Software Engineering (CS401)

Lab Assignment 5

U19CS012

Q1.) Classical Synchronization Problem

Dining Philosophers Problem

- ✓ 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 particularly ***slippery***, and a Philosopher needs **both forks** 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.

Code

```
/*Dining Philosophers Problem [U19CS012]*/

#define NUM_PHIL 4
int forks[NUM_PHIL];

proctype phil(int id)
{
end:
  /*The philosopher is allowed to be in any state*/
  do
    :: printf("Philosopher %d is thinking\n", id);

       /*Deduce which forks are ours*/
       int leftfork = id;
       int rightfork = id+1;
```

```
      /*Since it is Round Table*/
      if
        :: id+1 >= NUM_PHIL -> rightfork = 0;
        :: else -> rightfork = id+1;
      fi
      assert(rightfork < NUM_PHIL);

      /*Start Acquiring Forks*/
      bool leftforkacquired = false;
      bool rightforkacquired = false;
      do
         :: leftforkacquired && rightforkacquired -> break
         :: !leftforkacquired || !rightforkacquired ->
           /*Acquire Left Fork first*/
           atomic {
             do
               :: forks[leftfork] == 0 ->
                  forks[leftfork]++;
                  leftforkacquired = true;

               :: leftforkacquired -> break
             od
           }

           /*Acquire Right Fork*/
           atomic {
             if
               /*Right fork is usable*/
               :: forks[rightfork] == 0 ->
                  forks[rightfork]++;
                  rightforkacquired = true;

               /*Right fork is unusable, release left for and restart*/
               :: else ->
                  forks[leftfork]--;
                  leftforkacquired = false;
             fi
           }
      od
      assert(leftforkacquired && rightforkacquired);
      assert(forks[leftfork] == 1 && forks[rightfork] == 1);

      /*Eat*/
      printf("Philosopher %d is eating with forks %d and %d\n", id, leftfork, rightfork);
progress:
      /*Consider passing eat as progress*/
      /*Return forks*/
      forks[rightfork]--;
      forks[leftfork]--;
  od
```

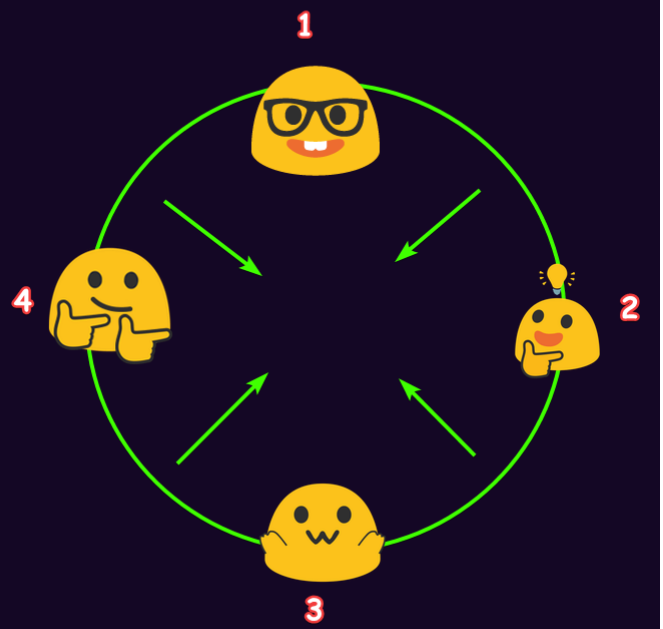
```
}

init
{
  int i = 0;
  do
    :: i >= NUM_PHIL -> break
    :: else -> run phil(i);
               i++
  od
}

ltl verify {
  [](
    forks[0] <= 1 &&
    forks[1] <= 1 &&
    forks[2] <= 1 &&
    forks[3] <= 1 &&
    forks[4] <= 1
  )
}
```

Output

```
Admin/Desktop/SEL5
 spin -n2 DiningPhilosophers.pml
ltl verify: [] ((((((forks[0]<=1)) && ((forks[1]<=1))) && ((forks[2]<=1))) && ((forks[3]<=1))) && ((forks[4]<=1)))
          Philosopher 0 is thinking
              Philosopher 1 is thinking
                  Philosopher 2 is thinking
                      Philosopher 3 is thinking
              Philosopher 1 is eating with forks 1 and 2
              Philosopher 1 is thinking
                      Philosopher 3 is eating with forks 3 and 0
                      Philosopher 3 is thinking
                  Philosopher 2 is eating with forks 2 and 3
          Philosopher 0 is eating with forks 0 and 1
                  Philosopher 2 is thinking
          Philosopher 0 is thinking
              Philosopher 1 is eating with forks 1 and 2
                      Philosopher 3 is eating with forks 3 and 0
              Philosopher 1 is thinking
                      Philosopher 3 is thinking
                  Philosopher 2 is eating with forks 2 and 3
          Philosopher 0 is eating with forks 0 and 1
                  Philosopher 2 is thinking
          Philosopher 0 is thinking
              Philosopher 1 is eating with forks 1 and 2
              Philosopher 1 is thinking
          Philosopher 0 is eating with forks 0 and 1
          Philosopher 0 is thinking
                  Philosopher 2 is eating with forks 2 and 3
                  Philosopher 2 is thinking
                      Philosopher 3 is eating with forks 3 and 0
              Philosopher 1 is eating with forks 1 and 2
                      Philosopher 3 is thinking
              Philosopher 1 is thinking
          Philosopher 0 is eating with forks 0 and 1
          Philosopher 0 is thinking
                  Philosopher 2 is eating with forks 2 and 3
                  Philosopher 2 is thinking
```



SUBMITTED BY: U19CS012 BHAGYA VINOD RANA