

SPIN

Simple Promela Interpreter

고급 소프트웨어공학

이종원, 이상진

2017.04.03

목차

- 01 Introduce of SPIN
- 02 Installation of SPIN
- 03 Using SPIN

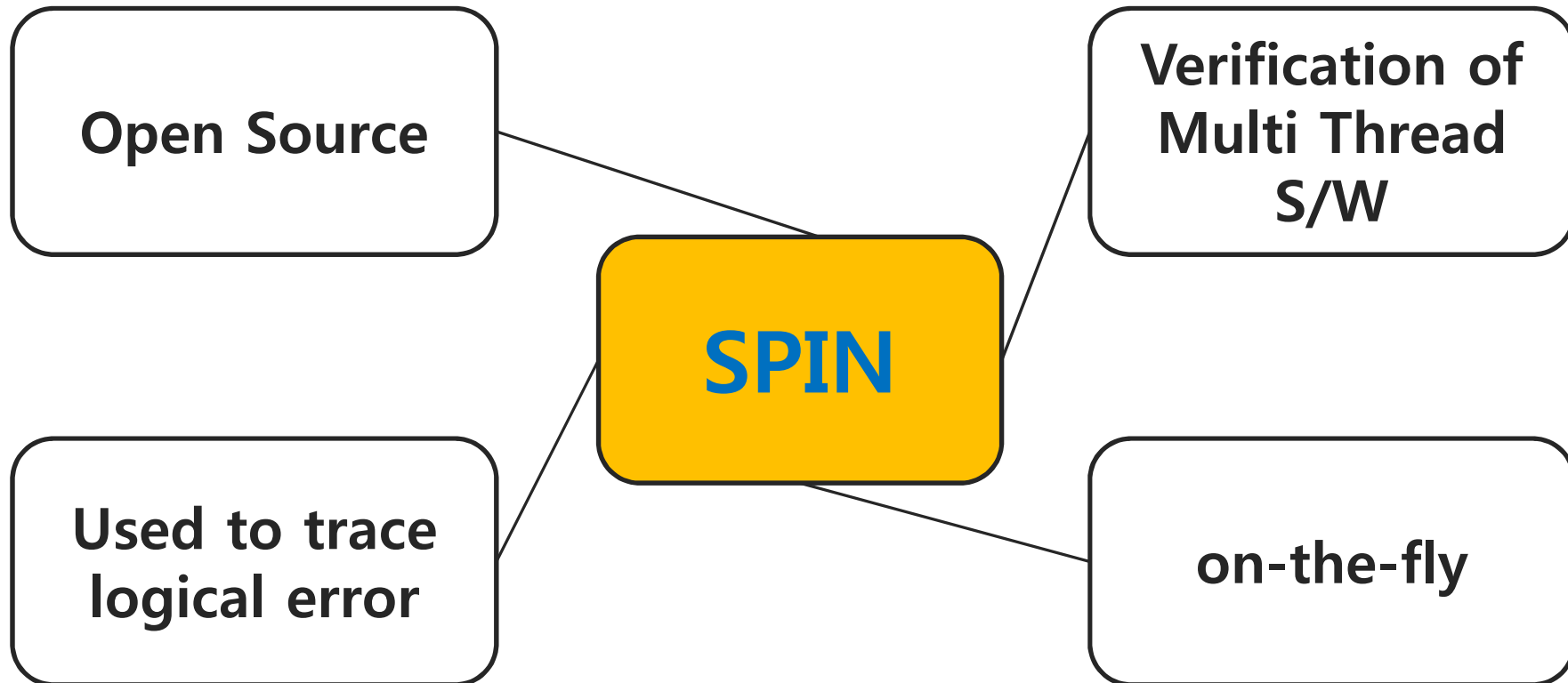


01

Introduce of SPIN

01. Introduce of SPIN

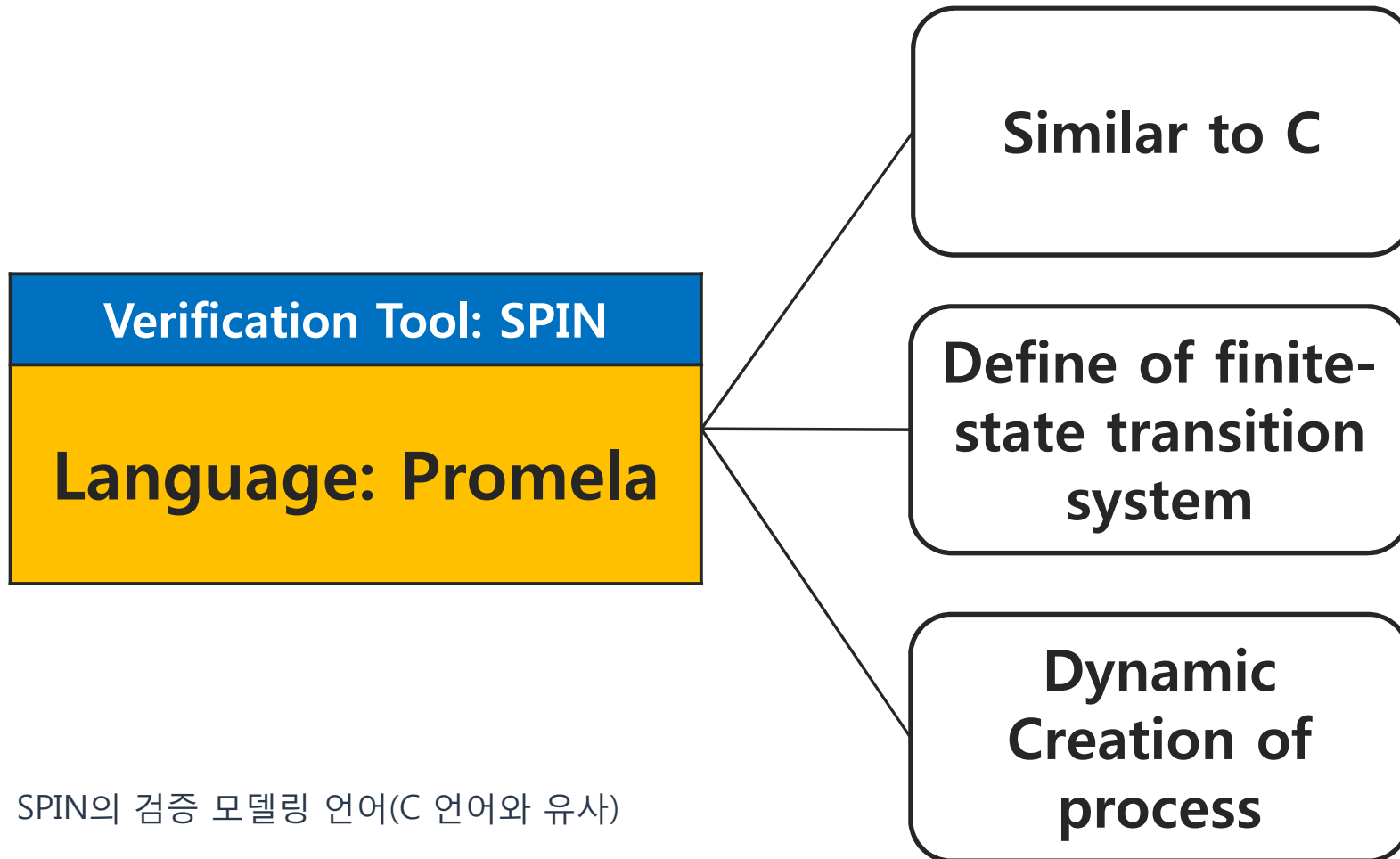
- **SPIN: Simple Promela Interpreter**



- ✓ 오픈 소스 기반으로, 멀티 쓰레드 소프트웨어의 검증을 위한 도구
 - ❖ (운영체제, 프로토콜, 분산·제어 소프트웨어 등의 시스템 설계 시) 논리적 오류를 추적
- ✓ 시스템 확인을 위한 조건을 사전 구성 불필요 ∴ 즉석으로 동작
- ✓ LTL 모델 검사 시스템으로도 사용되며, 명시적 상태의 모델을 효율적으로 검증!

01. Introduce of SPIN

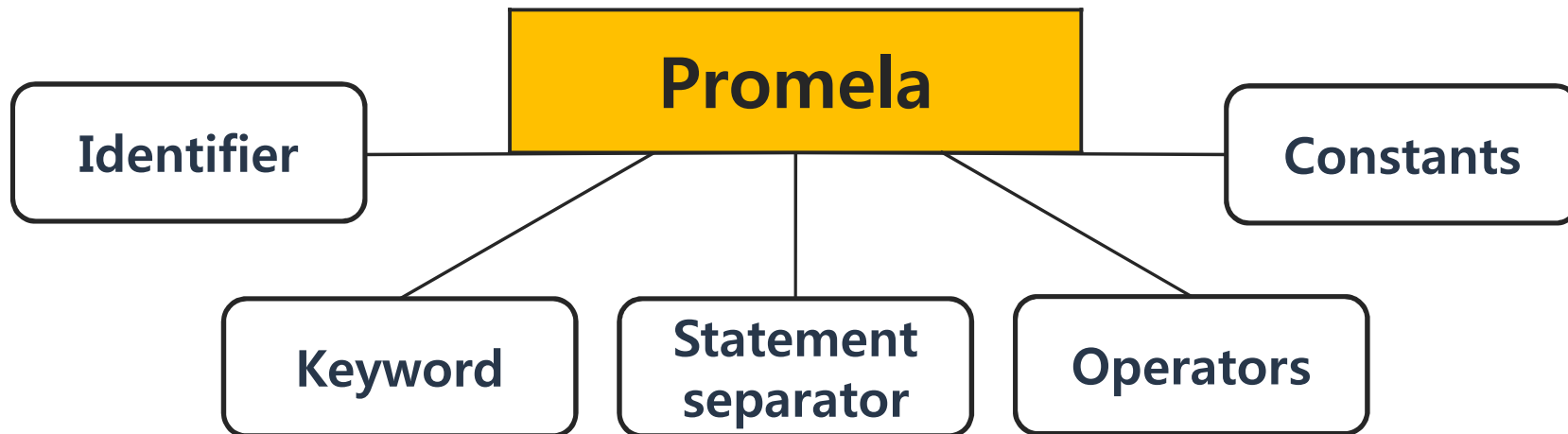
- **Promela: (Process·Protocol) Meta Language**



- ✓ SPIN의 검증 모델링 언어(C 언어와 유사)
- ✓ 유한 상태 시스템을 정의
- ✓ 프로세스들의 동적 생성 및 메시지 채널을 이용한 메시지의 비·동기 통신 모델링 가능

01. Introduce of SPIN

- **Promela: (Process·Protocol) Meta Language**



- ✓ 5개의 토큰으로 구성
 - ① 식별자 (Identifier): 단일 문자
 - ② 키워드 (Keyword): bool, goto, int, break, if, fi
 - ③ 상수 (Constants): #define NAME 5
 - ④ 연산자 (Operators): +, -, /, %, !=, ||
 - ⑤ 구문 분리 기호(Statement separator): ->, ;

02

Installation of SPIN

02. Installation of SPIN

- Process of installation



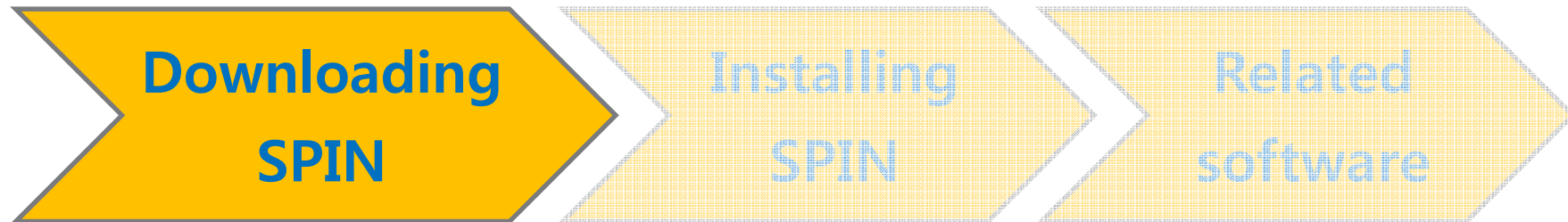
- Linux

- ✓ Oracle VM VirtualBox
- ✓ Ubuntu(32-bit)

```
inns@inns-VirtualBox: ~/Spin/iSpin
inns@inns-VirtualBox:~/Spin/iSpin$ grep . /etc/*-release
/etc/lsb-release:DISTRIB_ID=Ubuntu
/etc/lsb-release:DISTRIB_RELEASE=16.04
/etc/lsb-release:DISTRIB_CODENAME=xenial
/etc/lsb-release:DISTRIB_DESCRIPTION="Ubuntu 16.04.2 LTS"
/etc/os-release:NAME="Ubuntu"
/etc/os-release:VERSION="16.04.2 LTS (Xenial Xerus)"
/etc/os-release:ID=ubuntu
/etc/os-release:ID_LIKE=debian
/etc/os-release:PRETTY_NAME="Ubuntu 16.04.2 LTS"
/etc/os-release:VERSION_ID="16.04"
/etc/os-release:HOME_URL="http://www.ubuntu.com/"
/etc/os-release:SUPPORT_URL="http://help.ubuntu.com/"
/etc/os-release:BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
/etc/os-release:VERSION_CODENAME=xenial
/etc/os-release:UBUNTU_CODENAME=xenial
```


02. Installation of SPIN

- Process of installation



1. Downloading SPIN

① `$ wget http://spinroot.com/spin/Src/spin646.tar.gz`

```
inns@inns-VirtualBox:~$ sudo wget http://spinroot.com/spin/Src/spin646.tar.gz
[sudo] password for inns:
2017-04-02 16:33:15-- http://spinroot.com/spin/Src/spin646.tar.gz
Resolving spinroot.com (spinroot.com)... 104.28.11.21, 104.28.10.21, 2400:cb00:2048:1::681c:a15, ...
접속 spinroot.com (spinroot.com)|104.28.11.21|:80... 접속됨.
HTTP request sent, awaiting response... 200 OK
Content-Length: 673334 (658K) [application/x-gzip]
Saving to: 'spin646.tar.gz.1'

spin646.tar.gz.1          100%[=====] 657.55K  774KB/s  in 0.8s
2017-04-02 16:33:16 (774 KB/s) - 'spin646.tar.gz.1' saved [673334/673334]
```

02. Installation of SPIN

■ Process of installation



2. Installing SPIN

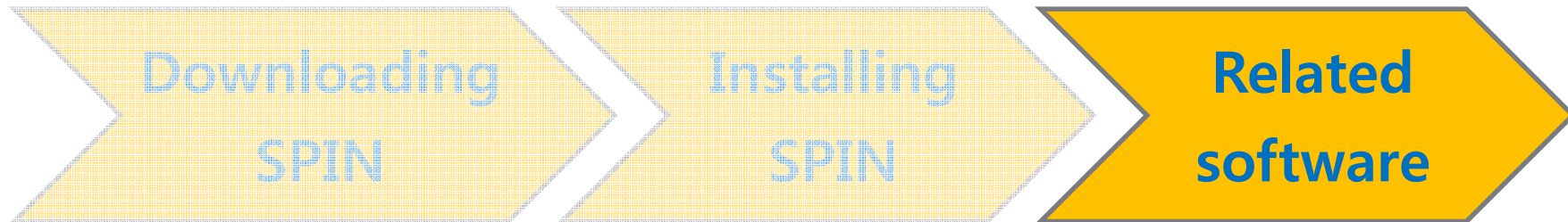
- ① `$ gunzip *.tar.gz`
- ② `$ tar -xf *.tar`
- ③ `$ cd Spin/Src*`
- ④ `$ apt-get install bison`
- ⑤ `$ make`

```
spin646.tar.gz
vitanin@vitanin-VirtualBox:~/spin_dir$ gunzip *.tar.gz
vitanin@vitanin-VirtualBox:~/spin_dir$ ls
spin646.tar
vitanin@vitanin-VirtualBox:~/spin_dir$ tar -xf *.tar
vitanin@vitanin-VirtualBox:~/spin_dir$ ls
ln spin646.tar
vitanin@vitanin-VirtualBox:~/spin_dir$ cd Spin/
vitanin@vitanin-VirtualBox:~/spin_dir/Spin$ ls
ac Examples Man Src6.4.6 lspin
vitanin@vitanin-VirtualBox:~/spin_dir/Spin$ cd Src*
vitanin@vitanin-VirtualBox:~/spin_dir/Spin/Src6.4.6$ ls
LICENSE main.c msg.c pangen2.c pangen4.c pangen6.c reprosrc.c spin.y tl.h tl_main.c tl_trans.c
step.c make32 msc_tcl.c pangen2.h pangen4.h pangen6.h run.c spinlex.c tl_bucht.c tl_main.c vars.c
low.c make_pc pangen1.c pangen3.c pangen5.c pangen7.c sched.c structs.c tl_cache.c tl_parse.c version.h
lded.c makefile pangen1.h pangen3.h pangen5.h pangen7.h spin.h syn.c tl_lex.c tl_rewrt.c
vitanin@vitanin-VirtualBox:~/spin_dir/Spin/Src6.4.6$ make
gcc -v -d spin.y
make: yacc: 명령을 찾지 못했음
makefile:48: 'spin.o' 타겟에 대한 명령이 실패했습니다

vitanin@vitanin-VirtualBox:~/spin_dir/Spin/Src6.4.6$ make
gcc -v -d spin.y
cc -O2 -DNXT -c y7tab.c
cc -f y7tab.c
cc -O2 -DNXT -c y7tab.o spin.o
cc -O2 -DNXT -c spinlex.o spinlex.c
cc -O2 -DNXT -c syn.o syn.c
cc -O2 -DNXT -c vars.o vars.c
cc -O2 -DNXT -c main.o main.c
cc -O2 -DNXT -c msc_tcl.o msc_tcl.c
cc -O2 -DNXT -c msg.o msg.c
cc -O2 -DNXT -c flow.o flow.c
cc -O2 -DNXT -c sched.o sched.c
cc -O2 -DNXT -c run.o run.c
cc -O2 -DNXT -c pangen1.o pangen1.c
cc -O2 -DNXT -c pangen2.o pangen2.c
cc -O2 -DNXT -c pangen3.o pangen3.c
cc -O2 -DNXT -c pangen4.o pangen4.c
cc -O2 -DNXT -c pangen5.o pangen5.c
cc -O2 -DNXT -c guided.o guided.c
cc -O2 -DNXT -c dstep.o dstep.c
cc -O2 -DNXT -c structs.o structs.c
cc -O2 -DNXT -c pangen6.o pangen6.c
cc -O2 -DNXT -c pangen7.o pangen7.c
cc -O2 -DNXT -c reprosrc.o reprosrc.c
cc -O2 -DNXT -c tl_parse.o tl_parse.c
cc -O2 -DNXT -c tl_lex.o tl_lex.c
cc -O2 -DNXT -c tl_main.o tl_main.c
cc -O2 -DNXT -c tl_trans.o tl_trans.c
cc -O2 -DNXT -c tl_bucht.o tl_bucht.c
```

02. Installation of SPIN

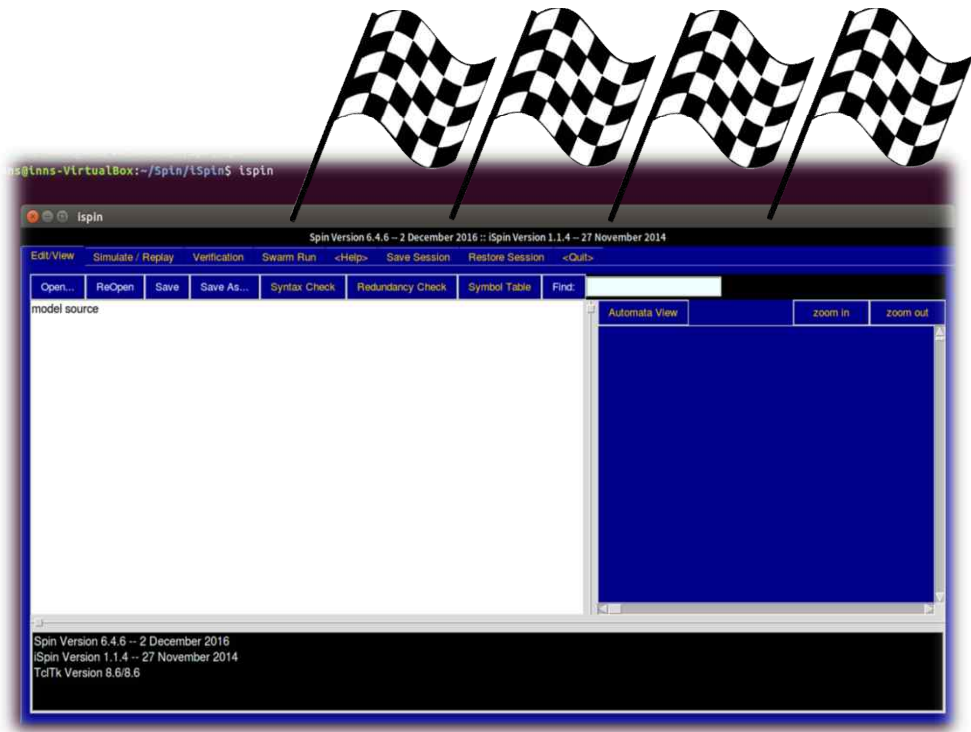
Process of installation



3. Related software

- ① \$ install.sh 실행
- ② \$ apt-get install graphviz

```
시 실행을 위한 중입니다... 완료
성 트리를 만드는 중입니다
이 정보를 읽는 중입니다... 완료
following additional packages will be installed:
libcdt5 libcgraph6 libgvc6 libgvpr2 libpathplan4
하는 패키지:
graphviz-doc
새 패키지를 설치할 것입니다:
graphviz libcdt5 libcgraph6 libgvc6 libgvpr2 libpathplan4
업그레이드, 6개 새로 설치, 0개 제거 및 239개 업그레이드 안 함.
35 k바이트 아카이브를 받아야 합니다.
작업 후 12.4 M바이트의 디스크 공간을 더 사용하게 됩니다.
하시겠습니까? [Y/n] y
:1 http://kr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libcdt5 amd64 2.38.0-12ubuntu2.1 [23.4 kB]
:2 http://kr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libcgraph6 amd64 2.38.0-12ubuntu2.1 [43.6 kB]
:3 http://kr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libpathplan4 amd64 2.38.0-12ubuntu2.1 [26.6 kB]
:4 http://kr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libgvc6 amd64 2.38.0-12ubuntu2.1 [591 kB]
:5 http://kr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libgvpr2 amd64 2.38.0-12ubuntu2.1 [176 kB]
:6 http://kr.archive.ubuntu.com/ubuntu xenial-updates/main amd64 graphviz amd64 2.38.0-12ubuntu2.1 [688 kB]
받기 1,535 k바이트, 소요시간 0초 (1,651 k바이트/초)
Selecting previously unselected package libcdt5.
이터베이스 읽는중 ... 현재 212085개의 파일과 디렉터리가 설치되어 있습니다.)
Preparing to unpack .../libcdt5_2.38.0-12ubuntu2.1_amd64.deb ...
Unpacking libcdt5 (2.38.0-12ubuntu2.1) ...
Selecting previously unselected package libcgraph6.
Preparing to unpack .../libcgraph6_2.38.0-12ubuntu2.1_amd64.deb ...
Unpacking libcgraph6 (2.38.0-12ubuntu2.1) ...
Selecting previously unselected package libpathplan4.
Preparing to unpack .../libpathplan4_2.38.0-12ubuntu2.1_amd64.deb ...
Unpacking libpathplan4 (2.38.0-12ubuntu2.1) ...
```

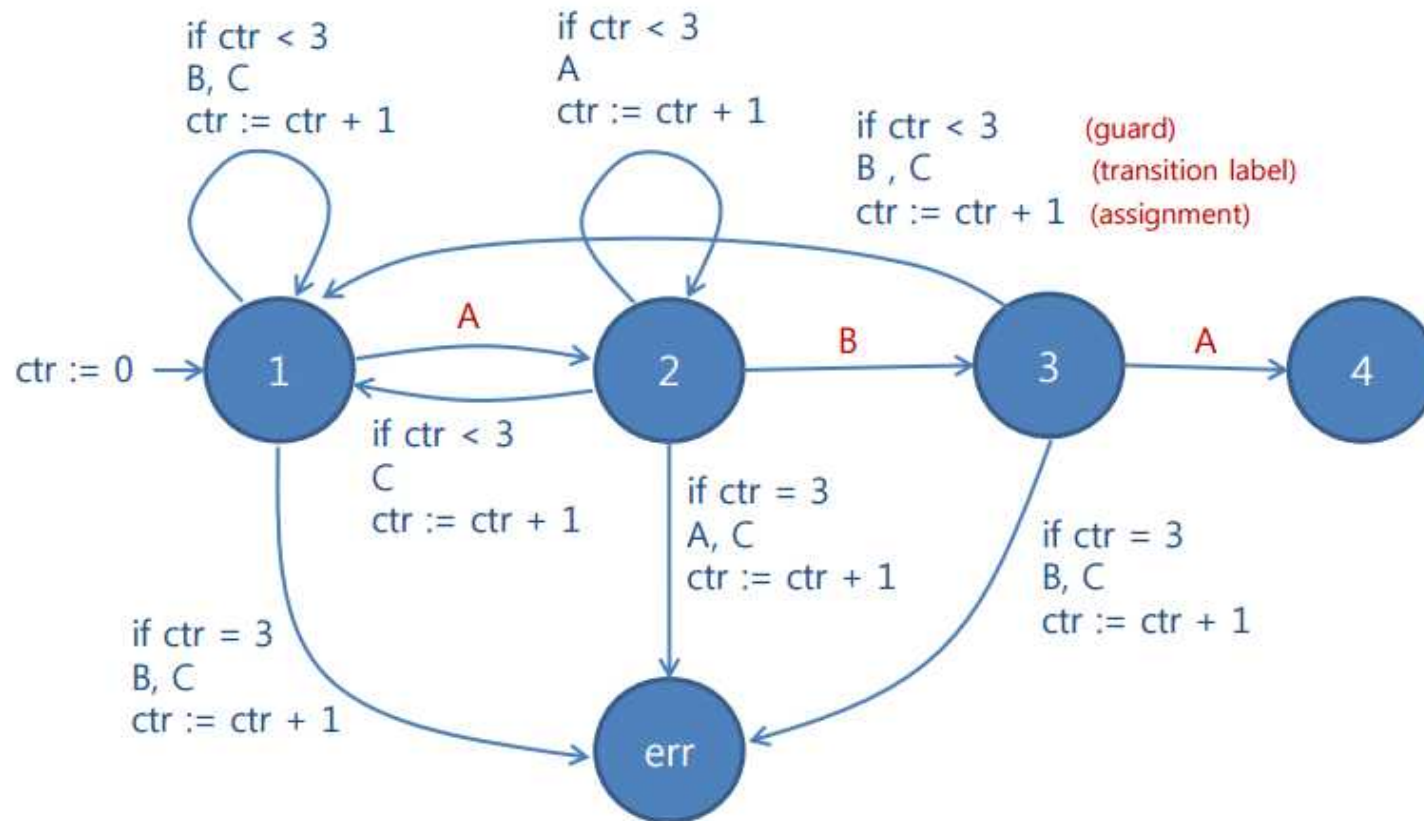


03

Using ISPIN

03. Using ISPIN

Verification of digital door lock



✓ Ref. 2017-1-Advanced Software Engineering-Lecture Note-19p

03. Using ISPIN

▪ Verification of digital door lock

✓ Trial and error

```
#define A 10
#define B 20
#define C 30

int ctr;
int order;
int password[3];
bool state;
mtype = {btn_A,btn_B,btn_C}
chan input_channel = [1] of {mtype}
chan door_channel = [1] of {mtype}

proctype user(){
  if
  ::(state==true)
    if
      ::input_channel!btn_A;
        door_channel?btn_A;
      ::input_channel!btn_B;
        door_channel?btn_B;
      ::input_channel!btn_C;
        door_channel?btn_C;
    fi;
  fi;
}
```

```
proctype press_button (int _order){
  do::
    if
      ::(door_channel!=password[_order-1])->
        order = 1;
        if
          ::(ctr<3)->
            ctr = ctr+1;
          ::(ctr==3)->
            state = false;
        fi;
      ::else->
        order = order+1;
    fi;
  od;
}

proctype matching(){
  order = 1;
  L1: if
    ::(state==true && order != 4)->
      run press_button(order);
    ::else->
      goto L1
  fi;
}
```

```
proctype result(){
  if
    ::(state==false)->
      printf("error");
    ::(order==4)->
      printf("open");
  fi;
}

init {
  ctr = 0;
  state = true;
  password[0] = A;
  password[1] = B;
  password[2] = A;

  run user();
  run matching();
  run result();
}
```

03. Using ISPIN

- Verification of digital door lock
- ✓ Trial and error

The screenshot displays the ISPIN software interface. The left pane shows a C++ code editor with the following code:

```
1 #define A 10
2 #define B 20
3 #define C 30
4
5 int ctr;
6 int order;
7 int password[3];
8 bool state;
9 mtype = {btn_A,btn_B,btn_C}
10 chan input_channel = [1] of {mtype}
11 chan door_channel = [1] of {mtype}
12
13 proctype user(){
14     if
15         ::(state==true)
16         if
17             ::input_channel!btn_A;
18             door_channel?btn_A;
19             ::input_channel!btn_B;
20             door_channel?btn_B;
21             ::input_channel!btn_C;
22             door_channel?btn_C;
23         fi;
24     fi;
25 }
26
27 proctype press_button (int _order){
28     do::
29         if
```

The right pane shows the Automata View window, which displays a state transition diagram. The states are S0, S1, S3, S6, and S8. The transitions are as follows:

- S0 to S1: matching
- S1 to S6: order = 1
- S6 to S3: ((state==1)&&(order!=4))
- S3 to S8: (run_press_button(order))
- S8 to S0: -end-

There is also a self-loop on S6 labeled "else".

03. Using ISPIN

Verification of digital door lock

✓ Code

```
int ctr;
bool door_open;
bool press_A;
bool press_B;
bool press_C;
mtype = {btn_A, btn_B, btn_C,unlock,error}

chan door_state = [1] of {mtype}
chan input = [1] of {mtype}

proctype person(){
  if
  ::input!btn_A->
    press_A = true;
    press_B = false;
    press_C = false;
    input?btn_A;
  ::input!btn_B->
    press_A = false;
    press_B = true;
    press_C = false;
    input?btn_B;
  ::input!btn_C->
    press_A = false;
    press_B = false;
    press_C = true;
    input?btn_C;
  fi;
}
```

```
proctype doorlock(){
  S1:
  if
  ::{input?btn_A} ->
    goto S2;
  ::(ctr < 3 && !press_A) ->
    ctr = ctr + 1;
    goto S1;
  ::(ctr == 3 && !press_A) ->
    ctr = ctr + 1;
    goto Err;
  fi;

  S2:
  if
  ::{input?btn_B} ->
    goto S3;
  ::(ctr < 3 && !press_B) ->
    ctr = ctr + 1;
    goto S1;
  ::(ctr == 3 && !press_B) ->
    ctr = ctr + 1;
    goto Err;
  fi;
}
```

```
S3:
  if
  ::{input?btn_A} ->
    goto S4;
  ::(ctr < 3 && !press_A) ->
    ctr = ctr + 1;
    goto S1;
  ::(ctr == 3 && !press_A) ->
    ctr = ctr + 1;
    goto Err;
  fi;

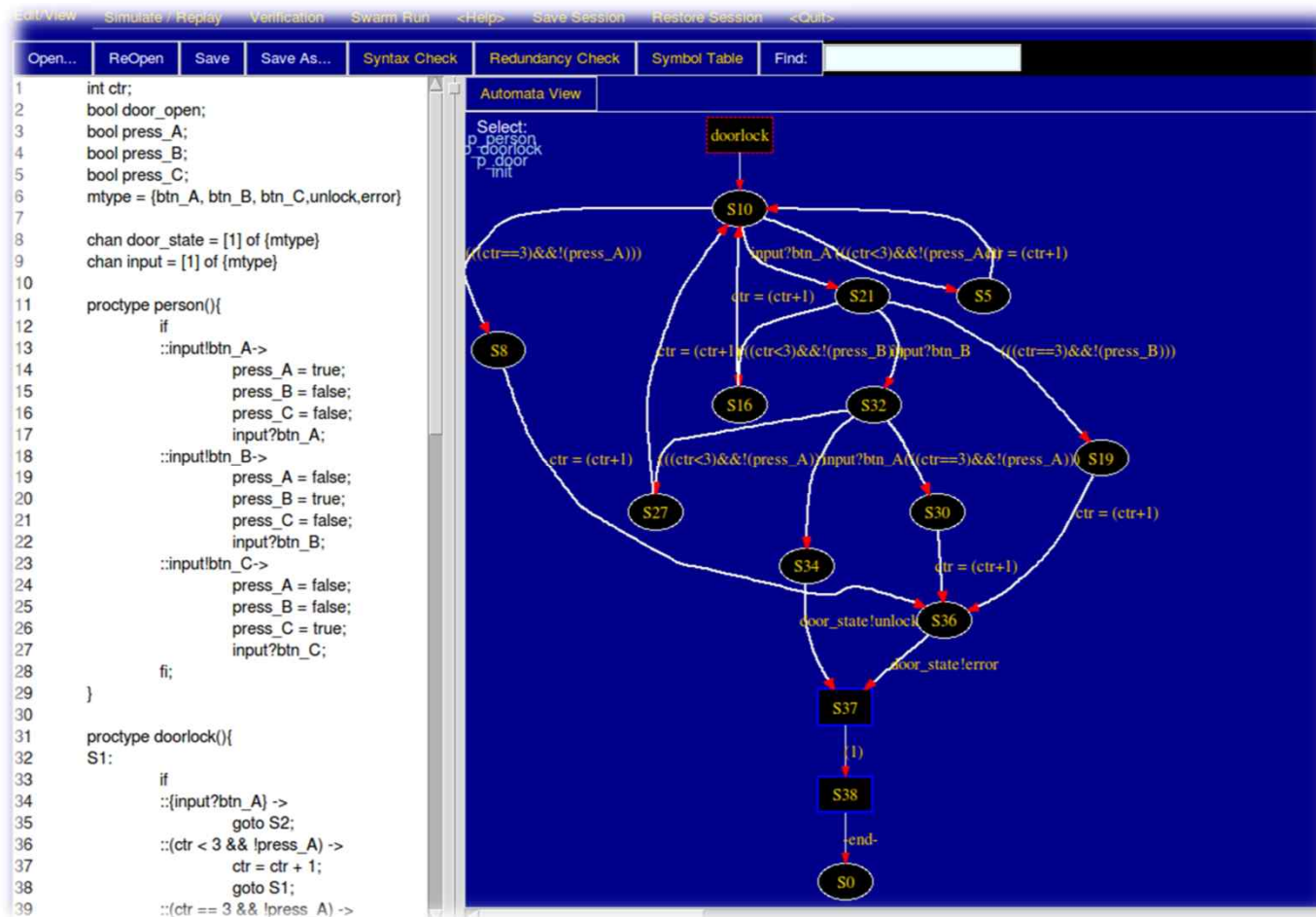
  S4:
  door_state!unlock;
  goto end;
  Err:
  door_state!error;
  end:
  skip;
}

proctype door(){
  if
  ::{door_state?unlock} -> door_open = true;
  ::{door_state?error} -> door_open = false;
  fi
}

init{
  ctr = 0;
  door_open = false;
  run person();
  run doorlock();
  run door();
}
```


03. Using ISPIN

- Verification of digital door lock
- ✓ Result - Automata view!



03. Using ISPIN

Verification of digital door lock

✓ Result – Simulate, Run!

The screenshot displays the ISPIN (Interactive Spin) software interface. The top menu bar includes options like 'Simulate / Replay', 'Verification', 'Swarm Run', and '<Help>'. Below the menu is a control panel with various simulation settings:

- Mode:** Random, with seed: 123
- A Full Channel:** blocks new messages (checked), loses new messages (unchecked), MSC+stmt (unchecked)
- Output Filtering (reg. exps.):** process ids, queue ids, var names, tracked variable, track scaling
- Buttons:** (Re)Run, Stop, Rewind, Step Forward, Step Backward
- Background command executed:** spin -p -s -r -X -v -n123 -l -g -u10000 doorlock01.pml
- Initial steps skipped:** 0
- Maximum number of steps:** 10000
- Track Data Values (this can be slow):** checked

The main workspace is divided into three panes:

- Left Pane (Code):** Contains the source code for the door lock simulation, including variable declarations and a type definition:

```
1 int ctr;
2 bool door_open;
3 bool press_A;
4 bool press_B;
5 bool press_C;
6 mtype = {btn_A, btn_B, btn_C, unlock, error}
```
- Right Pane (Sequence Diagram):** Shows a sequence diagram with participants 'person:1:1', 'doorlock:1:2', and 'door:1:3'. Messages include '1?btn_C', '1?btn_C', '2!error', and '2?error'.
- Bottom Pane (Log):** Displays the execution log, showing the state of processes and the sequence of events:

```
6: proc 2 (doorlock:1) doorlock01.pml:36 (state 4) [(((ctr<3)&&!press_A))]
7: proc 1 (person:1) doorlock01.pml:24 (state 12) [press_A = 0]
Starting door with pid 3
8: proc 0 (init::1) creates proc 3 (door)
8: proc 0 (init::1) doorlock01.pml:91 (state 5) [[run door()]]
9: proc 2 (doorlock:1) doorlock01.pml:37 (state 5) [ctr = (ctr+1)]
10: proc 1 (person:1) doorlock01.pml:25 (state 13) [press_B = 0]
11: proc 1 (person:1) doorlock01.pml:26 (state 14) [press_C = 1]
12: proc 2 (doorlock:1) doorlock01.pml:38 (state 6) [goto S1]
13: proc 1 (person:1) doorlock01.pml:27 (state 15) [input?btn_C]
15: proc 2 (doorlock:1) doorlock01.pml:36 (state 4) [(((ctr<3)&&!press_A))]
16: proc 2 (doorlock:1) doorlock01.pml:37 (state 5) [ctr = (ctr+1)]
17: proc 2 (doorlock:1) doorlock01.pml:38 (state 6) [goto S1]
18: proc 2 (doorlock:1) doorlock01.pml:36 (state 4) [(((ctr<3)&&!press_A))]
19: proc 2 (doorlock:1) doorlock01.pml:37 (state 5) [ctr = (ctr+1)]
20: proc 2 (doorlock:1) doorlock01.pml:38 (state 6) [goto S1]
21: proc 2 (doorlock:1) doorlock01.pml:39 (state 7) [(((ctr==3)&&!press_A))]
22: proc 2 (doorlock:1) doorlock01.pml:40 (state 8) [ctr = (ctr+1)]
23: proc 2 (doorlock:1) doorlock01.pml:41 (state 9) [goto Err]
24: proc 2 (doorlock:1) doorlock01.pml:73 (state 36) [door_state!error]
25: proc 2 (doorlock:1) doorlock01.pml:76 (state 37) [(1)]
26: proc 3 (door:1) doorlock01.pml:82 (state 4) [door_state?error]
27: proc 3 (door:1) doorlock01.pml:82 (state 6) [door_open = 0]
28: proc 3 (door:1) terminates
28: proc 2 (doorlock:1) terminates
28: proc 1 (person:1) terminates
28: proc 0 (init::1) terminates
4 processes created
```

?

Q&A