Some Classic Concurrency Problems

The Dining Philosophers

Joseph Kehoe[1]

[1] Department of Computing and Networking
Institute of Technology Carlow

CDD101, 2017

Dining Philosophers

- 5 philosophers sit at a table
- Each has a plate of spaghetti on a plate in front of him/her
- There are five forks
  - one between each plate
- Philosophers think and eat
- In order to eat they need two forks
  - one from each side

The Problem

- Only one philosopher can hold a fork at a time
- It must be impossible for deadlock to occur
- It must be impossible for a philosopher to starve waiting for a fork
- It must be possible for more than one philosopher to eat at the same time

Problem (continued)

```
while true:
think()
get_forks()
eat()
put_forks()
```

- Assume philosophers are numbered 0 to 5
- forks are numbered 0 to 5
- philosopher j has fork j on left
- and fork j+1 on right (modulo 5)

Bad Solution

```
def left(j): return j
def right(j): return (j+1)%5
forks=[semaphore(1) for j in range(5)]

def get_forks(j):
    fork[right(j)].wait()
    fork[left(j)].wait()

def put_forks(j):
    fork[right(j)].signal()
    fork[left(j)].signal()
```

Hint

- if only four philosophers are allowed to sit at the table at a time then deadlock is impossible
- isn't it?

```
footman=semaphore(4)
```

Solution One

```
def get_forks(j):
    footman.wait()
    fork[right(j)].wait()
    fork[left(j)].wait()
def put_forks(j):
    fork[right(j)].signal()
    fork[left(j)].signal()
    footman.signal()
```

Solution Two

- Problem is that everyone picks up their left fork first
- If one picks up their right fork first then we cannot deadlock
  - is this true?
  - Prove it! (proof by contradiction)