SPIN

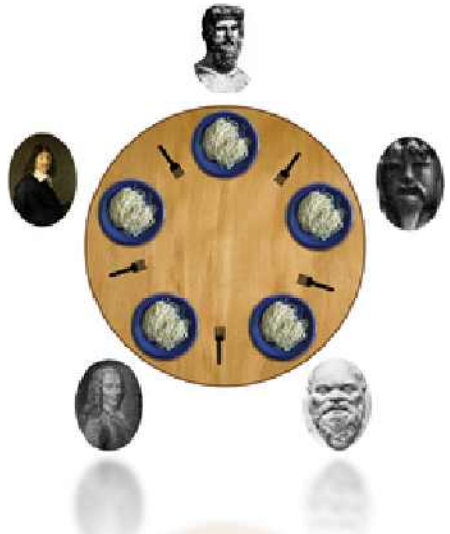
DATE 2014. 10. 10..

LAB DMS Lab

NAME Oh Jun

차례

1. 식사하는 철학자 문제



-Dining Philosophers

식사하는 철학자들 문제는 **<u>전산학에서 동시성과 교착 상태를 설명하는 예</u>**시로, 여러 프로세스가 동시에 돌아갈 때 교착 상태가 나타나는 원인을 직관적으로 알 수 있다.
다섯 명의 철학자가 원탁에 앉아 있고, 각자의 앞에는 스파게티가 있고 양 옆에 젓가락이 한 짝씩 있다. 그리고 각각의 철학자는 다른 철학자에게 말을 할 수 없다. 이때 철학자가 스파게티를 먹기 위해서는 양 옆의 젓가락 짝을 동시에 들고 있어야 한다. 이때 각각의 철학자가 왼쪽의 젓가락 짝을 들고 그 다음 오른쪽의 젓가락 짝을 들어서 스파게티를 먹는 알고리즘을 가지고 있으면, 다섯 철학자가 동시에 왼쪽의 젓가락 짝을 든 다음 오른쪽의 젓가락 짝을 들 때까지 무한정 기다리는 교착 상태에 빠지게 될 수 있다.
또한 어떤 경우에는 동시에 젓가락 양 짝을 집을 수 없어 **<u>식사를 하지 못하는 기아 상태가</u>** 발생할 수도 있고, 몇몇 철학자가 다른 철학자보다 식사를 적게 하는 경우가 발생하기도 한다.

2. SPIN

Spin 6.4.2 -- 8 October 2014

iSpin 1.1.3 – 27 September 2014

ActiveTcl8.6.1.0.297577-win32-x86_64-threaded

graphviz-2.38

Cygwin for 64-bit versions of Windows

Windows 7 Enterprise KN Service Pack 1 64bit

3. Promela code(1/3)

```
1
2.0         #define MAX_PHILOSOPHERS 5
2
3           mtype={fork}
4
5           #define left forks[my_id]
6           #define right forks[ (my_id+1) %MAX_PHILOSOPHERS ]
7           chan forks[ MAX_PHILOSOPHERS ] = [1] of {bit};
8           chan eatcount[MAX_PHILOSOPHERS] =[1] of {byte};
9
```

3. Promela code(2/3)

```
10.0    proctype philosopher(int my_id)
10      {
11
12                      bit ltest=0;
13                      bit rtest=0;
14
15                      do
16                              ::if
17                              ::ltest<1 -> left?fork;
18                                      printf("%d philo get left fork \n",my_id);
19                                      ltest =1;
20                                      if
21                                      ::rtest <1 ->
22                                              right?fork;
23                                              rtest=1;
24                                              printf("%d philo get two fork\n",my_id);
25                                              printf("%d philo eatting\n",my_id);
26                                              left!fork;
27                                              ltest=0;
28                                              printf("%d philo drop left fork after eat\n",my_id);
29                                              right!fork;
30                                              rtest=0;
31                                              printf("%d philo drop right fork after eat\n",my_id);
32                                      fi
33
34                              fi
35                      od
36
37      }
```

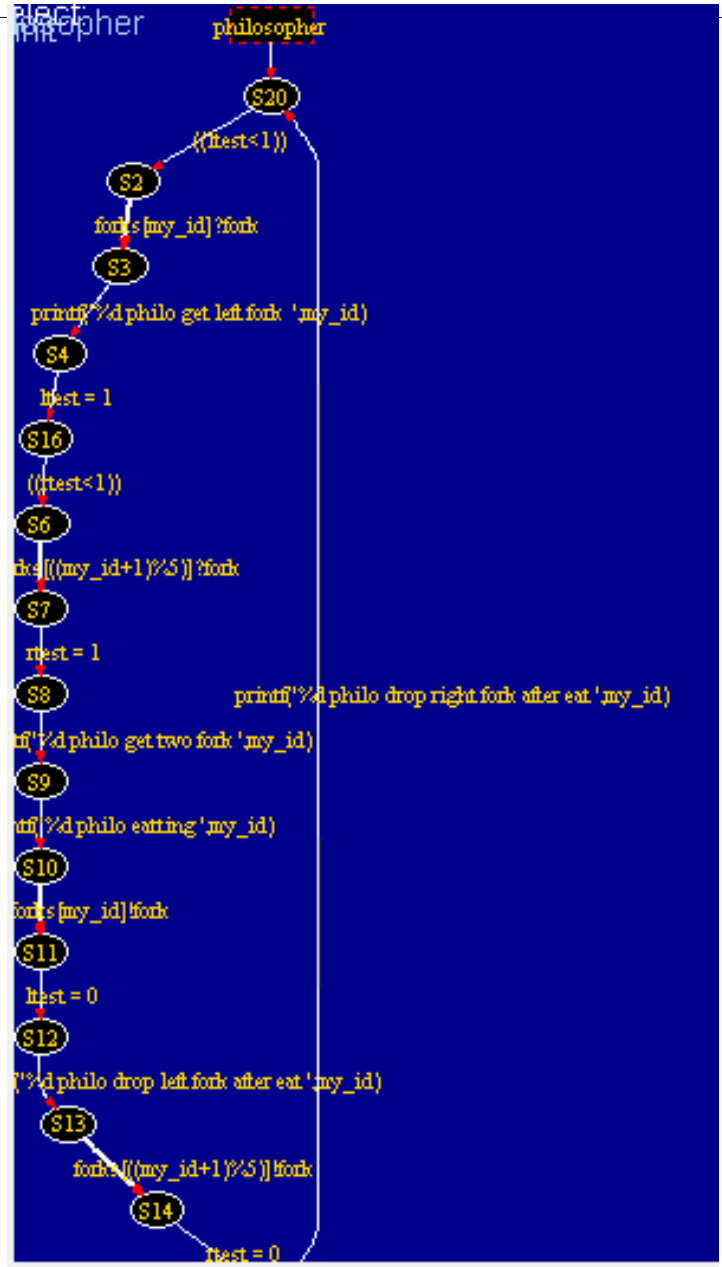
3. Promela code(3/3)

```
39          init
40          {
41                      byte philosophers=0;
42                                  atomic{
43                                  do
44                                  ::philosophers<MAX_PHILOSOPHERS->
45                                              forks[philosophers]!fork
46
47                                              run philosopher(philosophers);
48
49                                              philosophers = philosophers+1;
50                                  ::philosophers==MAX_PHILOSOPHERS->
51                                              printf("ready to start\n");
52                                              break
53                                  od
54                                  }
55
56          }
```

DMS LAB

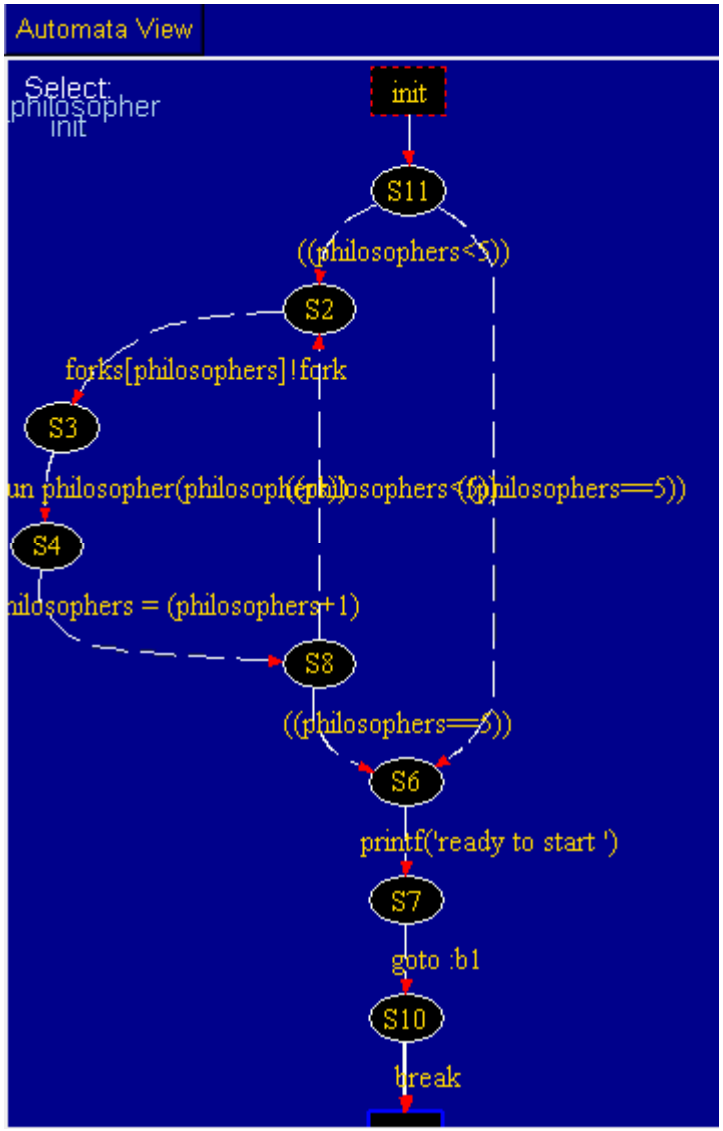
4. Automata View(1/2)

Proctype philosopher



4. Automata View(2/2)

init



5.Verification

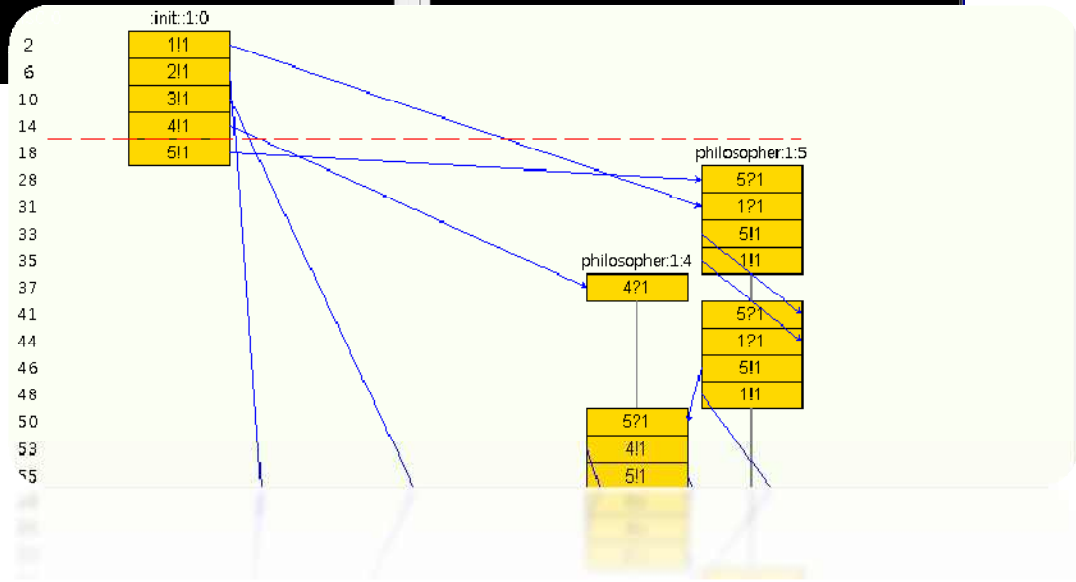
```
spin -a  MyDiningPhilosophers.pml
gcc -DMEMLIM=1024 -O2 -DXUSAFE -DSAFETY -DNOCLAIM -w -o pan pan.c
./pan -m10000  -c1
Pid: 5104
pan:1: invalid end state (at depth 581)
pan: wrote MyDiningPhilosophers.pml.trail

(Spin Version 6.4.2 -- 8 October 2014)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim         - (not selected)
        assertion violations +
        cycle checks        - (disabled by -DSAFETY)
        invalid end states +

State-vector 144 byte, depth reached 582, errors: 1
      562 states, stored
      260 states, matched
      822 transitions (= stored+matched)
       21 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.092   equivalent memory usage for states (stored*(State-vector + overhead))
    0.289   actual memory usage for states
  128.000 memory used for hash table (-w24)
    0.534   memory used for DFS stack (-m10000)
  128.730 total actual memory usage



pan: elapsed time 0.002 seconds
To replay the error-trail, goto Simulate/Replay and select "Run"
```

DMS LAB

6.Simulate(1/3)

```
 13:     proc  0 (:init::1) MyDiningPhilosophers.pml:44 (state 1) [((philosophers<5))]
 14:     proc  0 (:init::1) MyDiningPhilosophers.pml:45 (state 2) [forks[philosophers]!fork]
Starting philosopher with pid 4
 15:     proc  0 (:init::1) MyDiningPhilosophers.pml:47 (state 3) [(run philosopher(philosophers))]
 16:     proc  0 (:init::1) MyDiningPhilosophers.pml:49 (state 4) [philosophers = (philosophers+1)]
 17:     proc  0 (:init::1) MyDiningPhilosophers.pml:44 (state 1) [((philosophers<5))]
 18:     proc  0 (:init::1) MyDiningPhilosophers.pml:45 (state 2) [forks[philosophers]!fork]
Starting philosopher with pid 5
 19:     proc  0 (:init::1) MyDiningPhilosophers.pml:47 (state 3) [(run philosopher(philosophers))]
 20:     proc  0 (:init::1) MyDiningPhilosophers.pml:49 (state 4) [philosophers = (philosophers+1)]
 21:     proc  0 (:init::1) MyDiningPhilosophers.pml:50 (state 5) [((philosophers==5))]
ready to start
 21:     proc  0 (:init::1) MyDiningPhilosophers.pml:51 (state 6) [printf('ready to start\n')]
 22:     proc  0 (:init::1) MyDiningPhilosophers.pml:43 (state 10) [break]
 23:     proc  5 (philosopher:1) MyDiningPhilosophers.pml:17 (state 1) [((ltest<1))]
 24:     proc  4 (philosopher:1) MyDiningPhilosophers.pml:17 (state 1) [((ltest<1))]
 25:     proc  3 (philosopher:1) MyDiningPhilosophers.pml:17 (state 1) [((ltest<1))]
```

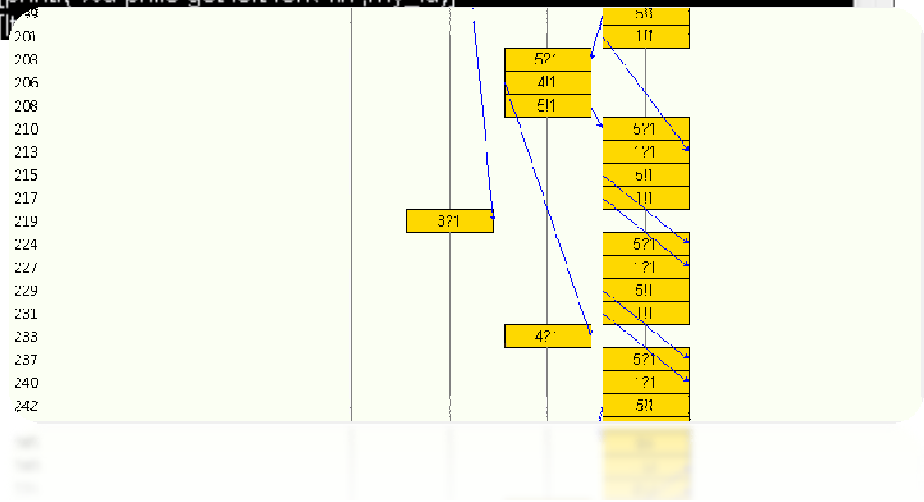
```
[queues, step  13]

q 1  :: (forks[0]): [1]
q 2  :: (forks[1]): [1]
q 3  :: (forks[2]): [1]
q 4  :: (forks[3]):
q 5  :: (forks[4]):
```



DMS LAB

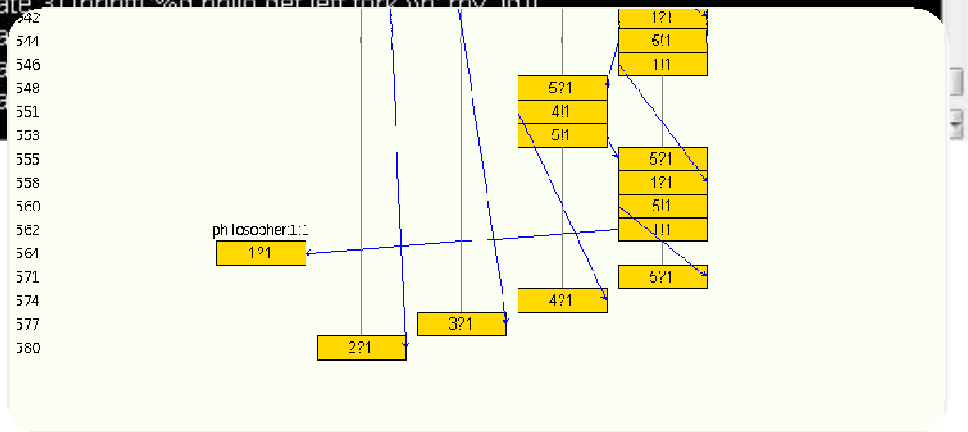
6.Simulate(2/3)

```
204:    proc  5 (philosopher:1) MyDiningPhilosophers.pml:17 (state 1) [((ltest<1))]
205:    proc  4 (philosopher:1) MyDiningPhilosophers.pml:23 (state 7) [rtest = 1]
3 philo get two fork
205:    proc  4 (philosopher:1) MyDiningPhilosophers.pml:24 (state 8) [printf('%d philo get two fork\\n',my_id)]
3 philo eatting
205:    proc  4 (philosopher:1) MyDiningPhilosophers.pml:25 (state 9) [printf('%d philo eatting\\n',my_id)]
206:    proc  4 (philosopher:1) MyDiningPhilosophers.pml:26 (state 10) [forks[my_id]!fork]
207:    proc  4 (philosopher:1) MyDiningPhilosophers.pml:27 (state 11) [ltest = 0]
3 philo drop left fork after eat
207:    proc  4 (philosopher:1) MyDiningPhilosophers.pml:28 (state 12) [printf('%d philo drop left fork after eat\\n',my_id)]
208:    proc  4 (philosopher:1) MyDiningPhilosophers.pml:29 (state 13) [forks[((my_id+1)%5)]!fork]
209:    proc  4 (philosopher:1) MyDiningPhilosophers.pml:30 (state 14) [rtest = 0]
3 philo drop right fork after eat
209:    proc  4 (philosopher:1) MyDiningPhilosophers.pml:31 (state 15) [printf('%d philo drop right fork after eat\\n',my_id)]
210:    proc  5 (philosopher:1) MyDiningPhilosophers.pml:17 (state 2) [forks[my_id]?fork]
4 philo get left fork
211:    proc  5 (philosopher:1) MyDiningPhilosophers.pml:18 (state 3) [printf('%d philo get left fork \\n',my_id)]
211:    proc  5 (philosopher:1) MyDiningPhilosophers.pml:19 (state 4) [
```



6.Simulate(3/3)

```
4 philo drop right fork after eat
563:        proc  5 (philosopher:1) MyDiningPhilosophers.pml:31 (state 15) [printf('%d philo drop right fork after eat\\n',my_id)]
564:        proc  1 (philosopher:1) MyDiningPhilosophers.pml:17 (state 2) [forks[my_id]?fork]
565:        proc  5 (philosopher:1) MyDiningPhilosophers.pml:17 (state 1) [((ltest<1))]
566:        proc  4 (philosopher:1) MyDiningPhilosophers.pml:17 (state 1) [((ltest<1))]
567:        proc  3 (philosopher:1) MyDiningPhilosophers.pml:17 (state 1) [((ltest<1))]
568:        proc  2 (philosopher:1) MyDiningPhilosophers.pml:17 (state 1) [((ltest<1))]
0 philo get left fork
569:        proc  1 (philosopher:1) MyDiningPhilosophers.pml:18 (state 3) [printf('%d philo get left fork \\n',my_id)]
569:        proc  1 (philosopher:1) MyDiningPhilosophers.pml:19 (state 4) [ltest = 1]
570:        proc  1 (philosopher:1) MyDiningPhilosophers.pml:21 (state 5) [((rtest<1))]
571:        proc  5 (philosopher:1) MyDiningPhilosophers.pml:17 (state 2) [forks[my_id]?fork]
4 philo get left fork
572:        proc  5 (philosopher:1) MyDiningPhilosophers.pml:18 (state 3) [printf('%d philo get left fork \\n',my_id)]
572:        proc  5 (philosopher:1) MyDiningPhilosophers.pml:19 (state 4) [ltest = 1]
573:        proc  5 (philosopher:1) MyDiningPhilosophers.pml:21 (state 5) [((rtest<1))]
574:        proc  4 (philosopher:1) MyDiningPhilosophers.pml:17 (state 2) [forks[my_id]?fork]
3 philo get left fork
575:        proc  4 (philosopher:1) MyDiningPhilosophers.pml:18 (state 3) [printf('%d philo get left fork \\n',my_id)]
575:        proc  4 (philosopher:1) MyDiningPhilosophers.pml:19 (sta
576:        proc  4 (philosopher:1) MyDiningPhilosophers.pml:21 (sta
577:        proc  3 (philosopher:1) MyDiningPhilosophers.pml:17 (sta
2 philo get left fork
```



The End