66.249       total actual memory usage
unreached in proctype isr
        (0 of 12 states)
unreached in proctype main
        (0 of 8 states)
pan: elapsed time 0.077 seconds
pan: rate 1169415.6 states/second
```

```
bin\spin.exe -a No1.pml ... done!
bin\spin.exe -a No1.pml ... done!
C:\cygwin\bin\gcc.exe  -DNP  -o pan pan.c ... done!
E:\jspin-master\jspin-examples\pan  -f  -l  -m20000 -X ... done!
```

Figure 1: Output of `jSpin`

The simulation results are not fully included in the report due to the large number of checks for the variable `timerCount` and also `inISR` (to control access to the variable while the `ISR` is running).


jSpin Version 5.0
File Edit Spin Convert Options Settings Output SpinSpider Help
LTL formula
Open Check Random Interactive Guided Weak fairness ☑ Non-progress Verify Stop Translate Load LTL name SpinSpider Maximize
No1.pml / sem-ch.pml

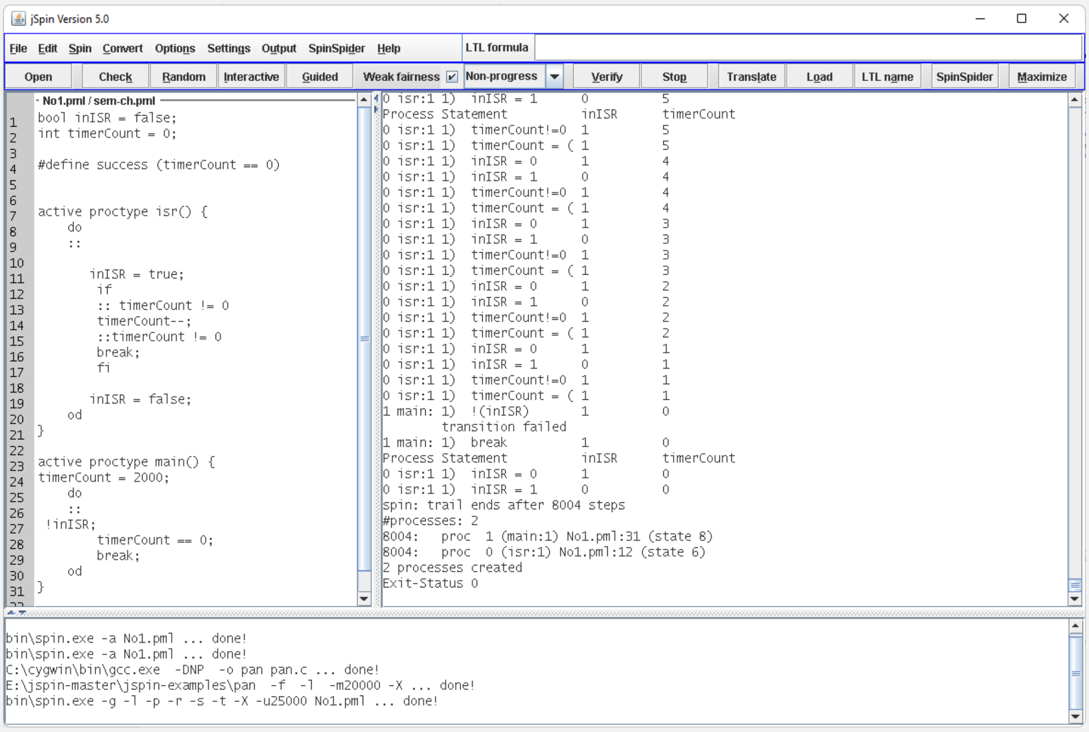
```
bool inISR = false;
int timerCount = 0;

#define success (timerCount == 0)


active proctype isr() {
    do
    ::

       inISR = true;
        if
        :: timerCount != 0
        timerCount--;
        ::timerCount != 0
        break;
        fi

       inISR = false;
    od
}

active proctype main() {
timerCount = 2000;
    do
    ::
 !inISR;
        timerCount == 0;
        break;
    od
}
```

```
1 main: 1)  timerCount = 2
Process Statement          timerCount
0 isr:1 1)  inISR = 1      2000
Process Statement          inISR      timerCount
0 isr:1 1)  timerCount!=0  1          2000
0 isr:1 1)  timerCount = ( 1          2000
0 isr:1 1)  inISR = 0      1          1999
0 isr:1 1)  inISR = 1      0          1999
0 isr:1 1)  timerCount!=0  1          1999
0 isr:1 1)  timerCount = ( 1          1999
0 isr:1 1)  inISR = 0      1          1998
0 isr:1 1)  inISR = 1      0          1998
0 isr:1 1)  timerCount!=0  1          1998
0 isr:1 1)  timerCount = ( 1          1998
0 isr:1 1)  inISR = 0      1          1997
0 isr:1 1)  inISR = 1      0          1997
0 isr:1 1)  timerCount!=0  1          1997
0 isr:1 1)  timerCount = ( 1          1997
0 isr:1 1)  inISR = 0      1          1996
0 isr:1 1)  inISR = 1      0          1996
0 isr:1 1)  timerCount!=0  1          1996
0 isr:1 1)  timerCount = ( 1          1996
0 isr:1 1)  inISR = 0      1          1995
0 isr:1 1)  inISR = 1      0          1995
Process Statement          inISR      timerCount
0 isr:1 1)  timerCount!=0  1          1995
0 isr:1 1)  timerCount = ( 1          1995
0 isr:1 1)  inISR = 0      1          1994
0 isr:1 1)  inISR = 1      0          1994
0 isr:1 1)  timerCount!=0  1          1994
0 isr:1 1)  timerCount = ( 1          1994
0 isr:1 1)  inISR = 0      1          1993
0 isr:1 1)  inISR = 1      0          1993
```

```
bin\spin.exe -a No1.pml ... done!
bin\spin.exe -a No1.pml ... done!
C:\cygwin\bin\gcc.exe  -DNP  -o pan pan.c ... done!
E:\jspin-master\jspin-examples\pan  -f  -l  -m20000 -X ... done!
bin\spin.exe -g -l -p -r -s -t -X -u25000 No1.pml ... done!
```

Figure 2: Output of `jSpin`

Soloution


jSpin Version 5.0
File Edit Spin Convert Options Settings Output SpinSpider Help
LTL formula
Open Check Random Interactive Guided Weak fairness ☑ Non-progress Verify Stop Translate Load LTL name SpinSpider Maximize
No1.pml / sem-ch.pml

```
bool inISR = false;
int timerCount = 0;

#define success (timerCount == 0)


active proctype isr() {
    do
    ::

       inISR = true;
        if
        :: timerCount != 0
        timerCount--;
        ::timerCount != 0
        break;
        fi

       inISR = false;
    od
}

active proctype main() {
timerCount = 2000;
    do
    ::
 !inISR;
        timerCount == 0;
        break;
    od
}
```

```
0 isr:1 1)  inISR = 1      0          5
Process Statement          inISR      timerCount
0 isr:1 1)  timerCount!=0  1          5
0 isr:1 1)  timerCount = ( 1          5
0 isr:1 1)  inISR = 0      1          4
0 isr:1 1)  inISR = 1      0          4
0 isr:1 1)  timerCount!=0  1          4
0 isr:1 1)  timerCount = ( 1          4
0 isr:1 1)  inISR = 0      1          3
0 isr:1 1)  inISR = 1      0          3
0 isr:1 1)  timerCount!=0  1          3
0 isr:1 1)  timerCount = ( 1          3
0 isr:1 1)  inISR = 0      1          2
0 isr:1 1)  inISR = 1      0          2
0 isr:1 1)  timerCount!=0  1          2
0 isr:1 1)  timerCount = ( 1          2
0 isr:1 1)  inISR = 0      1          1
0 isr:1 1)  inISR = 1      0          1
0 isr:1 1)  timerCount!=0  1          1
0 isr:1 1)  timerCount = ( 1          1
1 main: 1)  !(inISR)       1          0
        transition failed
1 main: 1)  break          1          0
Process Statement          inISR      timerCount
0 isr:1 1)  inISR = 0      1          0
0 isr:1 1)  inISR = 1      0          0
spin: trail ends after 8004 steps
#processes: 2
8004:   proc  1 (main:1) No1.pml:31 (state 8)
8004:   proc  0 (isr:1) No1.pml:12 (state 6)
2 processes created
Exit-Status 0
```

```
bin\spin.exe -a No1.pml ... done!
bin\spin.exe -a No1.pml ... done!
C:\cygwin\bin\gcc.exe  -DNP  -o pan pan.c ... done!
E:\jspin-master\jspin-examples\pan  -f  -l  -m20000 -X ... done!
bin\spin.exe -g -l -p -r -s -t -X -u25000 No1.pml ... done!
```

Figure 3: Output of `jSpin`

It should be noted that to use this tool on Windows, you must use the 32-bit version of Cygwin (even on a 64-bit operating system), otherwise, you will encounter a word size error message. Additionally, the Cygwin installation path must be added to the environment variables; otherwise, you will encounter a compiler access error.

Question 3

The notion of reachability has a nice symmetry. Instead of describing all states that are reachable from some initial state, it is just as easy to describe all states from which some state can be reached. Given a finite-state system M , the **backward reachable states** of a set F of states is the set B of all states from which some state in F can be reached. The following algorithm computes the set of backward reachable states for a given set of states F :

Algorithm 1 Backward Reachability

```
Require: A set $F$ of states and transition relation $\delta$ for closed finite-state system $M$
Ensure: Set $B$ of backward reachable states from $F$ in $M$
Initialize: $B := F$
$B_{\text{new}} := B$
while $B_{\text{new}} \neq \emptyset$ do
    $B_{\text{new}} := \{s \mid \exists s' \in B \text{ s.t. } s' \in \delta(s) \wedge s \notin B\}$
    $B := B \cup B_{\text{new}}$
end while
```

Explain how this algorithm can check the property $\mathbf{G}p$ on M , where p is some property that is easily checked for each state s in M . You may assume that M has exactly one initial state s_0 .

Soloution

This algorithm actually checks access to a set of states recursively by checking the access chain. (As explained in the problem statement, since the given pseudo-code seems incorrect in the initial assignment.) Now, if we want to check a case of Gp, according to LTL formulas, we can check $\neg F \neg p$. In this case, if we select a set of states where p is not true ($\neg p$) and the algorithm determines that these states are reachable, then the given condition is not met, and therefore the inverse is true. Of course, all these cases are correct provided that the interpretation of the problem statement is accurate.

End of Homework 10