CS766
Analysis of Concurrent Programs

09 Feb 2021
Mutual exclusion protocols

Szymanski Algorithm

A (class) presentation by
Karnika Shivhare
IITB 204050010

SZYMANSKI ALGORITHM

Algorithm for mutual exclusion

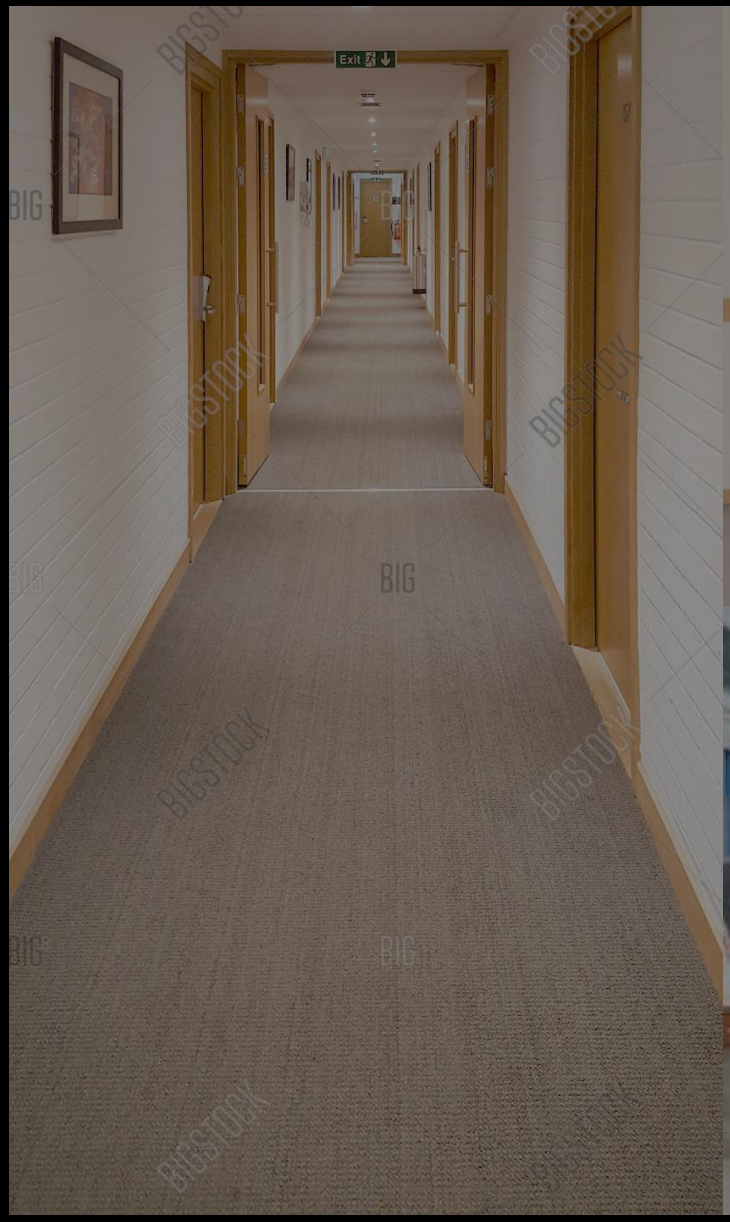
Karnika Shivhare 204050010 IITB CS 766 9th Feb 2021

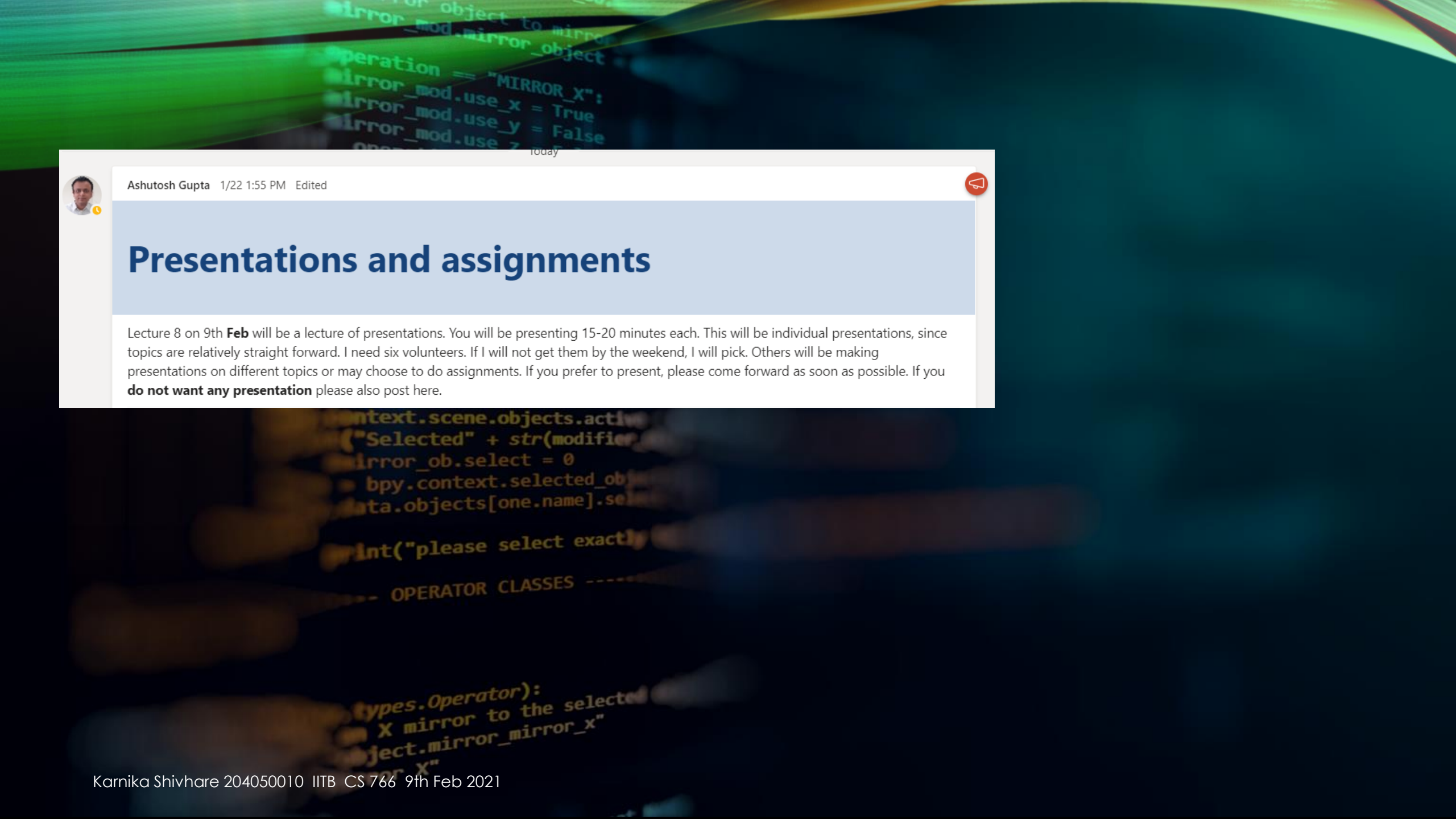
SZYMANSKI ALGORITHM

Drawbacks of previous discussed algorithms:

PETERSON	O(n2) pre-protocol
BAKERY	Use unbounded ticket numbers
DEKKER	Don't scale well beyond 2 processes

Out of scope of the 15 minutes.


Exit


Today
Ashutosh Gupta 1/22 1:55 PM Edited
Presentations and assignments
Lecture 8 on 9th **Feb** will be a lecture of presentations. You will be presenting 15-20 minutes each. This will be individual presentations, since topics are relatively straight forward. I need six volunteers. If I will not get them by the weekend, I will pick. Others will be making presentations on different topics or may choose to do assignments. If you prefer to present, please come forward as soon as possible. If you **do not want any presentation** please also post here.

Presentations and assignments

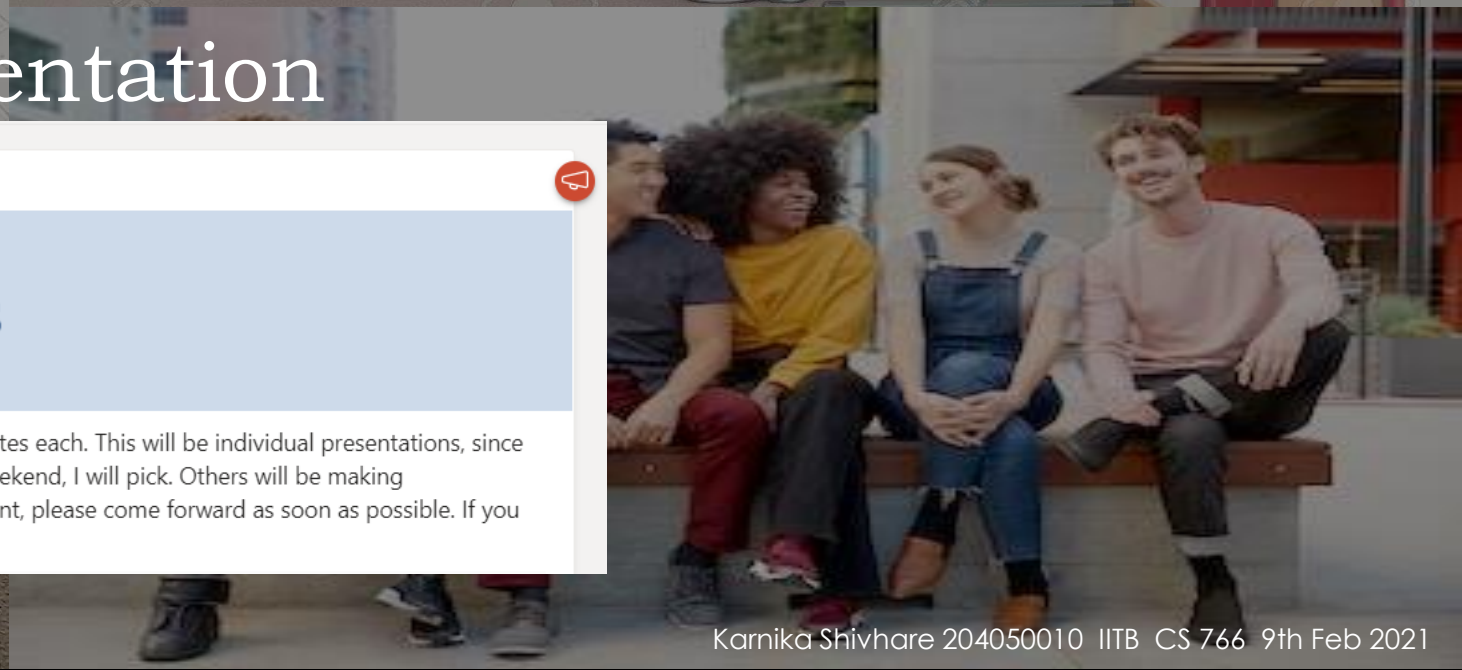
Critical Section = Presentation

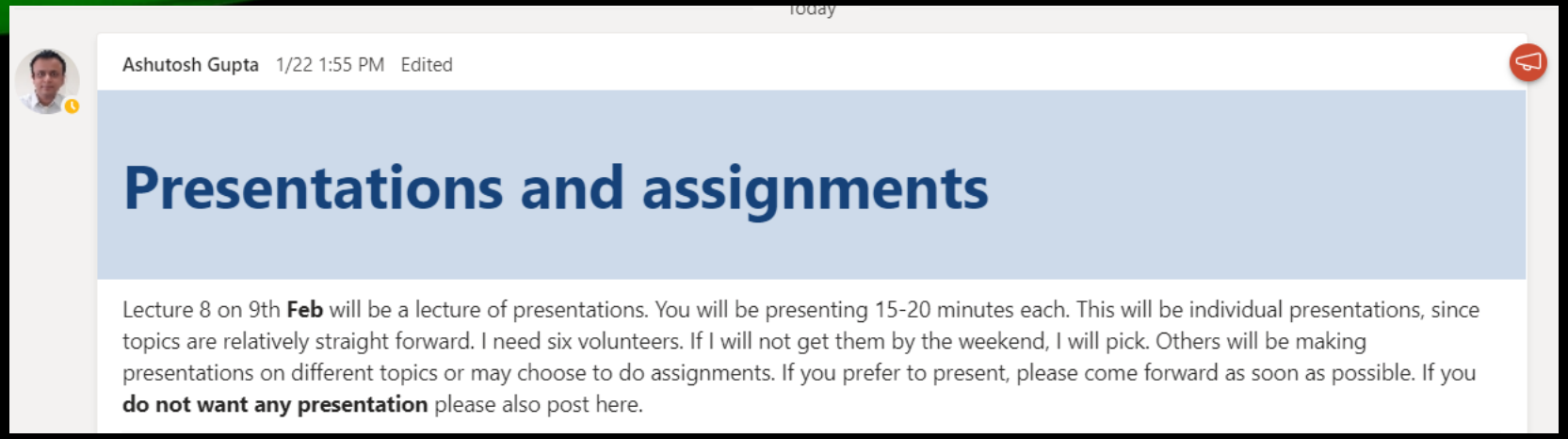
Today

Ashutosh Gupta 1/22 1:55 PM Edited

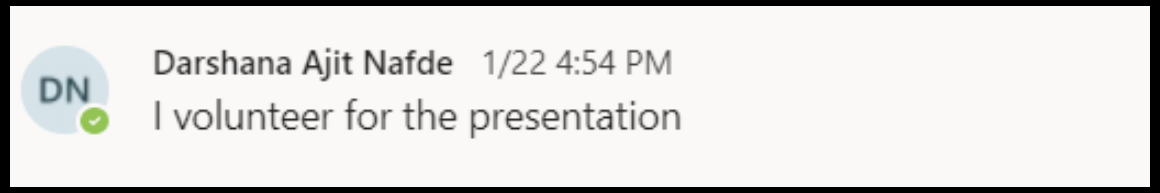
Presentations and assignments

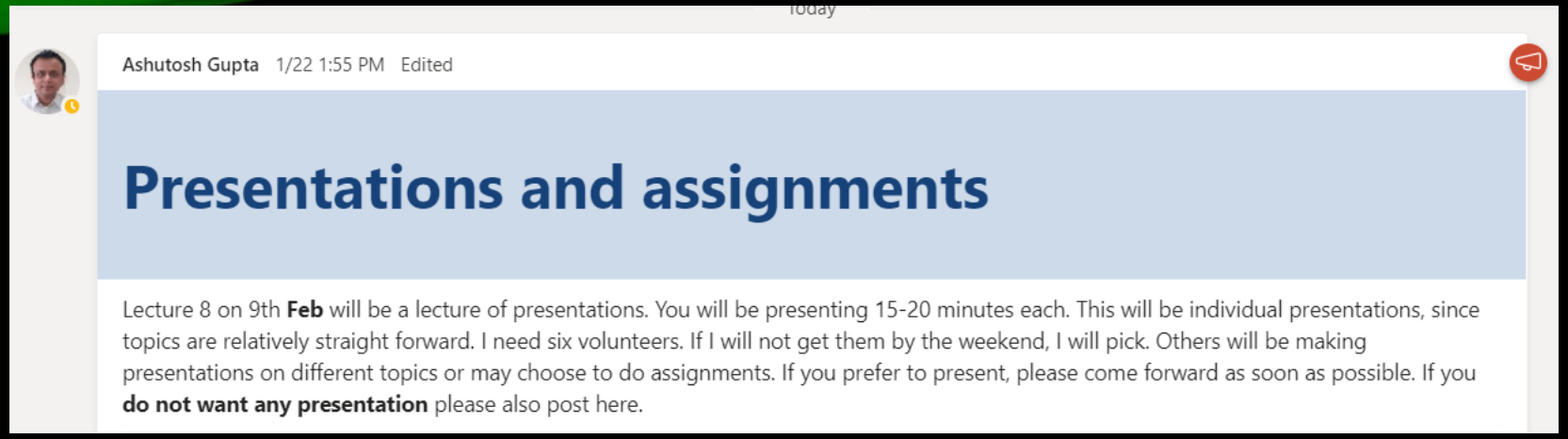
Lecture 8 on 9th **Feb** will be a lecture of presentations. You will be presenting 15-20 minutes each. This will be individual presentations, since topics are relatively straight forward. I need six volunteers. If I will not get them by the weekend, I will pick. Others will be making presentations on different topics or may choose to do assignments. If you prefer to present, please come forward as soon as possible. If you **do not want any presentation** please also post here.



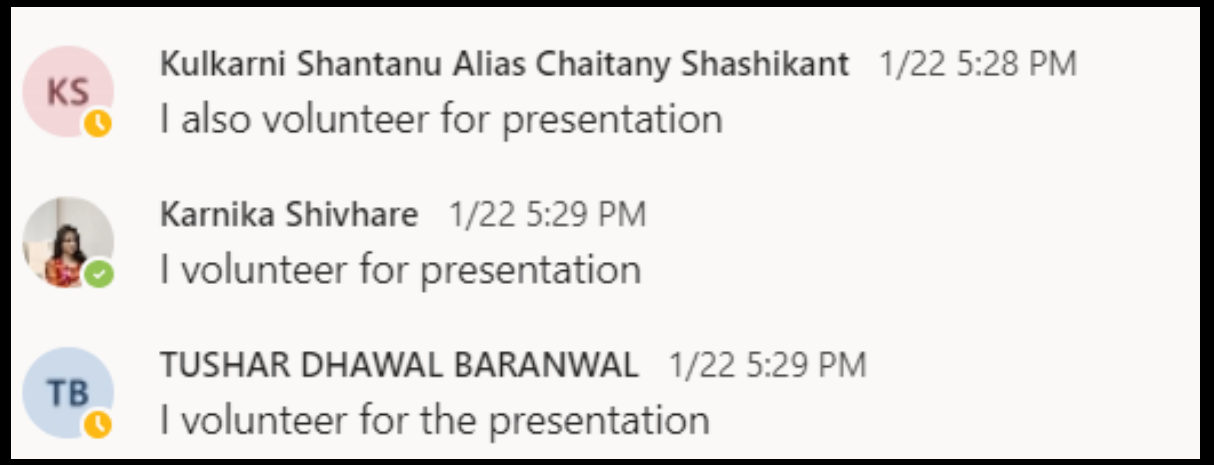

Today
Ashutosh Gupta 1/22 1:55 PM Edited
Presentations and assignments
Lecture 8 on 9th Feb will be a lecture of presentations. You will be presenting 15-20 minutes each. This will be individual presentations, since topics are relatively straight forward. I need six volunteers. If I will not get them by the weekend, I will pick. Others will be making presentations on different topics or may choose to do assignments. If you prefer to present, please come forward as soon as possible. If you do not want any presentation please also post here.

Flag[self] = 1


DN
Darshana Ajit Nafde 1/22 4:54 PM
I volunteer for the presentation



Everyone raised their flags = 1 who wished to enter critical section at around same time.



Today

Ashutosh Gupta 1/22 1:55 PM Edited

Presentations and assignments

Lecture 8 on 9th **Feb** will be a lecture of presentations. You will be presenting 15-20 minutes each. This will be individual presentations, since topics are relatively straight forward. I need six volunteers. If I will not get them by the weekend, I will pick. Others will be making presentations on different topics or may choose to do assignments. If you prefer to present, please come forward as soon as possible. If you **do not want any presentation** please also post here.

DN **Darshana Ajit Nafde** 1/22 4:54 PM
I volunteer for the presentation

KS **Kulkarni Shantanu Alias Chaitany Shashikant** 1/22 5:28 PM
I also volunteer for presentation

Karnika Shivhare 1/22 5:29 PM
I volunteer for presentation

TB **TUSHAR DHAWAL BARANWAL** 1/22 5:29 PM
I volunteer for the presentation

Today

Ashutosh Gupta 1/22 1:55 PM Edited

Presentations and assignments

Lecture 8 on 9th **Feb** will be a lecture of presentations. You will be presenting 15-20 minutes each. This will be individual presentations, since topics are relatively straight forward. I need six volunteers. If I will not get them by the weekend, I will pick. Others will be making presentations on different topics or may choose to do assignments. If you prefer to present, please come forward as soon as possible. If you **do not want any presentation** please also post here.

Ashutosh Gupta 1/22 7:20 PM
I have six volunteers now.

Pooja Verma 1/22 8:02 PM
I m interested too for presentation

Ashutosh Gupta 1/22 8:03 PM
The first six presentations are already scheduled. Please look at the course page.

Darshana Ajit Nafde 1/22 4:54 PM
I volunteer for the presentation

Kulkarni Shantanu Alias Chaitany Shashikant 1/22 5:28 PM
I also volunteer for presentation

Karnika Shivhare 1/22 5:29 PM
I volunteer for presentation

TUSHAR DHAWAL BARANWAL 1/22 5:29 PM
I volunteer for the presentation

Today

Ashutosh Gupta 1/22 1:55 PM Edited

Presentations and assignments

Lecture 8 on 9th **Feb** will be a lecture of presentations. You will be presenting 15-20 minutes each. This will be individual presentations, since topics are relatively straight forward. I need six volunteers. If I will not get them by the weekend, I will pick. Others will be making presentations on different topics or may choose to do assignments.
do not want any presentation please also post here.

it Nafde 1/22 4:54 PM
for the presentation

ntanu Alias Chaitany Shashikant 1/22 5:28 PM
nteer for presentation

hare 1/22 5:29 PM
for presentation

AWAL BARANWAL 1/22 5:29 PM
for the presentation

Ashutosh Gupta 1/22 7:20 PM
I have six volunteers now.

Pooja Verma 1/22 8:02 PM
I m interested too for presentation

Ashutosh Gupta 1/22 8:03 PM
The first six presentations are already scheduled. Please look at the course page.

In Crictical Section

SZYMANSKI ALGORITHM

Idea

The prologue is modeled after a waiting room with two doors.

SZYMANSKI ALGORITHM

Idea

The prologue is modeled after a waiting room with two doors.

All processes requesting entry to the CS at roughly the same time gather first in the waiting room.

SZYMANSKI ALGORITHM

Idea

The prologue is modeled after a waiting room with two doors – entry and exit.
All processes requesting entry to the CS at roughly the same time gather first in the waiting room.

Last of them closes the entry door and opens the exit door.

SZYMANSKI ALGORITHM

Idea

The prologue is modeled after a waiting room with two doors.
All processes requesting entry to the CS at roughly the same time gather first in the waiting room.
Last of them closes the entry door and opens the exit door.

From there, one by one, they enter their CS.

SZYMANSKI ALGORITHM

Idea

The prologue is modeled after a waiting room with two doors.
All processes requesting entry to the CS at roughly the same time gather first in the waiting room.
Last of them closes the entry door and opens the exit door.
From there, one by one, they enter their CS.

The last process to leave the CS, closes the exit door and reopens the entry door, so that the next batch of processes may enter.

SZYMANSKI ALGORITHM

Idea

The prologue is modeled after a waiting room with two doors.
All processes requesting entry to the CS at roughly the same time gather first in the waiting room.
Then, when there are no more processes requesting entry, waiting processes move to the end of the prologue.
From there, one by one, they enter their CS.

Any other process requesting entry to its CS at that time has to wait in the initial part of the prologue (before the waiting room).

SHARED VARIABLES

Each process exclusively writes a variable flag, which is read by all the other processes.

It assumes one of five values:

0 Executing non-CS
1 i wants to enter the CS
2 i waits for other processes to enter waiting room
3 i just entered the waiting room
4 i is leaving the waiting room. entry door is closed and exit door is open.

SZYMANSKI ALGORITHM

ENTRY PROTOCOL

flag[self] ← 1
await(all flag[1..N] ∈ {0, 1, 2})
flag [self] ← 3
if any flag[1..N] = 1:
    flag[self] ← 2
    await(any flag[1..N] = 4)
flag[self] ← 4
await(flag[1..self-1] ∈ {0, 1})

Critical Section

EXIT PROTOCOL

await(all flag [self+1..N] ∈ {0, 1, 4})
flag[self] ← 0

ENTRY PROTOCOL process 1

flag[self =1] ← 1

await(flag[2] ∈ {0, 1, 2})

flag [self = 1] ← 3

if flag[2] = 1:

    flag[self =1] ← 2

    await(if flag[2] = 4)

flag[self = 1] ← 4

//await(flag[1..self-1] ∈ {0, 1})

process 1 in Critical Section

EXIT PROTOCOL

await(flag[2] ∈ {0, 1, 4})

flag[self = 1] ← 0

ENTRY PROTOCOL process 2

flag[self =2] ← 1

await(flag[1] ∈ {0, 1, 2})

flag [self = 2] ← 3

if flag[1] = 1:

    flag[self =2] ← 2

    await(flag[1] = 4)

flag[self = 2] ← 4

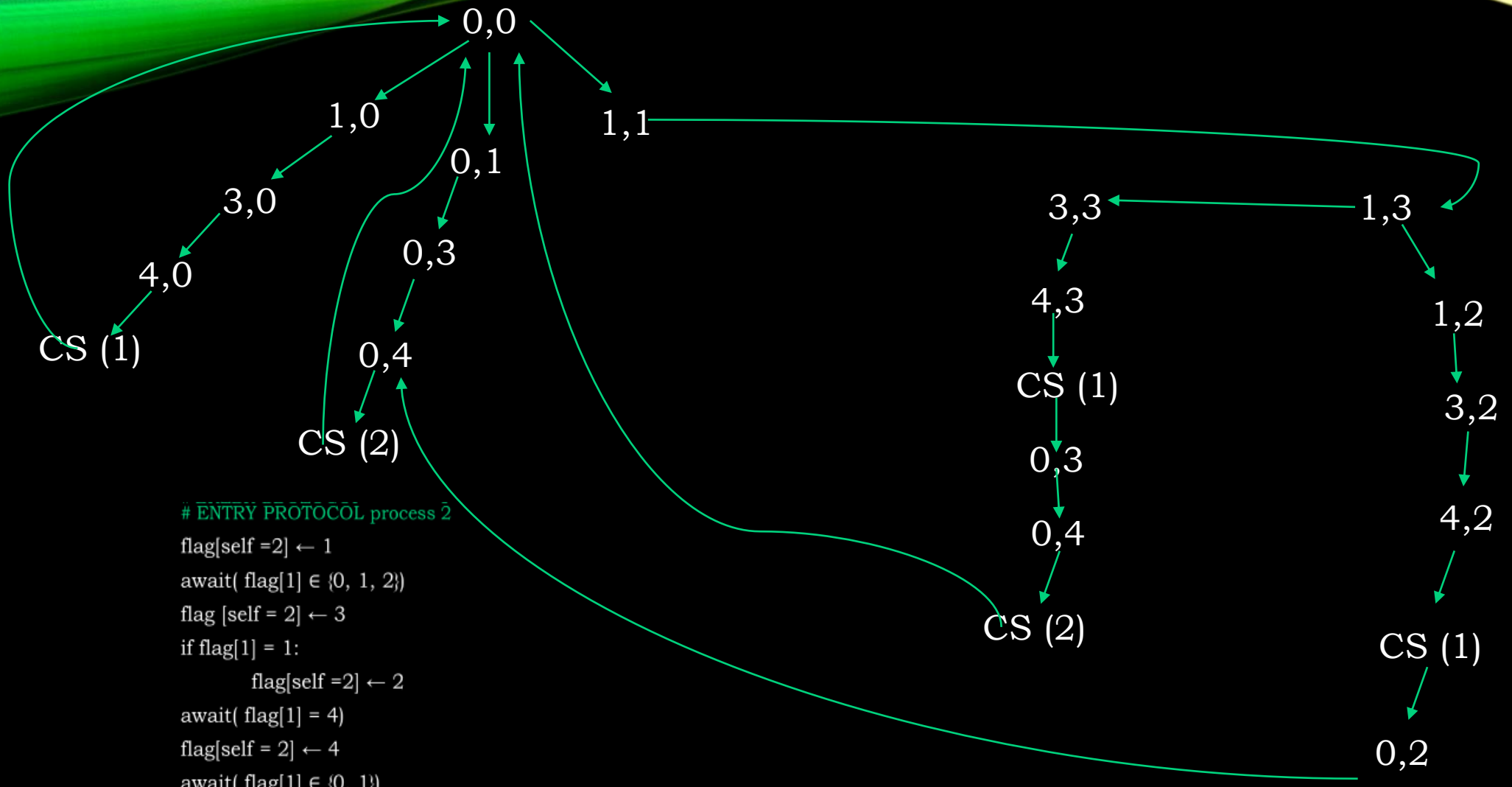
await(flag[1] ∈ {0, 1})

Process 2 in Critical Section

EXIT PROTOCOL

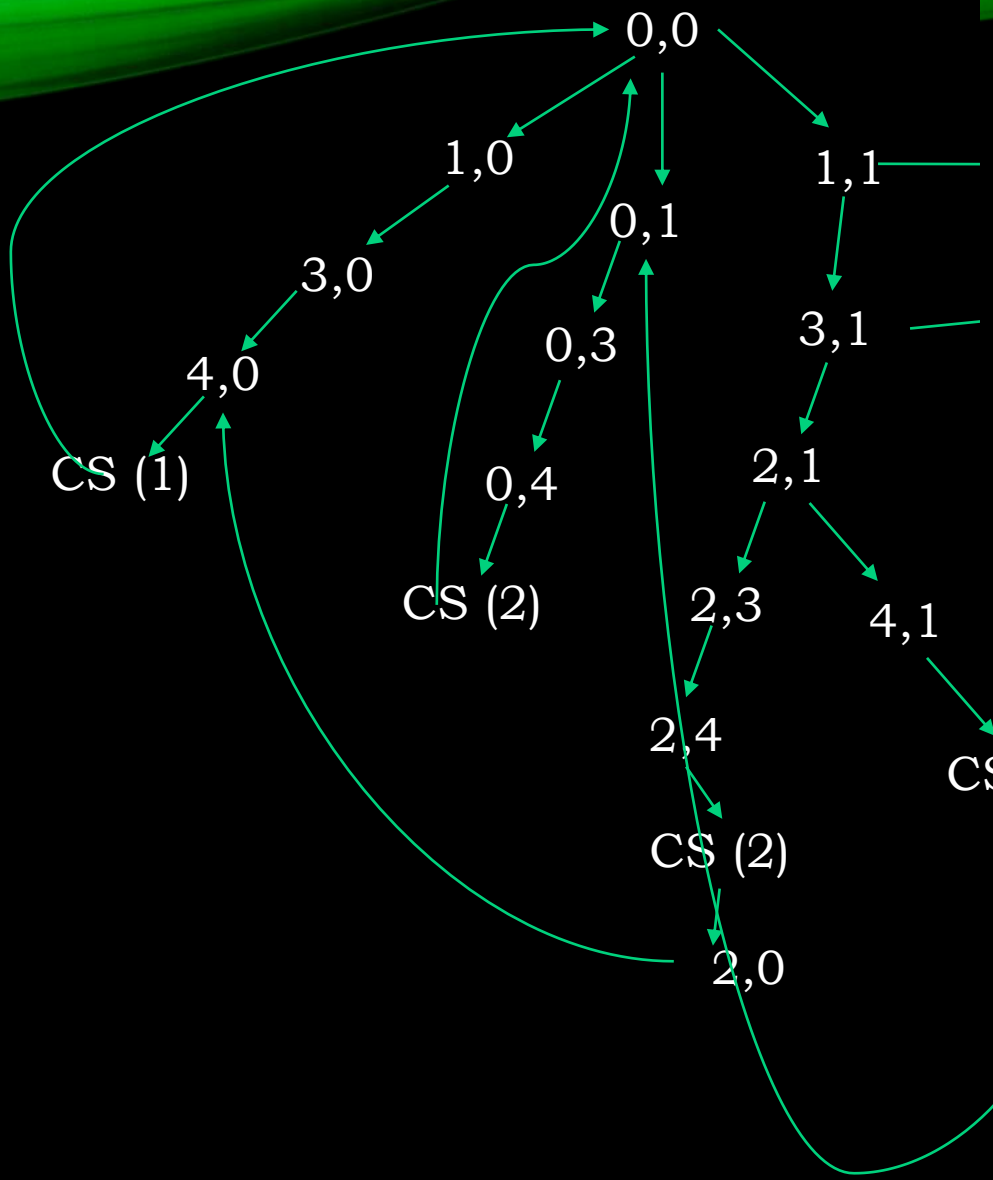
await(flag[1] ∈ {0, 1, 4})

flag[self = 2] ← 0



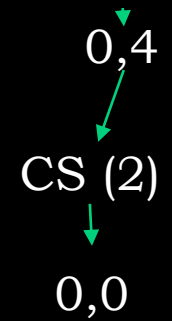

```
# ENTRY PROTOCOL process 1
flag[self =1] ← 1
await( flag[2] ∈ {0, 1, 2})
flag [self = 1] ← 3
if flag[2] = 1:
        flag[self =1] ← 2
await(if flag[2] = 4)
flag[self = 1] ← 4
//await( flag[1..self-1] ∈ {0, 1})

process 1 in Critical Section

# EXIT PROTOCOL
await( flag[2] ∈ {0, 1, 4})
flag[self = 1] ← 0
```

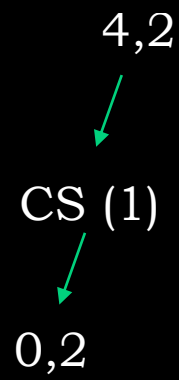
```
# ENTRY PROTOCOL process 2
flag[self =2] ← 1
await( flag[1] ∈ {0, 1, 2})
flag [self = 2] ← 3
if flag[1] = 1:
        flag[self =2] ← 2
await( flag[1] = 4)
flag[self = 2] ← 4
await( flag[1] ∈ {0, 1})

Process 2 in Critical Section

# EXIT PROTOCOL
await( flag[1] ∈ {0, 1, 4})
flag[self = 2] ← 0
```

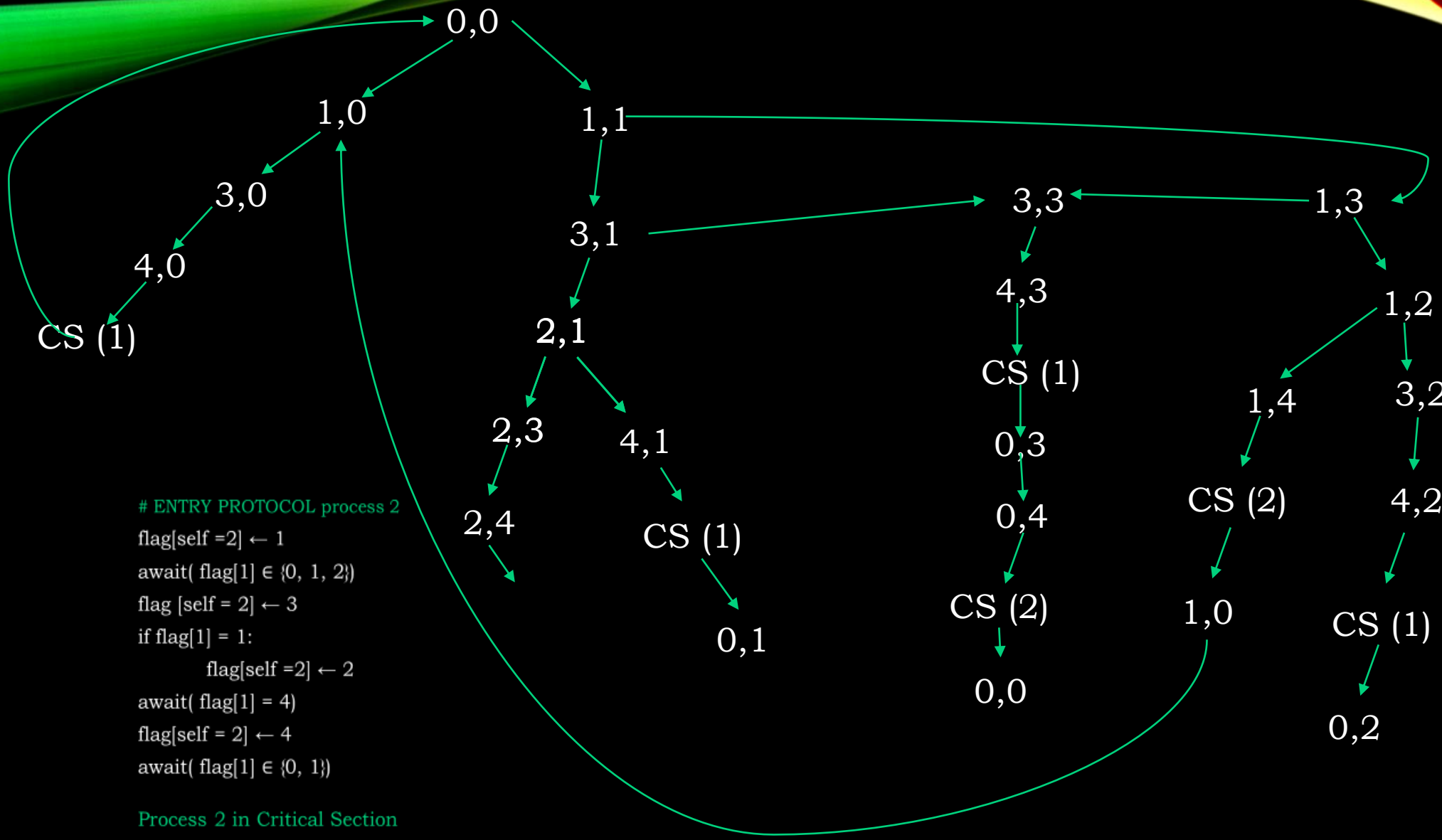



```
# ENTRY PROTOCOL process 1
flag[self =1] ← 1
await( flag[2] ∈ {0, 1, 2})
flag [self = 1] ← 3
if flag[2] = 1:
        flag[self =1] ← 2
await(if flag[2] = 4)
flag[self = 1] ← 4
//await( flag[1..self-1] ∈ {0, 1})

process 1 in Critical Section

# EXIT PROTOCOL
await( flag[2] ∈ {0, 1, 4})
flag[self = 1] ← 0
```

```
# ENTRY PROTOCOL process 2
flag[self =2] ← 1
await( flag[1] ∈ {0, 1, 2})
flag [self = 2] ← 3
if flag[1] = 1:
        flag[self =2] ← 2
await( flag[1] = 4)
flag[self = 2] ← 4
await( flag[1] ∈ {0, 1})

Process 2 in Critical Section

# EXIT PROTOCOL
await( flag[1] ∈ {0, 1, 4})
flag[self = 2] ← 0
```

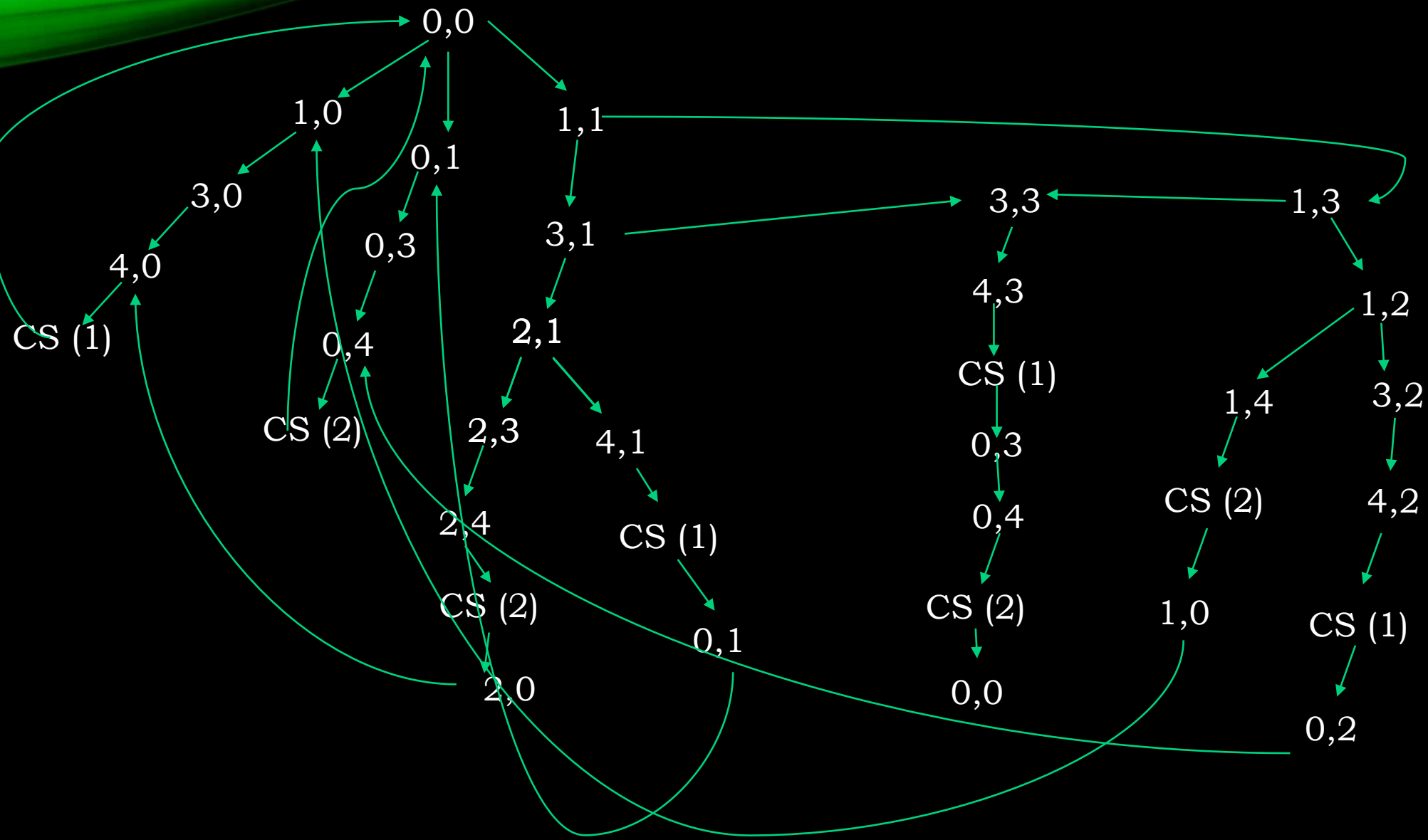






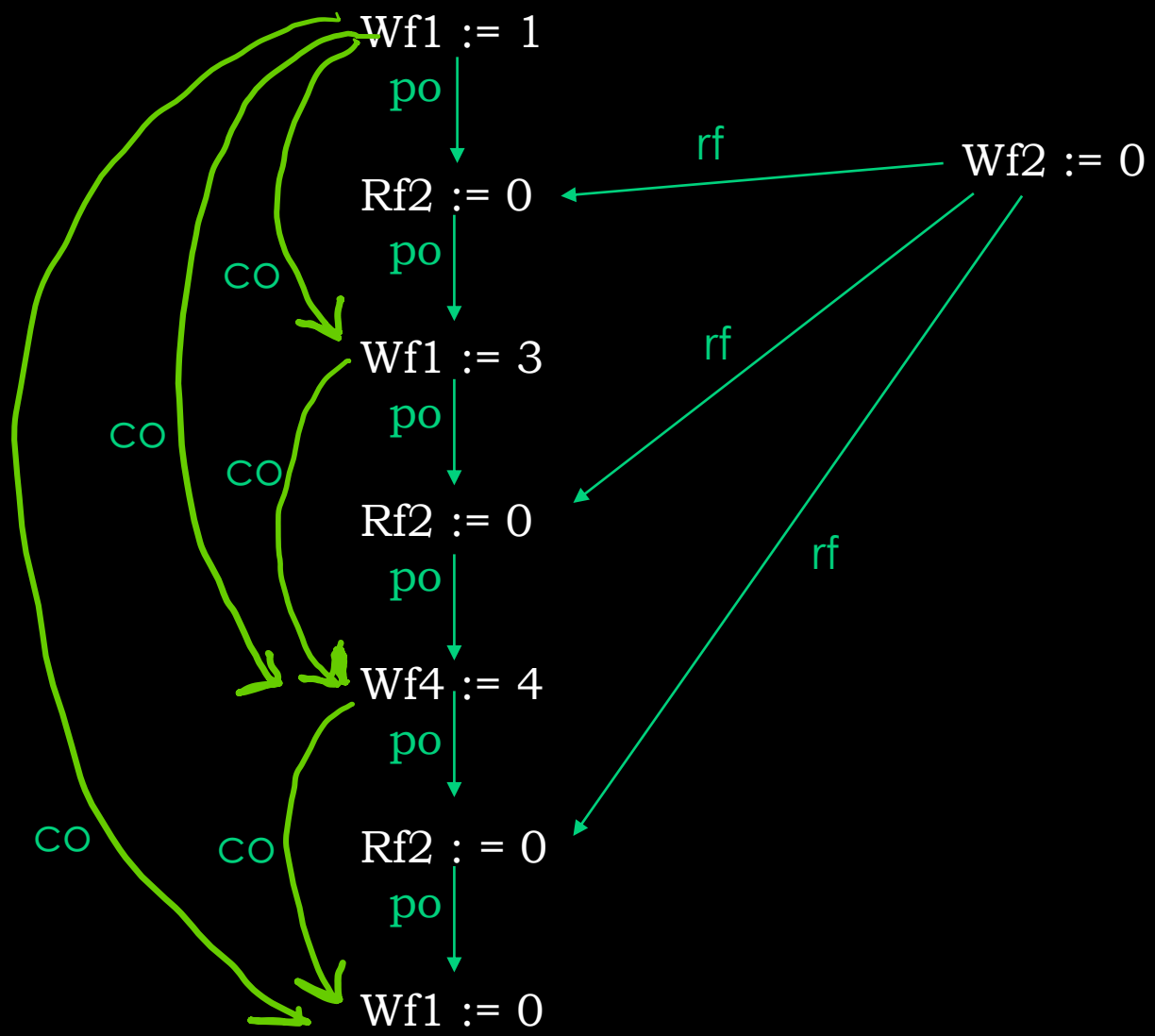

```
# ENTRY PROTOCOL process 1
flag[self =1] ← 1
await( flag[2] ∈ {0, 1, 2})
flag [self = 1] ← 3
if flag[2] = 1:
        flag[self =1] ← 2
await(if flag[2] = 4)
flag[self = 1] ← 4
//await( flag[1..self-1] ∈ {0, 1})

process 1 in Critical Section

# EXIT PROTOCOL
await( flag[2] ∈ {0, 1, 4})
flag[self = 1] ← 0
```

```
# ENTRY PROTOCOL process 2
flag[self =2] ← 1
await( flag[1] ∈ {0, 1, 2})
flag [self = 2] ← 3
if flag[1] = 1:
        flag[self =2] ← 2
await( flag[1] = 4)
flag[self = 2] ← 4
await( flag[1] ∈ {0, 1})

Process 2 in Critical Section

# EXIT PROTOCOL
await( flag[1] ∈ {0, 1, 4})
flag[self = 2] ← 0
```


0,0
1,0
1,1
0,1
3,0
3,3
1,3
0,3
3,1
4,0
4,3
1,2
CS (1)
0,4
2,1
CS (1)
1,4
3,2
CS (2)
2,3
4,1
0,3
2,4
CS (1)
0,4
CS (2)
4,2
CS (2)
CS (2)
1,0
CS (1)
0,1
0,0
2,0
0,2

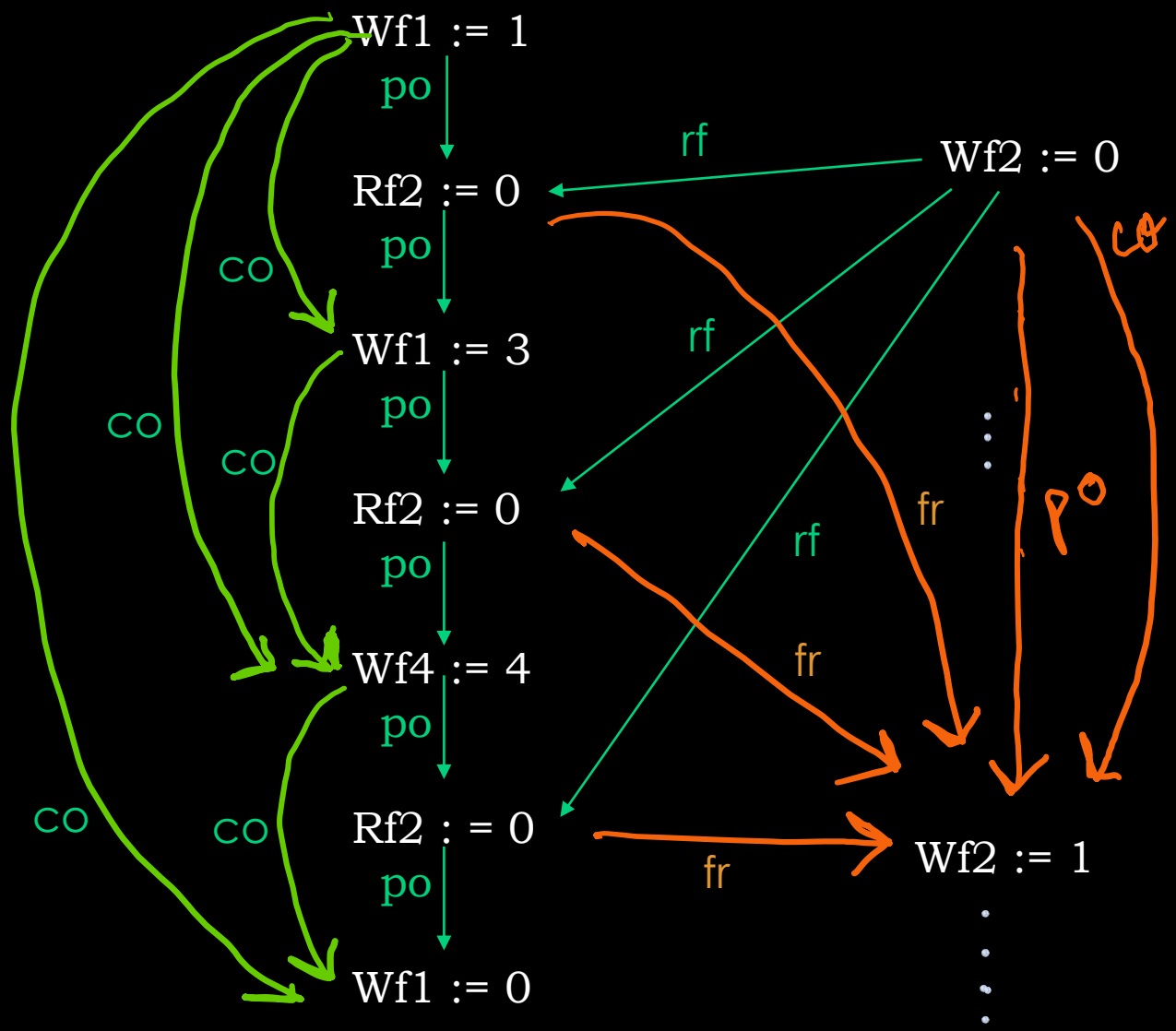
```
# ENTRY PROTOCOL process 1
flag[self =1] ← 1
await( flag[2] ∈ {0, 1, 2})
flag [self = 1] ← 3
if flag[2] = 1:
        flag[self =1] ← 2
        await(if flag[2] = 4)
flag[self = 1] ← 4
//await( flag[1..self-1] ∈ {0, 1})

process 1 in Critical Section

# EXIT PROTOCOL
await( flag[2] ∈ {0, 1, 4})
flag[self = 1] ← 0
```

```
# ENTRY PROTOCOL process 2
flag[self =2] ← 1
await( flag[1] ∈ {0, 1, 2})
flag [self = 2] ← 3
if flag[1] = 1:
        flag[self =2] ← 2
        await( flag[1] = 4)
flag[self = 2] ← 4
await( flag[1] ∈ {0, 1})

Process 2 in Critical Section

# EXIT PROTOCOL
await( flag[1] ∈ {0, 1, 4})
flag[self = 2] ← 0
```



```
# ENTRY PROTOCOL process 1
flag[self =1] ← 1
await( flag[2] ∈ {0, 1, 2})
flag [self = 1] ← 3
if flag[2] = 1:
        flag[self =1] ← 2
        await(if flag[2] = 4)
flag[self = 1] ← 4
//await( flag[1..self-1] ∈ {0, 1})

process 1 in Critical Section

# EXIT PROTOCOL
await( flag[2] ∈ {0, 1, 4})
flag[self = 1] ← 0
```

```
# ENTRY PROTOCOL process 2
flag[self =2] ← 1
await( flag[1] ∈ {0, 1, 2})
flag [self = 2] ← 3
if flag[1] = 1:
        flag[self =2] ← 2
        await( flag[1] = 4)
flag[self = 2] ← 4
await( flag[1] ∈ {0, 1})

Process 2 in Critical Section

# EXIT PROTOCOL
await( flag[1] ∈ {0, 1, 4})
flag[self = 2] ← 0
```



- order of the "all" and "any" tests must be uniform.
- Also the "any" tests should be satisfied by a thread other than self.

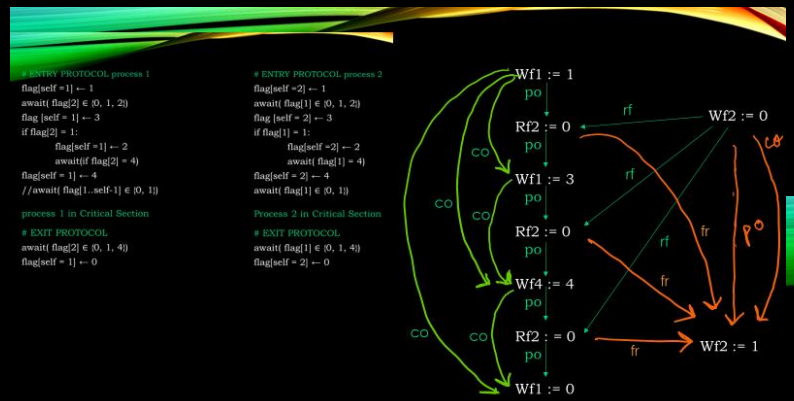
For example,

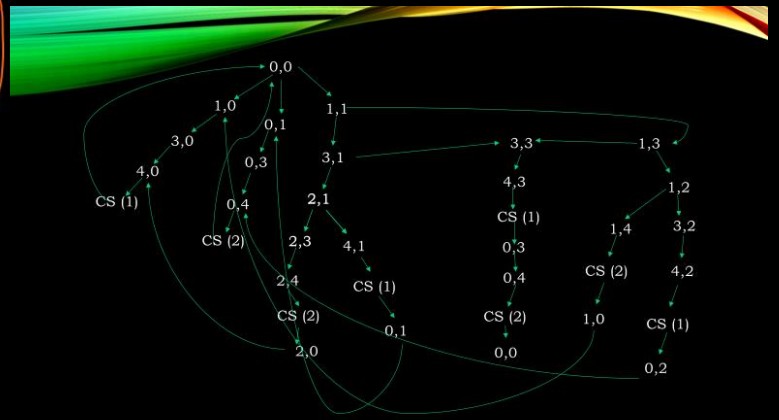
if the test is any flag[1..N] = 1 and only flag[self] = 1, then the test is said to have failed/returned 0

- The protocol,
- correctness arguments,
- n-thread version, and
- discussion about the cost of the protocol

SZYMANSKI ALGORITHM

```
# ENTRY PROTOCOL
flag[self] ← 1
await( all flag[1..N] ∈ {0, 1, 2})
flag [self] ← 3
if any flag[1..N] = 1:
        flag[self] ← 2
        await( any flag[1..N] = 4)
flag[self] ← 4
await( flag[1..self-1] ∈ {0, 1})

Critical Section

# EXIT PROTOCOL
await( all flag [self+1..N] ∈ {0, 1, 4})
flag[self] ← 0
```





- order of the "all" and "any" tests must be uniform.
- Also the "any" tests should be satisfied by a thread other than self.

For example,

if the test is any flag[1..N] = 1 and only flag[self] = 1, then the test is said to have failed/returned 0

Thank You

1.) “A simple solution to Lamport's concurrent programming problem with linear wait”
https://www.researchgate.net/publication/221235887_A_simple_solution_to_Lamport's_concurrent_programming_problem_with_linear_wait

2.) https://en.wikipedia.org/wiki/Szyma%C5%84ski%27s_algorithm

3.) CS766 https://www.cse.iitb.ac.in/~akg/courses/2021-concurrency/

4.) https://www.sarc-iitb.org/events/core-talks/

References

1.) “A simple solution to Lamport's concurrent programming problem with linear wait”

https://www.mimuw.edu.pl/~sl/teaching/13_14/PWiR/zadanie1/szymanski.88.pdf

2.) https://en.wikipedia.org/wiki/Szyma%C5%84ski%27s_algorithm

3.) The Art of Multiprocessor Programming by Nir Shavit

4.) https://www.cse.iitb.ac.in/~akg/courses/2021-concurrency/lec-concurrency-how-to-think.pdf

5.) https://www.youtube.com/playlist?list=PLssOvQUpC9clEGbrWDTbroK63tzx0Lwcu

6.) https://link.springer.com/content/pdf/10.1007%2FBFb0054187.pdf