

SE LAB 5

SAHIL BONDRE: U18CO021

1. There are four philosophers sitting around a round table. There are forks on the table, one between each pair of philosophers. The philosophers want to eat spaghetti from a large bowl in the center of the table. Unfortunately the spaghetti is of a particularly slippery type, and a philosopher needs both forks in order to eat it. The philosophers have agreed on the following protocol to obtain the forks: Initially philosophers think about philosophy, when they get hungry they do the following:

- Take the left fork
- Take the right fork and start eating
- Return both forks simultaneously, and repeat from the beginning.

Build a SPIN model for this scenario.

```
int SIZE = 4;
int FORKS[SIZE];

init {
    FORKS[0] = 0
    FORKS[1] = 0
    FORKS[2] = 0
    FORKS[3] = 0

    run philosopher(0);
    run philosopher(1);
    run philosopher(2);
```

```

run philosopher(3);
}

proctype philosopher(int i) {

    int low = i, high = (i + 1) % SIZE;

    // Graph based ordering for FORK allocation

    if

    :: (i == SIZE - 1) -> low = (i + 1) % SIZE; high = i;

    :: else low = i; high = (i + 1) % SIZE;

    fi

    do

    :: printf("Philosopher %d is thinking...\n", i + 1);

    (FORKS[low] == 0) -> FORKS[low] = 1; printf("Philosopher %d has picked fork
%d...\n", i + 1, low + 1);

    (FORKS[high] == 0) -> FORKS[high] = 1; printf("Philosopher %d has picked
fork %d...\n", i + 1, high + 1);

    printf("Philosopher %d is eating...\n", i + 1);

    FORKS[low] = 0;

    FORKS[high] = 0;

    printf("Philosopher %d done eating...\n", i + 1);

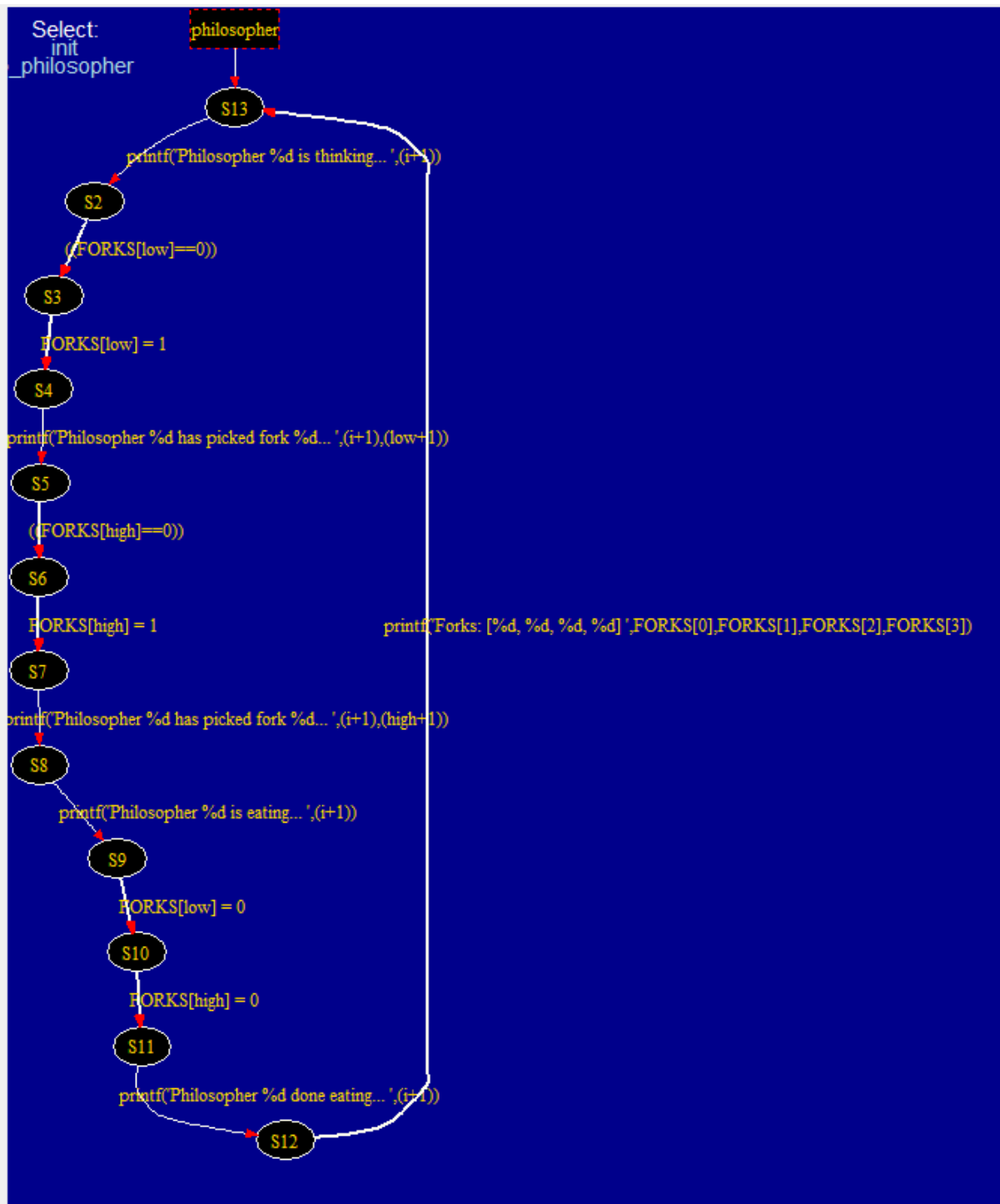
    printf("Forks: [%d, %d, %d, %d]\n", FORKS[0], FORKS[1], FORKS[2], FORKS[3])

    od

}

```

Without Graph-Based Allocation:



```

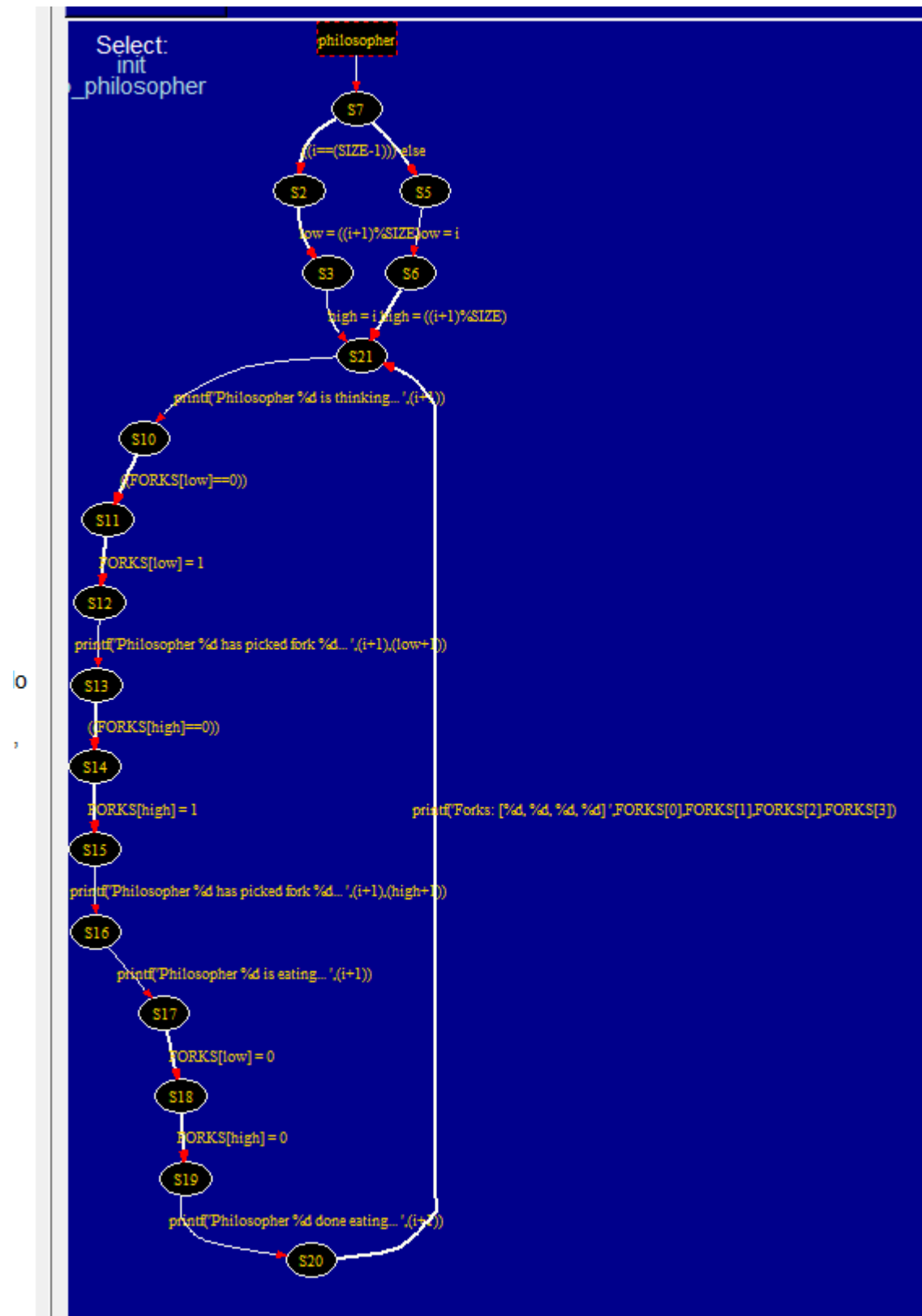
Philosopher 1 done eating...
Forks: [0, 0, 0, 0]
    Philosopher 4 has picked fork 1...
    Philosopher 4 is eating...
Philosopher 1 is thinking...
    Philosopher 3 done eating...
    Forks: [1, 0, 0, 0]
    Philosopher 2 has picked fork 2...
    Philosopher 3 is thinking...
    Philosopher 4 done eating...
Philosopher 1 has picked fork 1...
    Forks: [1, 1, 0, 0]
    Philosopher 4 is thinking...
    Philosopher 3 has picked fork 3...
    Philosopher 4 has picked fork 4...

    timeout
#processes: 5
    SIZE = 4
    FORKS[0] = 1
    FORKS[1] = 1
    FORKS[2] = 1
    FORKS[3] = 1
1883:  proc  4 (philosopher:1) .\\lab-05-q1.pml:27 (state 5)
1883:  proc  3 (philosopher:1) .\\lab-05-q1.pml:27 (state 5)
1883:  proc  2 (philosopher:1) .\\lab-05-q1.pml:27 (state 5)
1883:  proc  1 (philosopher:1) .\\lab-05-q1.pml:27 (state 5)
1883:  proc  0 (:init::1) .\\lab-05-q1.pml:13 (state 9) <valid end state>
5 processes created
PS F:\code\github.com\godcrampy\college-notes\se\pc_spin651> |

```

Deadlock occurs and the system times out. Each philosopher has one fork.

With Graph-Based Allocation:



No Deadlock! (Infinite loop next page)

```
Forks: [1, 1, 1, 0]
  Philosopher 2 is eating...
Philosopher 1 has picked fork 2...
Philosopher 1 is eating...
  Philosopher 3 is thinking...
  Philosopher 2 done eating...
  Forks: [0, 0, 1, 0]
  Philosopher 2 is thinking...
    Philosopher 3 has picked fork 3...
Philosopher 1 done eating...
Forks: [0, 1, 1, 1]
  Philosopher 2 has picked fork 2...
Philosopher 1 is thinking...
  Philosopher 4 has picked fork 1...
  Philosopher 3 has picked fork 4...
  Philosopher 3 is eating...
  Philosopher 2 has picked fork 3...
  Philosopher 3 done eating...
  Philosopher 2 is eating...
  Forks: [1, 1, 1, 1]
  Philosopher 3 is thinking...
    Philosopher 4 has picked fork 4...
    Philosopher 4 is eating...
    Philosopher 3 has picked fork 3...
  Philosopher 2 done eating...
    Philosopher 4 done eating...
    Forks: [0, 0, 1, 0]
  Forks: [0, 0, 1, 0]
    Philosopher 4 is thinking...
    Philosopher 3 has picked fork 4...
    Philosopher 3 is eating...
  Philosopher 1 has picked fork 1...
  Philosopher 2 is thinking...
    Philosopher 3 done eating...
    Forks: [1, 1, 0, 0]
    Philosopher 3 is thinking...
  Philosopher 2 has picked fork 2...
  Philosopher 2 has picked fork 3...
    Philosopher 3 has picked fork 3...
  Philosopher 1 has picked fork 2...
  Philosopher 2 is eating...
  Philosopher 1 is eating...
  Philosopher 2 done eating...
  Forks: [0, 0, 0, 1]
  Philosopher 1 done eating...
    Philosopher 3 has picked fork 4...
    Philosopher 3 is eating...
  Philosopher 2 is thinking...
  Forks: [1, 0, 0, 1]
    Philosopher 4 has picked fork 1...
```