SPIN INSTALLTION TUTORIAL

Fan Zhiyu

Abstract

This is a simple instruction to get SPIN setup for Mac/Linux/Windows systems. It includes three versions, which are Spin (for terminal user), jSpin, iSpin (for GUI user).

Table of Contents

I. Installations

In this section, we will present how to install SPIN model checker in Mac/Linux/Windows

A. *Spin*

1. Linux and Mac

You can download Spin binary files (`spin*_System*.gz`) from https://github.com/nimble-code/Spin/tree/master/Bin or Luminus (Files → SPIN Assignments → SPIN Binaries).
Commands to setup Spin

1. Unzip Spin from *.gz file: `gunzip *.gz`
2. Change the permission of spin file: `chmod u+x spin`
3. Modify the environment path: `export PATH=$PATH:<spin-path>`
4. Open terminal and check installation status: `spin -V`

2. Windows

1. Unzip Spin from *.gz file
2. Add `spin*_Windows*.exe` to System environment
3. Open Windows command line and check installation status: `spin -V`
4. Notice that you should install C compiler to use SPIN

If the terminal outputs `Spin Version *.*.* -- [Date]`, then it installed successfully

B. *jSpin*

You can download jSpin from Linux and Mac Downloads and Windows Download or get it from Luminus.

System environment prerequisites:

1. Spin (refer to I)
2. Java 1.5+
3. gcc compiler (Linux and Mac system should have this already)

4. brew (For mac only)

5. Dot (For Linux `sudo apt install graphviz`, For Mac `brew install graphviz`, For Windows, refer to this video)

Setup steps

1. Unzip jspin*.zip

2. Replace config.cfg line 11 to `SPIN=PATH_TO_SPIN`

3. Replace config.cfg line 17 to `SOURCE_DIRECTORY=WORKING_DIR`

4. Replace config.cfg line 26 to `C_COMPILER=PATH_TO_GCC`

5. Replace config.cfg line 28 to `DOT=PATH_TO_DOT`

6. Double click on `jSpin.jar` or run `java -jar jSpin.jar` to launch jSpin

C. iSpin

iSpin is written in tcl/tk. You can install tcl/tk by the following commands:

1. Linux: sudo apt install tk

2. Mac: brew install tcl-tk

3. Windows: Refer to this website Link

Execute `info tclversion` in `wish` (a Tcl/Tk command shell, type `wish` to enter) to check if you installed tcl/tk correctly.
Download iSpin from Luminus (Files → SPIN Assignments Description → SPIN Binaries → ispin).
Make the ispin.tcl executable, use `wish ispin.tcl` to launch ispin

II. Usage

In this section, we provide examples to show the basic usages of the three tools. Please download the examples from Luminus (Files → SPIN Assignments Description → SPIN Examples)

A. Spin

1. example-1.pml: Single Process

Command Line: `spin example-1.pml`
Output: state = 1

2. example-2.pml: Multiple Processes

Command Line: `spin example-2.pml`
Possible outputs:
1. state=1(from one A process)
state=1(from the other A process)
2. state=1(from one A process)
state=2(from the other A process)
3. state=1(from one A process)
timeout(from the other A process)

3. example-3.pml: Program Verification - Violation

Command Line:

```
spin -a example-3.pml
gcc -o pan pan.c
./pan -a
```

Output: assertion violated (sem!=2) (at depth 9)

4. example-4.pml: Program Verification - No Violation

Command Line:

```
spin -a example-4.pml
gcc -o pan pan.c
./pan -a
```

Output: No error

5. example-5.pml: Communication Channels

Command Line:

```
spin example-5.pml
```

Output: state=124
Command Line:

```
spin -M example-5.pml
```

Output: generate a message sequence chart like Figure 1

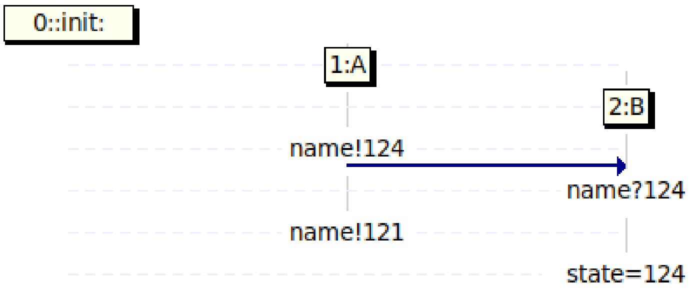


Figure 1: Sequence Chart generated by example-5.pml

6. example-6.pml: Mutual Exclusion - Semaphore

Command: spin -s example-6.pml
Output: endless loop. For each loop, the execution order is dijkstra_semaohore process followed by one of the users.

7. example-7.pml: Sender Receiver

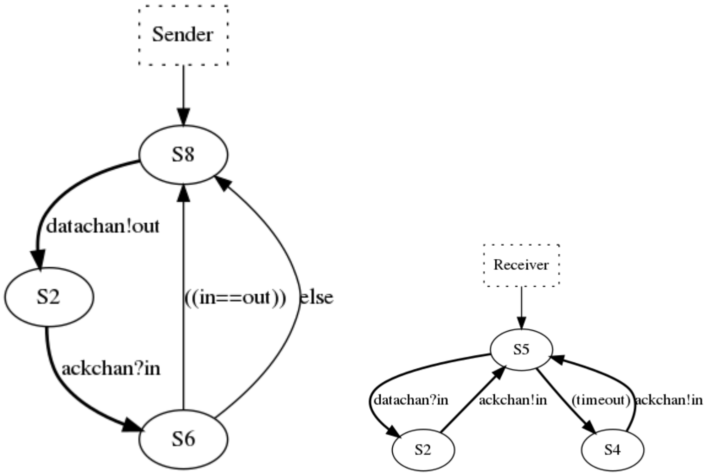
Command: `spin -s example-7.pml`
Output: endless loop. Send and Receiver are executed alternately
To generate the finite state machine for Sender and Receiver, run commands:

1. `spin -a example-7.pml`
2. `gcc -o pan pan.c`
3. `./pan -D > dot.tmp cat dot.tmp`

Output: Instructions for dot to generate the graphs. save each digraph separately (sender.dot and receiver.dot) and run the following command to generate graphs from the code.
Command: `dot -T png sender.dot -o sender.png`
You will get the graphs like Figure 2(a) and Figure 2(b):

8. example-8.pml: Handling Procedures - Factorial Function

Command: `spin example-8.pml`
Output: result is 3628800



(a) Sender (b) Receiver

Figure 2: Finite state machine for example-7.pml

9. example-9.pml: LTL Verification

Commands:

1. `spin -a example-9.pml`
2. `gcc -o pan pan.c`
3. `./pan -a -N p1`

Output: No error

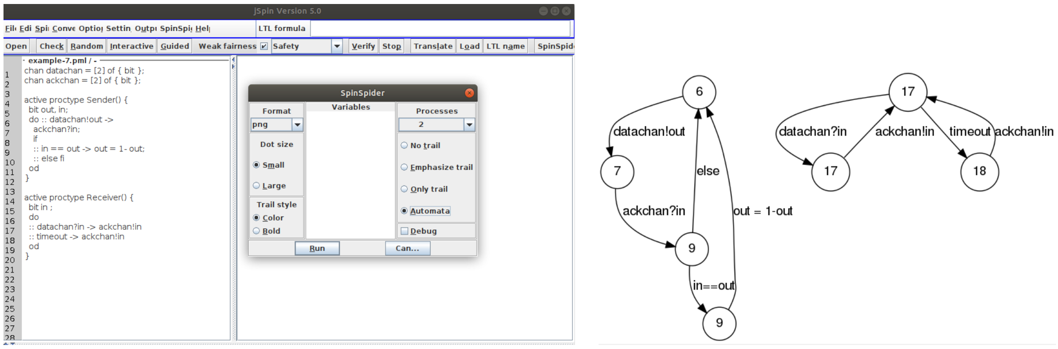
B. jSpin

Following examples will guide you to use jSpin as a GUI for Spin.

1. Finite State Machine

We will see how to use jSpin to generate the finite state machine for example-7.pml.

Open example-7.pml in jSpin, then click “SpinSpider”, in the appeared dialog box, select Automata option.



jSpin Version 5.0
File Edit Spin Convert Options Settings Output SpinSpider Help
LTL formula
Open Check Random Interactive Guided Weak fairness Safety Verify Stop Translate Load LTL name SpinSpider
example-7.pml / -
SpinSpider
Variables
Run
Can...

```
chan datachan = [2] of { bit };
chan ackchan = [2] of { bit };

active proctype Sender() {
 bit out, in;
 do :: datachan!out ->
   ackchan?in;
   if
   :: in == out -> out = 1- out;
   :: else fi
 od
}

active proctype Receiver() {
 bit in ;
 do
 :: datachan?in -> ackchan!in
 :: timeout -> ackchan!in
 od
}
```

Weak fairness ☑

Format: png

Dot size: ● Small ○ Large

Trail style: ● Color ○ Bold

Processes: 2

○ No trail ○ Emphasize trail ○ Only trail ● Automata

□ Debug

6
datachan!out
7
else
ackchan?in
out = 1-out
9
in==out
9
17
datachan?in
ackchan!in
timeout
ackchan!in
17
18

(a) UI (b) FSM

Figure 3: Finite State Machine

2. State Diagrams - All State

We will see how to use jSpin to generate the state diagrams for example-10.pml.

Open example-10.pml in jSpin, then click “SpinSpider”, in the appeared dialog box, select No trail option, also include the variable names under the “Variables” section.
Figure 4 shows the generated state diagram

C. iSpin

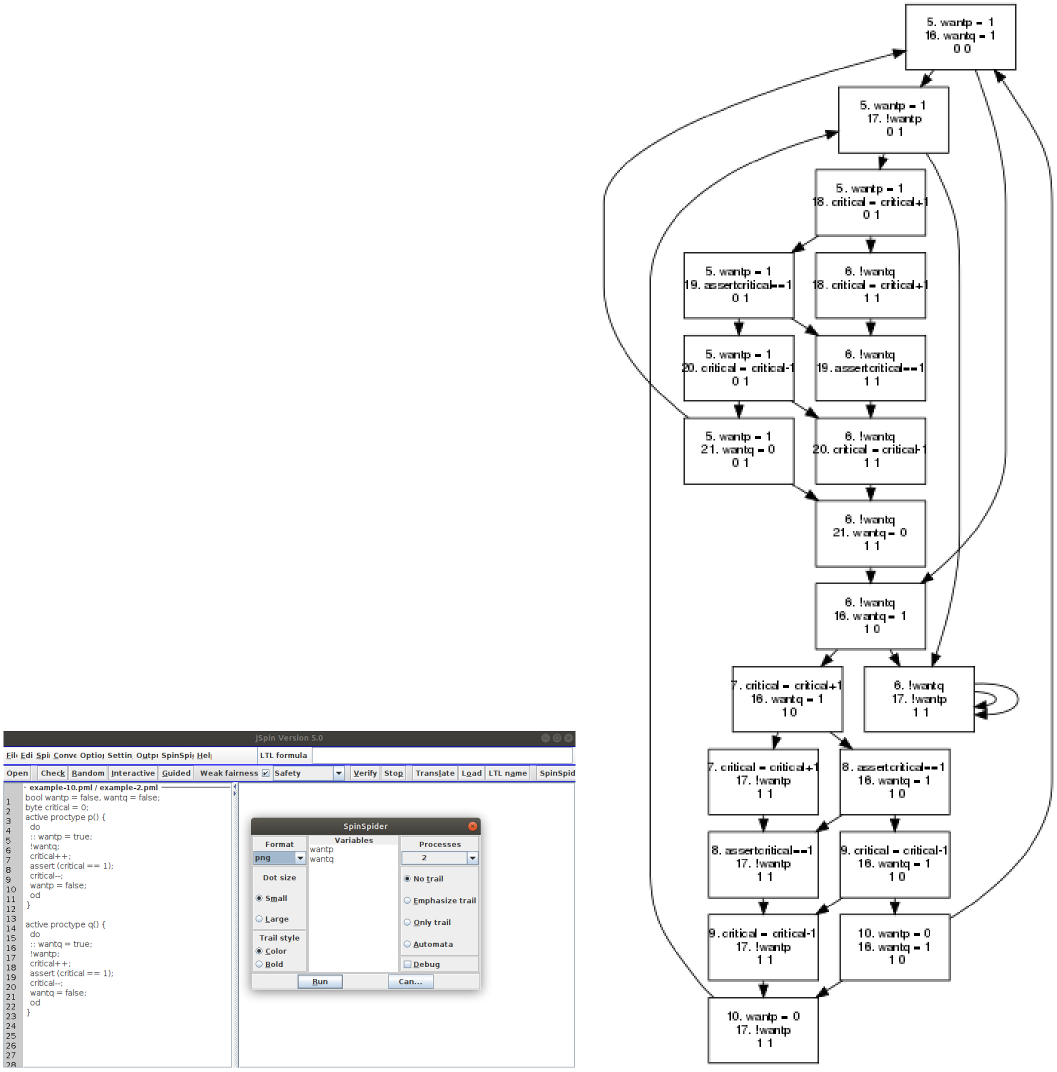
Following examples will guide you to use iSpin as a GUI for Spin.

1. Finite State Machine

We will see how to use iSpin to generate the finite state machine for example-7.pml.

Open example-7.pml through “Open” tab, then click “Automata View” on the right display panel which will generate FSM for each process. You need to select which FSM to render from the options given.

Click “p_Sender” ” and it will render the automata diagram for the p_Sender process as shown in Figure 5(b).



example-10.pml / example-2.pml

```
bool wantp = false, wantq = false;
byte critical = 0;
active proctype p() {
  do
  :: wantp = true;
  !wantq;
  critical++;
  assert (critical == 1);
  critical--;
  wantp = false;
  od
 }

active proctype q() {
  do
  :: wantq = true;
  !wantp;
  critical++;
  assert (critical == 1);
  critical--;
  wantq = false;
  od
 }
```

Weak fairness ☑ Safety

Format: png

Dot size: ● Small ○ Large

Trail style: ● Color ○ Bold

Variables: wantp wantq

Processes: 2

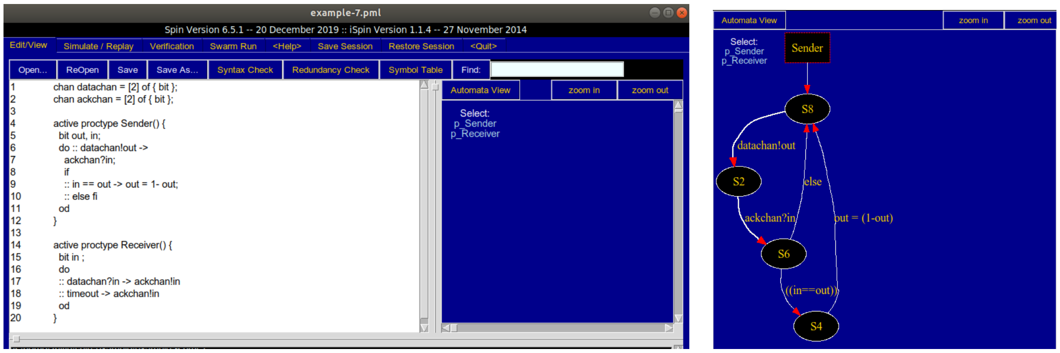
● No trail ○ Emphasize trail ○ Only trail ○ Automata

□ Debug

(a) UI

(b) FSM

Figure 4: State Diagram



```
chan datachan = [2] of { bit };
chan ackchan = [2] of { bit };

active proctype Sender() {
  bit out, in;
  do :: datachan!out ->
    ackchan?in;
    if
    :: in == out -> out = 1- out;
    :: else fi
  od
}

active proctype Receiver() {
  bit in ;
  do
  :: datachan?in -> ackchan!in
  :: timeout -> ackchan!in
  od
}
```

(a) UI

(b) FSM

Figure 5: Finite State Machine

2. Simulation Run

We will see how to use iSpin to run a simulation without using command line instructions for example-1.pml.

Open example-1.pml through “Open” tab, then Click on “Simulate/Replay” tab and click the “Run” button for simulation

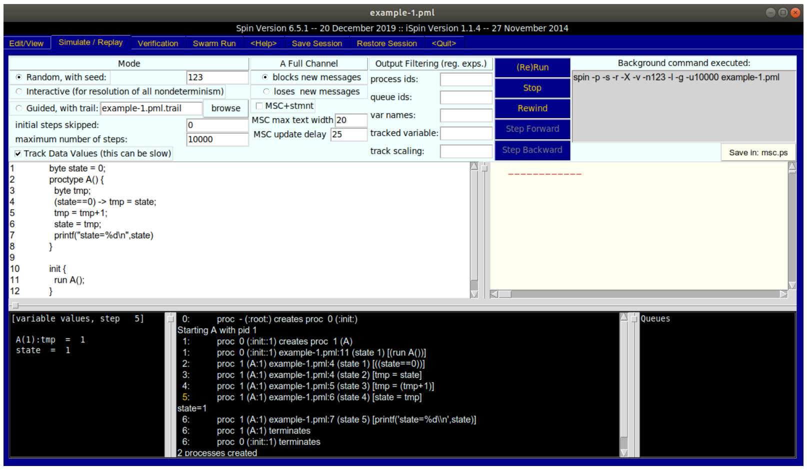
As shown in Figure 6, there are several outputs.

1. Log Window: At the upper right of the iSpin display shows which commands are being generated by iSpin and executed by Spin
2. Data Window: Lower left windows display the current values of variables
3. Channel Window: Lower right window display the values used in message passing
4. Simulation Output: Lower middle window shows the simulation log

3. Message Sequence Chart

We will see how to use iSpin to generate a message sequence chart without using command line instructions for example-6.pml.

Open example-6.pml through “Open” tab, then Click on “Simulate/Replay” tab and click the “Run” button for simulation, Click on “Stop” button to terminate the program.



Mode
- ● Random, with seed: 123
- ○ Interactive (for resolution of all nondeterminism)
- ○ Guided, with trail: example-1.pml.trail
- initial steps skipped: 0
- maximum number of steps: 10000
- ☑ Track Data Values (this can be slow)

A Full Channel
- ● blocks new messages
- ○ loses new messages
- □ MSC+stmnt
- MSC max text width 20
- MSC update delay 25

Output Filtering (reg. exps.)
- process ids:
- queue ids:
- var names:
- tracked variable:
- track scaling:

Background command executed:

```
spin -p -s -r -X -v -n123 -l -g -u10000 example-1.pml
```

```
byte state = 0;
proctype A() {
  byte tmp;
  (state==0) -> tmp = state;
  tmp = tmp+1;
  state = tmp;
  printf("state=%d\n",state)
}

init {
  run A();
}
```

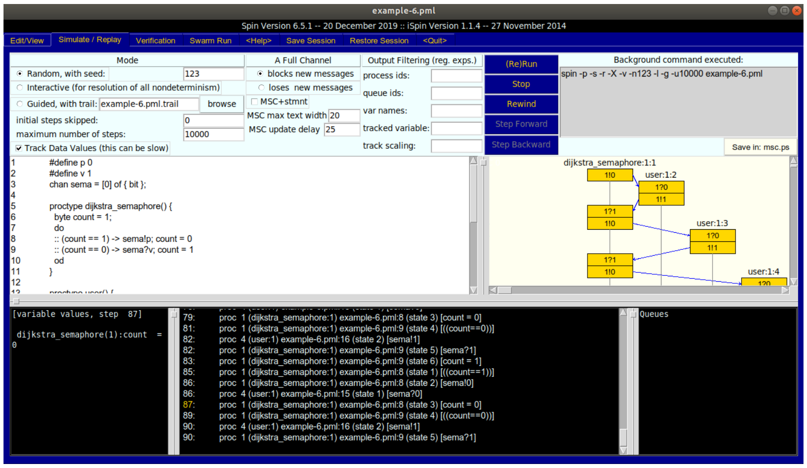
```
[variable values, step   5]

 A(1):tmp  =  1
 state  =  1
```

```
 0:      proc  - (:root:) creates proc  0 (:init:)
Starting A with pid 1
 1:      proc  0 (:init::1) creates proc  1 (A)
 1:      proc  0 (:init::1) example-1.pml:11 (state 1) [(run A())]
 2:      proc  1 (A:1) example-1.pml:4 (state 1) [((state==0))]
 3:      proc  1 (A:1) example-1.pml:4 (state 2) [tmp = state]
 4:      proc  1 (A:1) example-1.pml:5 (state 3) [tmp = (tmp+1)]
 5:      proc  1 (A:1) example-1.pml:6 (state 4) [state = tmp]
state=1
 6:      proc  1 (A:1) example-1.pml:7 (state 5) [printf('state=%d\\n',state)]
 6:      proc  1 (A:1) terminates
 6:      proc  0 (:init::1) terminates
2 processes created
```

Figure 6: Simulation Run for example-1.pml

As shown in the Figure 7, the display at middle right of the iSpin window generates the message sequence chart for the simulation we ran. You can save the diagram in PS format (which can be later used to produce png or pdf) by clicking the button "Save in: msc.ps"



Mode
- ● Random, with seed: 123
- ○ Interactive (for resolution of all nondeterminism)
- ○ Guided, with trail: example-6.pml.trail
- initial steps skipped: 0
- maximum number of steps: 10000
- ☑ Track Data Values (this can be slow)

A Full Channel
- ● blocks new messages
- ○ loses new messages
- □ MSC+stmnt
- MSC max text width 20
- MSC update delay 25

Output Filtering (reg. exps.)
- process ids:
- queue ids:
- var names:
- tracked variable:
- track scaling:

Background command executed:

```
spin -p -s -r -X -v -n123 -l -g -u10000 example-6.pml
```

```
#define p 0
#define v 1
chan sema = [0] of { bit };

proctype dijkstra_semaphore() {
  byte count = 1;
  do
  :: (count == 1) -> sema!p; count = 0
  :: (count == 0) -> sema?v; count = 1
  od
}

proctype user() {
```

```
[variable values, step  87]

 dijkstra_semaphore(1):count  =
0
```

```
79:      proc  1 (dijkstra_semaphore:1) example-6.pml:8 (state 3) [count = 0]
81:      proc  1 (dijkstra_semaphore:1) example-6.pml:9 (state 4) [((count==0))]
82:      proc  4 (user:1) example-6.pml:16 (state 2) [sema!1]
82:      proc  1 (dijkstra_semaphore:1) example-6.pml:9 (state 5) [sema?1]
83:      proc  1 (dijkstra_semaphore:1) example-6.pml:9 (state 6) [count = 1]
85:      proc  1 (dijkstra_semaphore:1) example-6.pml:8 (state 1) [((count==1))]
86:      proc  1 (dijkstra_semaphore:1) example-6.pml:8 (state 2) [sema!0]
86:      proc  4 (user:1) example-6.pml:15 (state 1) [sema?0]
87:      proc  1 (dijkstra_semaphore:1) example-6.pml:8 (state 3) [count = 0]
89:      proc  1 (dijkstra_semaphore:1) example-6.pml:9 (state 4) [((count==0))]
90:      proc  4 (user:1) example-6.pml:16 (state 2) [sema!1]
90:      proc  1 (dijkstra_semaphore:1) example-6.pml:9 (state 5) [sema?1]
```

Figure 7: Message Sequence Chart for example-6.pml