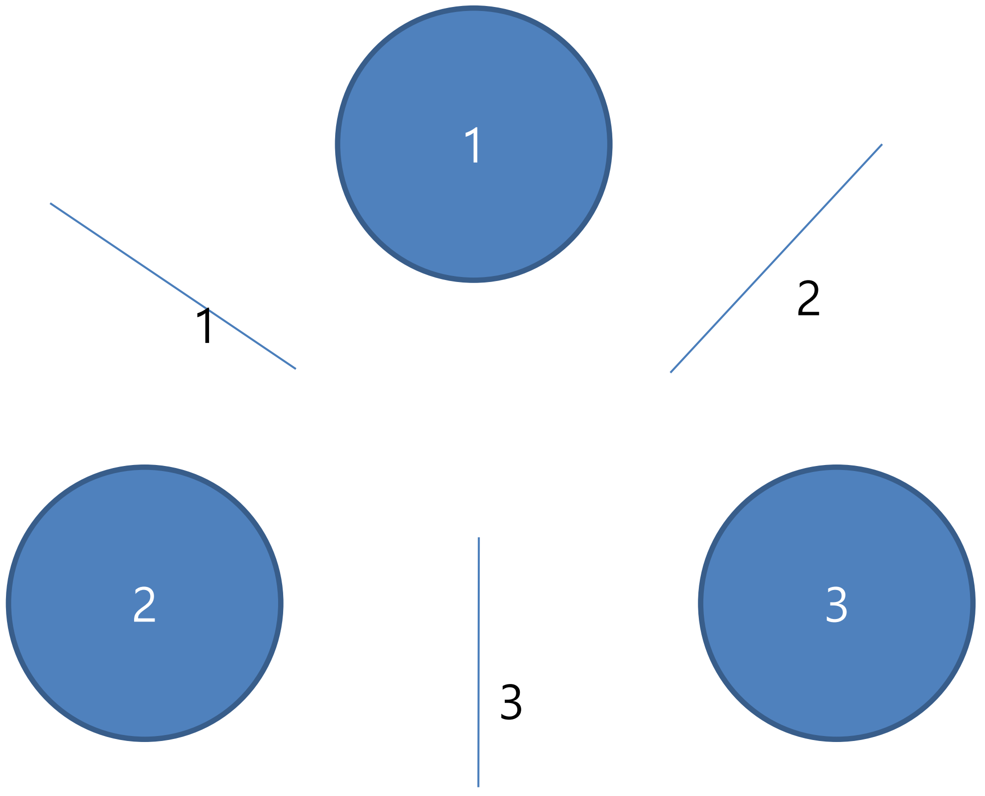
Dining Philosophers

정세진
김재엽

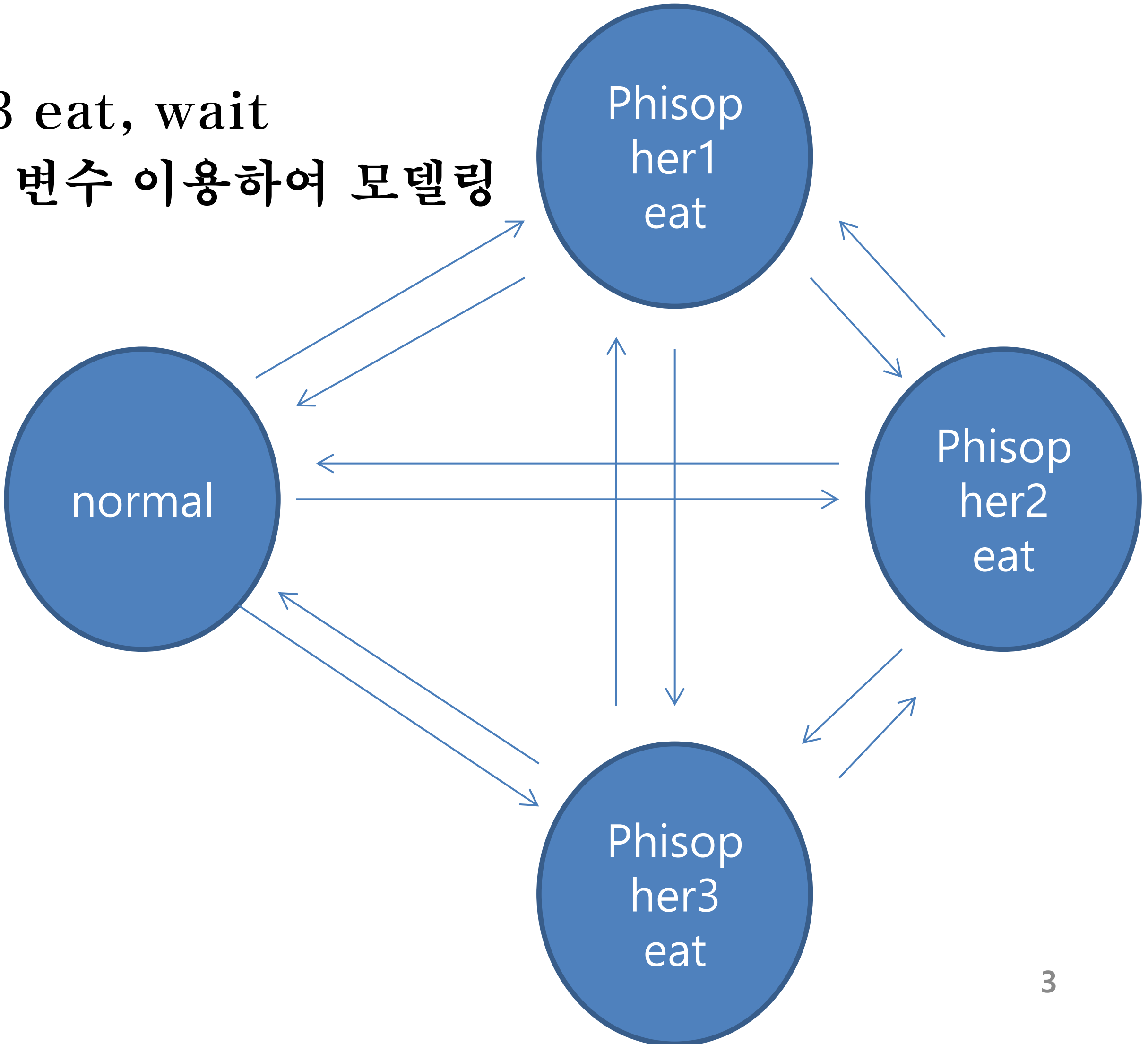
Dining Philosopher Problem

- 3 man



Dining Philosopher Problem

- 3man with state
- Philosopher 1,2,3 eat, wait
- Fork 1,2,3 사용 중 변수 이용하여 모델링



Dining Philosopher Problem

- 3man 1

```
module main(){
        onePut, twoPut, threePut, out1, out2, out3 : boolean;
        fork1, fork2, fork3 : 0..3;
        wait1, wait2, wait3 : boolean;
        eat1, eat2, eat3 : boolean;
        state :{normal, pheat1, pheat2, pheat3};
        default {
                init(fork1) := 0;
                init(fork2) := 0;
                init(fork3) := 0;
                init(wait1) := 0;
                init(wait2) := 0;
                init(wait3) := 0;
                init(eat1) := 0;
                init(eat2) := 0;
                init(eat3) := 0;
                init(state) := normal;
        }
```

```
in switch(state) {
        normal: {
                if(onePut) {
                        next(state) := pheat1;
                        next(eat1) := 1;
                        next(fork1) := 1;
                        next(fork2) := 1;
                } else if(twoPut){
                        next(state) := pheat2;
                        next(eat2) := 1;
                        next(fork1) := 2;
                        next(fork3) := 2;
                } else if(threePut) {
                        next(state) := pheat3;
                        next(eat3) := 1;
                        next(fork2) := 3;
                        next(fork3) := 3;
                } else {
                        next(state) := normal;
                }
        }
        ......중략
```

Dining Philosopher Problem

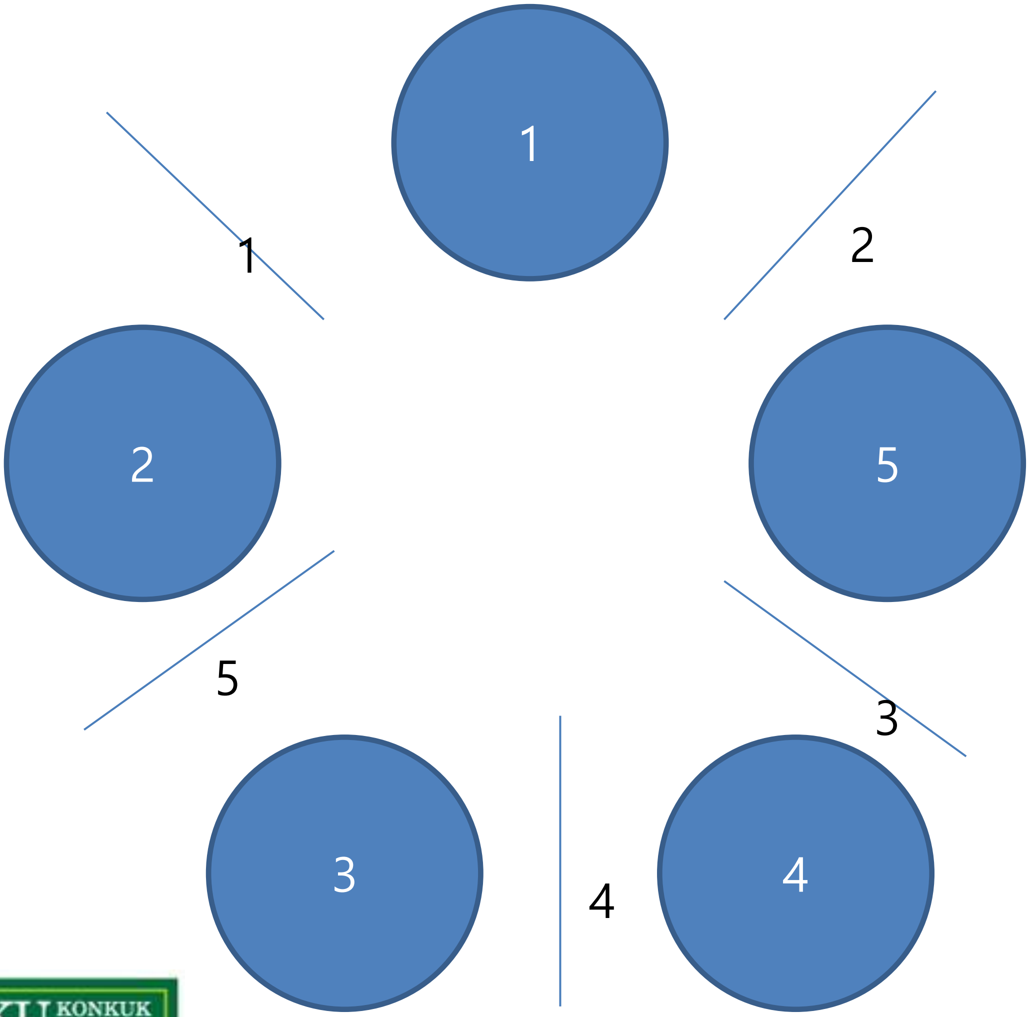
- 3man 2

```
.... 중략

in {
if(onePut) {
        if(eat1 =~1) {
                if(fork1 = 0 & fork2 = 0) {
                        next(fork1) := 1;
                        next(fork2) := 1;
                        next(eat1) := 1;
                } else{
                        next(wait1) := 1;
                        if(fork1 = 0){
                                next(fork1) := 1;
                        }
                        if(fork2 = 0){
                                next(fork2) := 1;
                        }
                }
        }
}
```

```
else if(twoPut){
        if(eat2 = ~1){
                if(fork1 = 0 & fork3 = 0) {
                        next(fork1) := 2;
                        next(fork3) := 2;
                        next(eat2) := 1;
                } else{
                        next(wait2) := 1;
                        if(fork1 = 0){
                                next(fork1) := 2;
                        }
                        if(fork3 = 0){
                                next(fork3) := 2;
                        }
                }
        }
}

.....중략
```

Dining Philosopher Problem

- 5 man



Dining Philosopher Problem

- 5 man

```
module main(){
        onePut, twoPut, threePut,fourPut, fivePut, out1, out2, out3, out4, out5 : boolean;
        fork1, fork2, fork3, fork4, fork5 : 0..5;
        wait1, wait2, wait3, wait4, wait5 : boolean;
        noone,eat1, eat2, eat3,eat4,eat5 : boolean;

……

in {
                if(onePut) {
                        if(eat1 =~1) {
                                if(fork1 = 0 & fork2 = 0) {
                                        next(fork1) := 1;
                                        next(fork2) := 1;
                                        next(eat1) := 1;
                                } else{
                                        next(wait1) := 1;
                                        if(fork1 = 0){
```

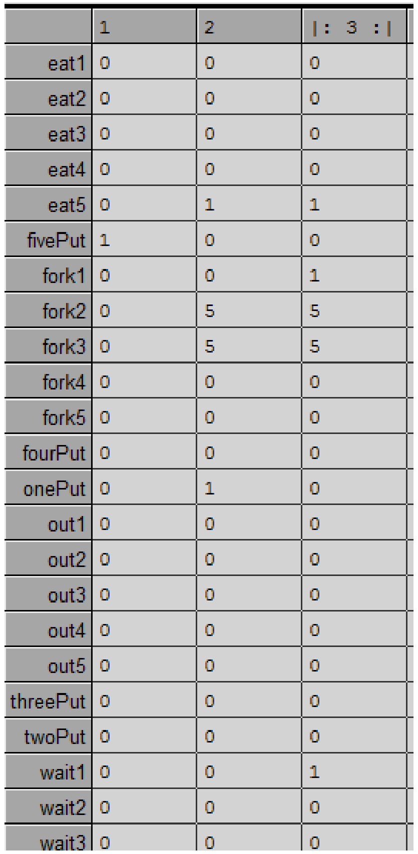
Dining Philosopher Problem

- 공통 property
- test1 : SPEC AF(EF(eat1 = 1) & EF(eat2 = 1) & EF(eat3 = 1) & EF(eat4 = 1) & EF(eat5 = 1));
  - 모두가 한번씩은 먹는 경우가 존재하는지 여부에 대한 property
- Test_deadlock : SPEC EF(fork1=1 & fork2=5 & fork3=4 & fork4=3 & fork5=2);
  - Deadlock이 생기는 경우가 존재하는지 여부에 대한 property

DEPENDABLE SOFTWARE LABORATORY
KU KONKUK UNIVERSITY

Dining Philosopher Problem

- test2 : SPEC AG(onePut ->AF(eat1 = 1));
  - 1번이 집으려 하면 먹는 상태로 갈 수 있는지에 대한 property
  - False -> 5번이 집어서 lock이 걸린 상태에서 1번이 집으려 하는 경우 문제



	1	2	|: 3 :|
eat1	0	0	0
eat2	0	0	0
eat3	0	0	0
eat4	0	0	0
eat5	0	1	1
fivePut	1	0	0
fork1	0	0	1
fork2	0	5	5
fork3	0	5	5
fork4	0	0	0
fork5	0	0	0
fourPut	0	0	0
onePut	0	1	0
out1	0	0	0
out2	0	0	0
out3	0	0	0
out4	0	0	0
out5	0	0	0
threePut	0	0	0
twoPut	0	0	0
wait1	0	0	1
wait2	0	0	0
wait3	0	0	0

Dining Philosopher Problem

- 10man, 15man…

Q & A

E N D