CSCI3150 - IPC2 SEMAPHORE

Calvin Kam (hckam@cse)

AGENDA

- Race Condition
- Mutual Exclusion
- Semaphore
- Examples

WHAT IS RACE CONDITION?

- The **outcome** of an execution depends on a **particular order** in which the **shared resource** is accessed.
- May have a different result and outcome if the program runs several times. -> Non deterministic.
- we should ensure **consistency.**

TOILET EXAMPLE

- During the lunch hour, it is also a “peak” hour for the toilet usage.
- Suppose on the 1/F the toilet has no lock, the student will check whether it is occupied or not. If not, he will go inside.

EXAMPLE

Time	Person A
1	Go to the toilet
2	Check if there is any person occupying, <u>no person now</u>
3	Go back and grab the laptop
4	Go into the toilet cubicle

IF MORE THAN ONE PERSON

Time	Person A	Person B
1	Go to the toilet	
2	Check if there is any person, NO	
3	Go back and grab the laptop	Go to the toilet
4	Go into the toilet cubicle	Check if there is any person, YES
5		Leave

IF MORE THAN ONE PERSON

Time	Person A	Person B
1	Go to the toilet	
2	Check if there is any person, NO	Go to Toilet
3	Go back and grab the laptop	Check if there is any person, NO
4	Go into the toilet cubicle	Go back and grab the laptop
5		Go into the toilet cubicle
		:0)

IF MORE THAN ONE PERSON..

- Of course we need to synchronise activities!

SOLUTION?

- We need make sure only one person is allowed to use the toilet at one time.
- The toilet has a lock, the person goes in and lock it. Then unlock when he finishes.
- Another person has to wait until the toilet is unlocked.

TERMINOLOGY

Shared Object

Lock

Mutual Exclusion

CODE TO SIMULATE STUDENT A

```
int fd;
    printf("A arrive toilet.\n");
    printf("A checks the toilet.\n");
    fd=open("toilet", O_CREAT|O_RDWR|O_APPEND, 0777);
```

Shared Object

```
if(lseek(fd,0,SEEK_END)==0){
        printf("A goes back and grab a laptop...\n");
        sleep(2);
         write(fd,"lock ",5);
        printf("A enters the toilet.\n");
        if(lseek(fd,0,SEEK_END)>5)
            printf("Oops! We have two persons in toilet\n");
    }else{
        printf("Toilet Occupied, A leaves the toilet\n");
    }
```

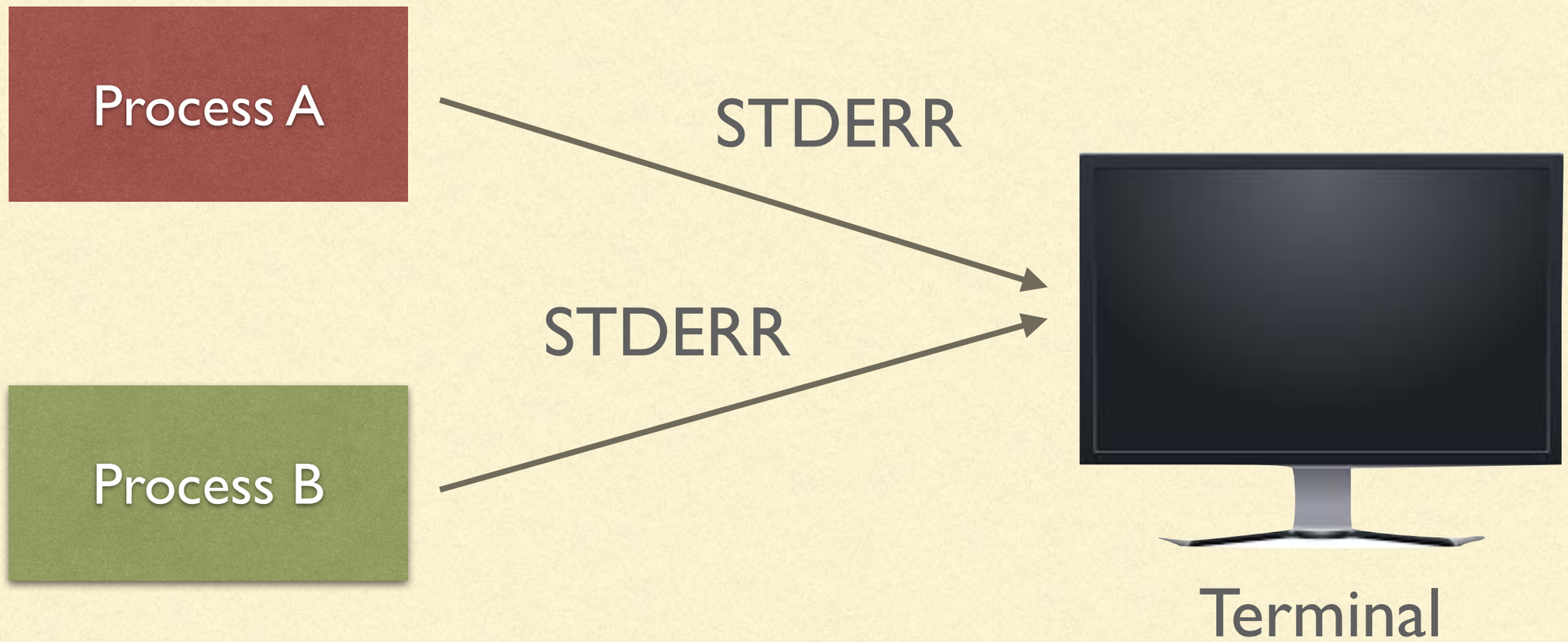
If two processes access the shared object

CODE TO SIMULATE STUDENT B

```
printf("B arrive toilet.\n");
 sleep(rand()%2+1);
 printf("B checks the toilet.\n");
 fd=open("toilet", O_CREAT|O_RDWR|O_APPEND, 0777);
```

To ensure different execution flow

PROBLEM 2 : SHARED OUTPUT



PROBLEM 2 : SHARED OUTPUT

- Two processes should access the standard error exclusively.
- Otherwise it will mess up the output like this.

```
Thhiis sis  CiSCsI 3C15S0-C-IAn3 1op5er0a-t-iAnng  soypsteemr actoiurnseg.
 system course.
```

SOLUTION: SEMAPHORE

- Acts like a signal.
- It tells the process how many resource available and maintains the wait list.
- Operates by incrementing/decrementing a value.
- If the current value of semaphore is zero, and is decrementing, the process will be blocked until it rises above zero.

SEMAPHORE IN CODING

<semaphore.h>

sem_open	Opens/creates a named semaphore for use by a process
sem_wait	Wait on a semaphore
sem_post	Signal a semaphore
sem_close	Deallocates the specified named semaphore
sem_unlink	Removes a specified named semaphore

SOLUTION: TOILET PROBLEM

- We use semaphore as a mutex (lock).
- If a student wants to get into the toilet, we wait on a semaphore (decrement - -)
- After finish, we unlock the semaphore (increment ++).
- This can ensure only one student at a time touching the toilet.

SOLUTION: TOILET PROBLEM

```
…
#include <semaphore.h>

int main(int argc, char * argv[]) {
    int fd;
    int value = 1;
    sem_t * mutex;
    mutex = sem_open("mutex", O_CREAT, 0666, value);
```

Header

Open the semaphore object, create if not exist.

SOLUTION: TOILET PROBLEM

```
printf("A arrive toilet.\n");
    sem_wait(mutex);
    printf("A checks the toilet.\n");
    fd=open("toilet", O_CREAT|O_RDWR|O_APPEND, 0777);
    if(lseek(fd,0,SEEK_END)==0){
        printf("A goes back and grab a laptop...\n");
        sleep(2);
        write(fd,"lock ",5);
        printf("A enters the toilet.\n");
        if(lseek(fd,0,SEEK_END)>5)
            printf("Oops! We have two persons in toilet\n");
    }else{
        printf("Toilet Occupied, A leaves the toilet\n");
    }
    close(fd);
    sem_post(mutex);
    sem_close(mutex);
    sem_unlink("mutex");
    return 0;
```

Decrement

Increment

SOLUTION: SHARED OUTPUT

```
/*process-1.c*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>
#include <unistd.h>
int main(int argc, char * argv[]

        char * c = "This is CSCI3
    // specify no buffering for s
    setbuf(stderr, NULL);

        while (* c != '\0') {
            fputc(* c, stderr);
            c++;
            sleep(1);
        }
    return 0;
}
```

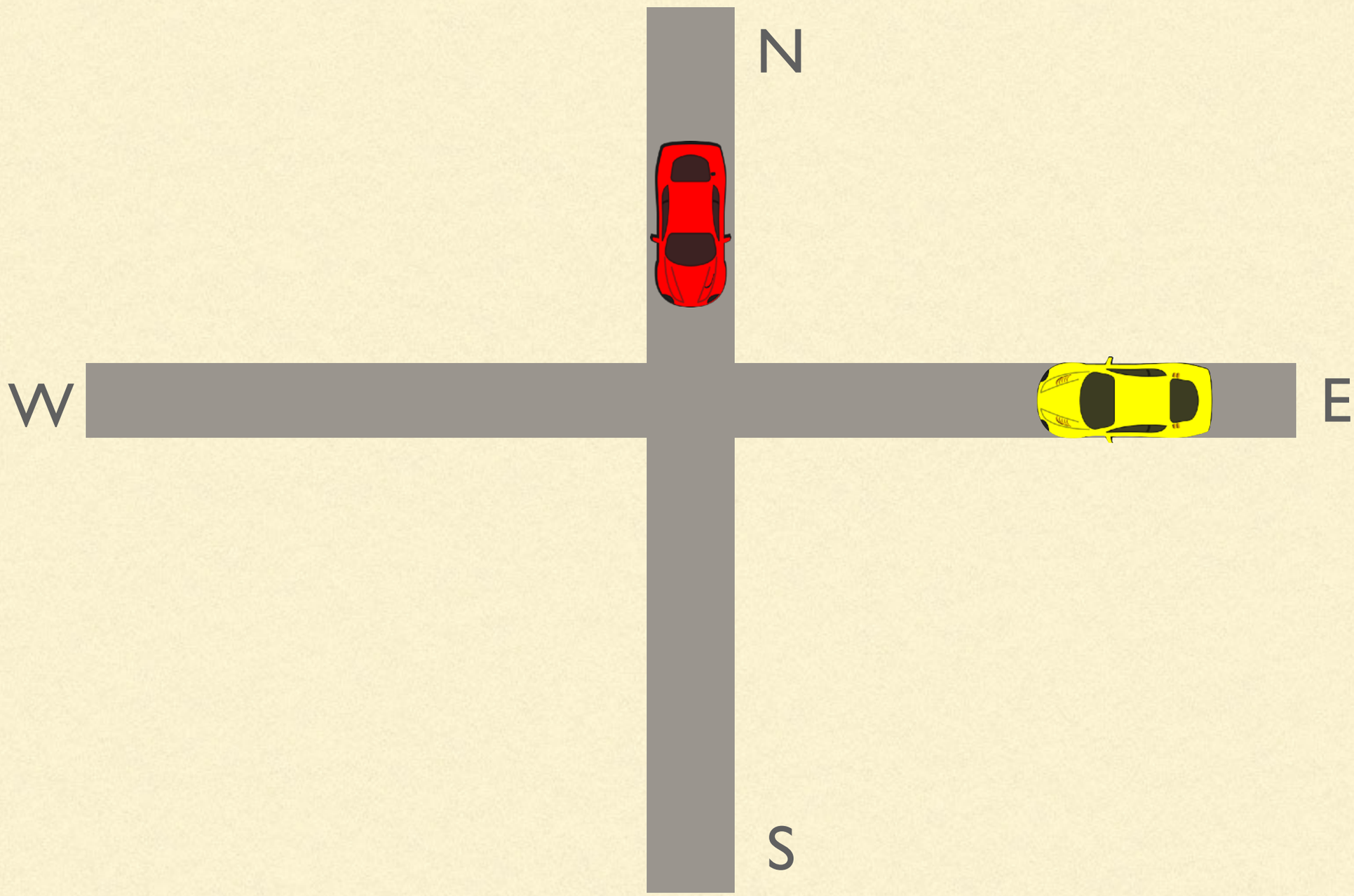
```
sem_t * mutex;
mutex = sem_open("mutex_for_stderr",
O_CREAT, 0666, 1);
sem_wait(mutex);
```

SOLUTION: SHARED OUTPUT

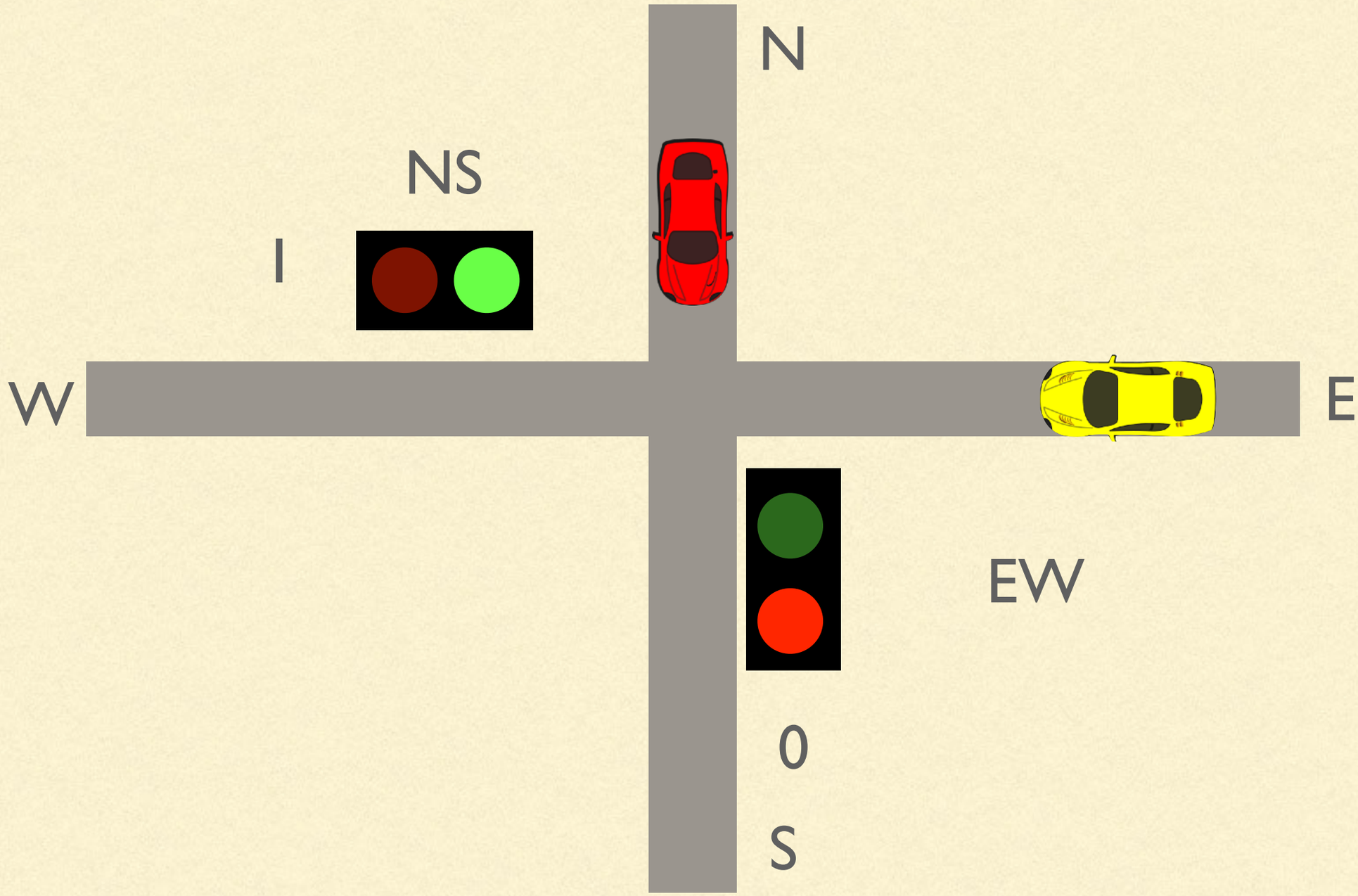
```
/*process-1.c*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <semaphore.h>
#include <unistd.h>
int main(int argc, char * argv[])

      char * c = "This is CSCI315
    // specify no buffering for st
    setbuf(stderr, NULL);

      while (* c != '\0') {
            fputc(* c, stderr);
            c++;
            sleep(1);
        }
    return 0;
}
```

```
sem_post(mutex);
    sem_close(mutex);
    sem_unlink("mutex_for_stderr");
```

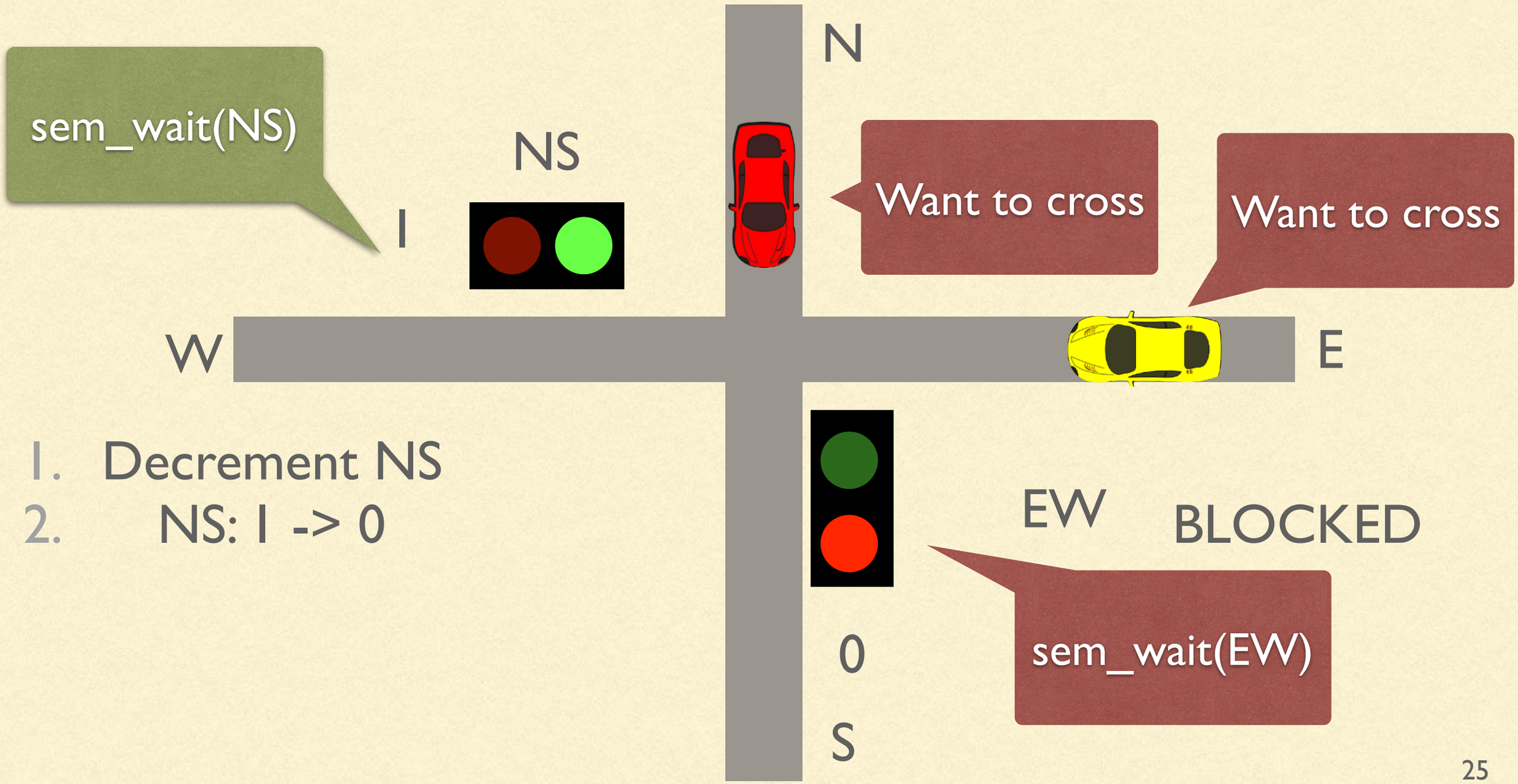
TRAFFIC CONTROL



TRAFFIC CONTROL: SEMAPHORE

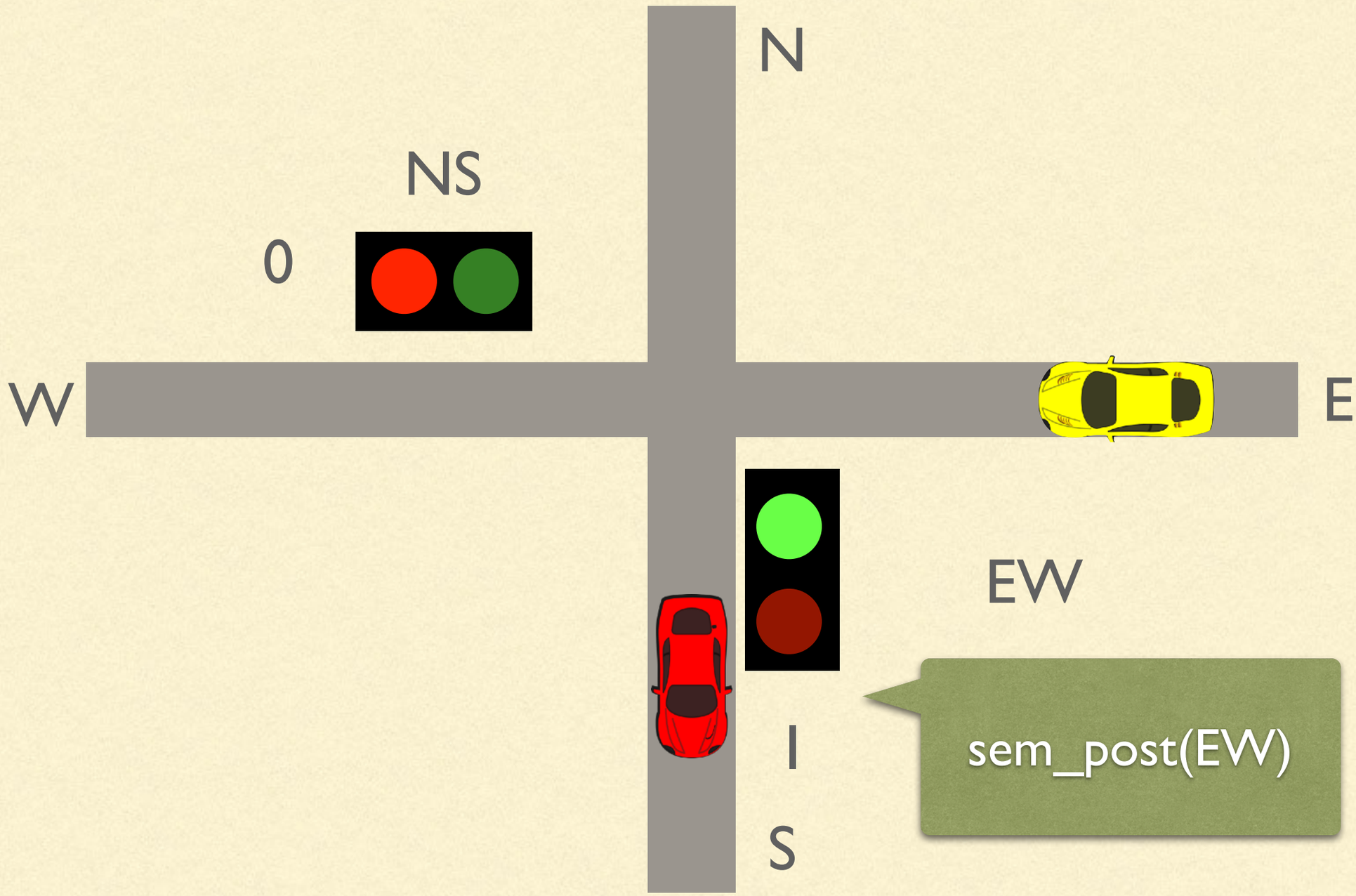


TRAFFIC CONTROL: SEMAPHORE

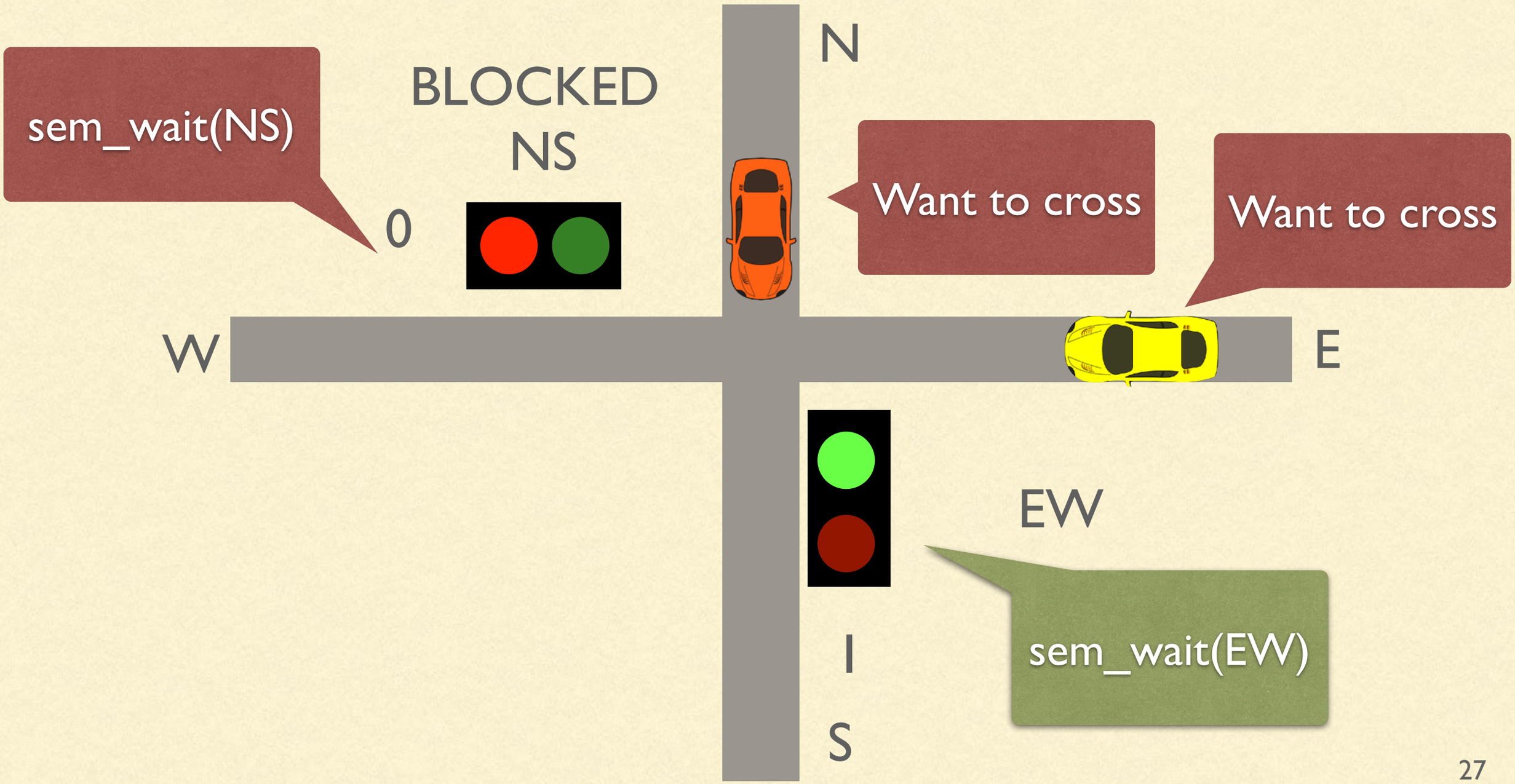


1. Decrement NS
2. NS: 1 -> 0

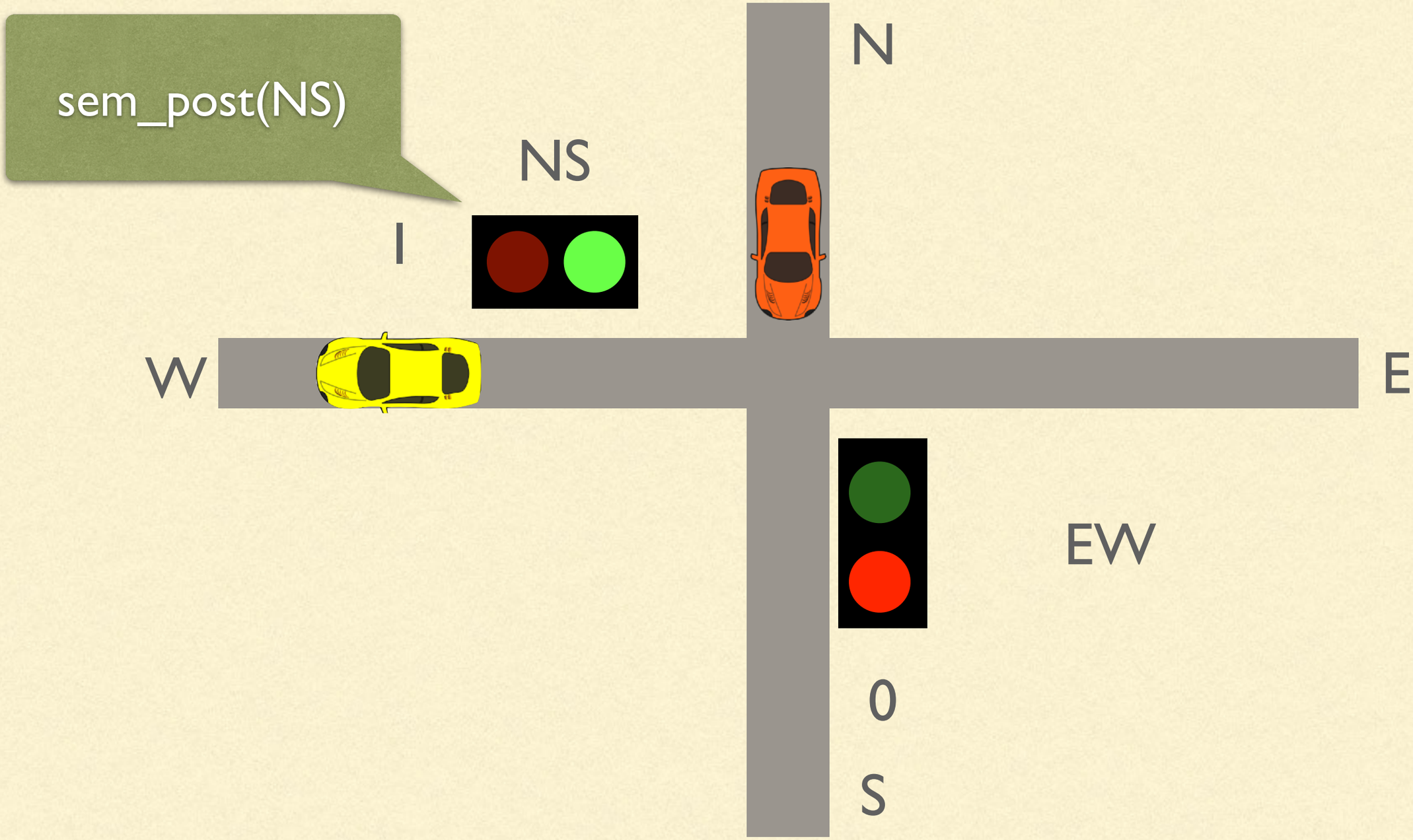
TRAFFIC CONTROL: SEMAPHORE



TRAFFIC CONTROL: SEMAPHORE



TRAFFIC CONTROL: SEMAPHORE



SUMMARY

- Race condition
- Mutual Exclusion, Mutex
- Semaphore