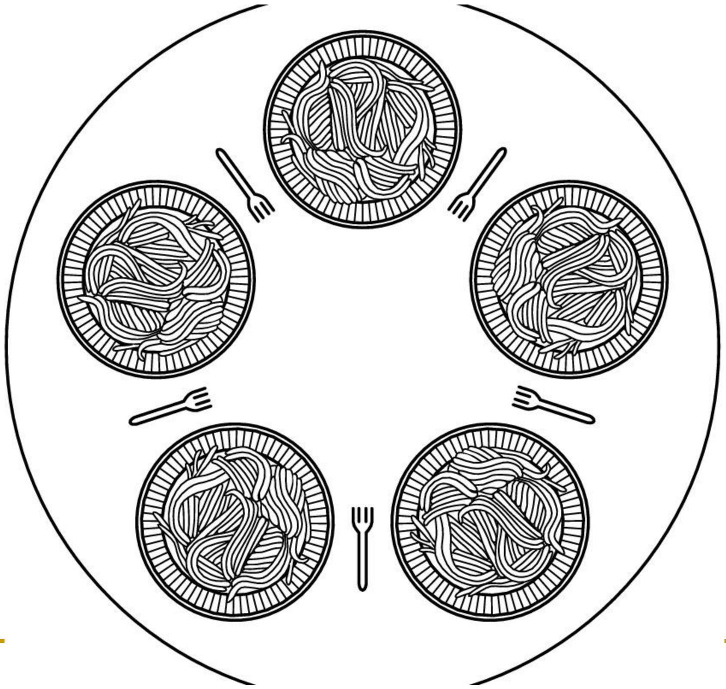
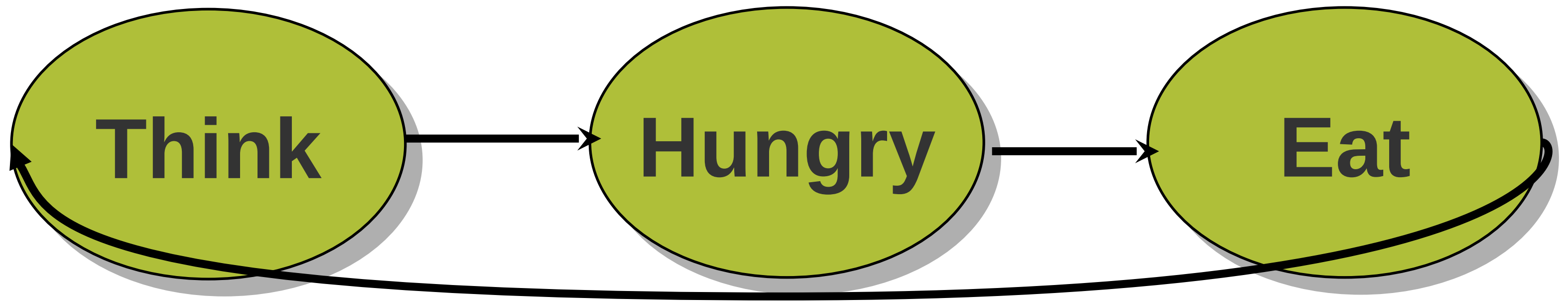
# Diners/Philosophers Problem

# Dining Philosophers Problem

- Five philosophers sit around a table
- Each philosopher has a plate of food
- There is one fork between any two philosophers



## Philosopher’s States



## Variables

- **`N = # of Philosophers`**
- **`Bool ForkAvailable[N] = {FALSE};`**
- Initially none of the forks is available
- **`Lock lock`**
- Lets define two utility functions

```
int Left(int i)
{       return (i + N -1)%N;}
Int Right(int i)
{       return (i + N)%N;   }
```

```
void Philosopher(int i){
  while( true){
   wait while Fork[Left(i)] is Unavailable;
   Fork[Left(i)] = UNAVAILABLE;
   wait while Fork[Right(i)] is
  Unavailable;
   Fork[Right(i)] = UNAVAILABLE;
   eat();
   Fork[Left(i)] = AVAILABLE;
   Fork[Right(i)] = AVAILABLE;
  }
}
```

**Do you see any problem here???**
**It can lead to deadlock.**

- int State[5]
  - Each phil. has his own state. thinking, hungry, eating.
- Condition self[5]
  - Each phil. has his own condition variable.
- Lock L
  - Lock to enter and leave the monitor.

## Design of each function

- void PickUp(int i)
  - {
    - Give me exclusive access
    - Set my status to Hungry
    - Test to see if I can get the forks to my right and left
    - As long as my neighbors are eating I cant eat
    - Now I’m eating
    - I’ll leave the forks when i’ll complete
  - }

```
void Philosopher(int i){
   lock->Acquire
   Status[i] = HUNGRY;
   while (Status[Left(i)] = EATING ||
         Status[Right(i)] = EATING)
                              CV[i]->wait(lock);
   Status[i] = EATING;
   lock->Release();
   EAT();
   lock->Acquire
   Status[i] = THINKING;
   CV[Left(i)]->Signal();
   CV[Right(i)]->Signal();
   lock->Release();
}
```