Lecture 06: Mutual Exclusion

COSC 273: Parallel and Distributed Computing

Spring 2023

This Week

1. Written Homework 01 Due Friday
    - can work in groups of up to 3
2. Today/Wednesday: Mutual Exclusion (pen & paper)
3. Friday: Locality of Reference (bring laptop)

Outline

1. Counter Example
2. Mutual Exclusion

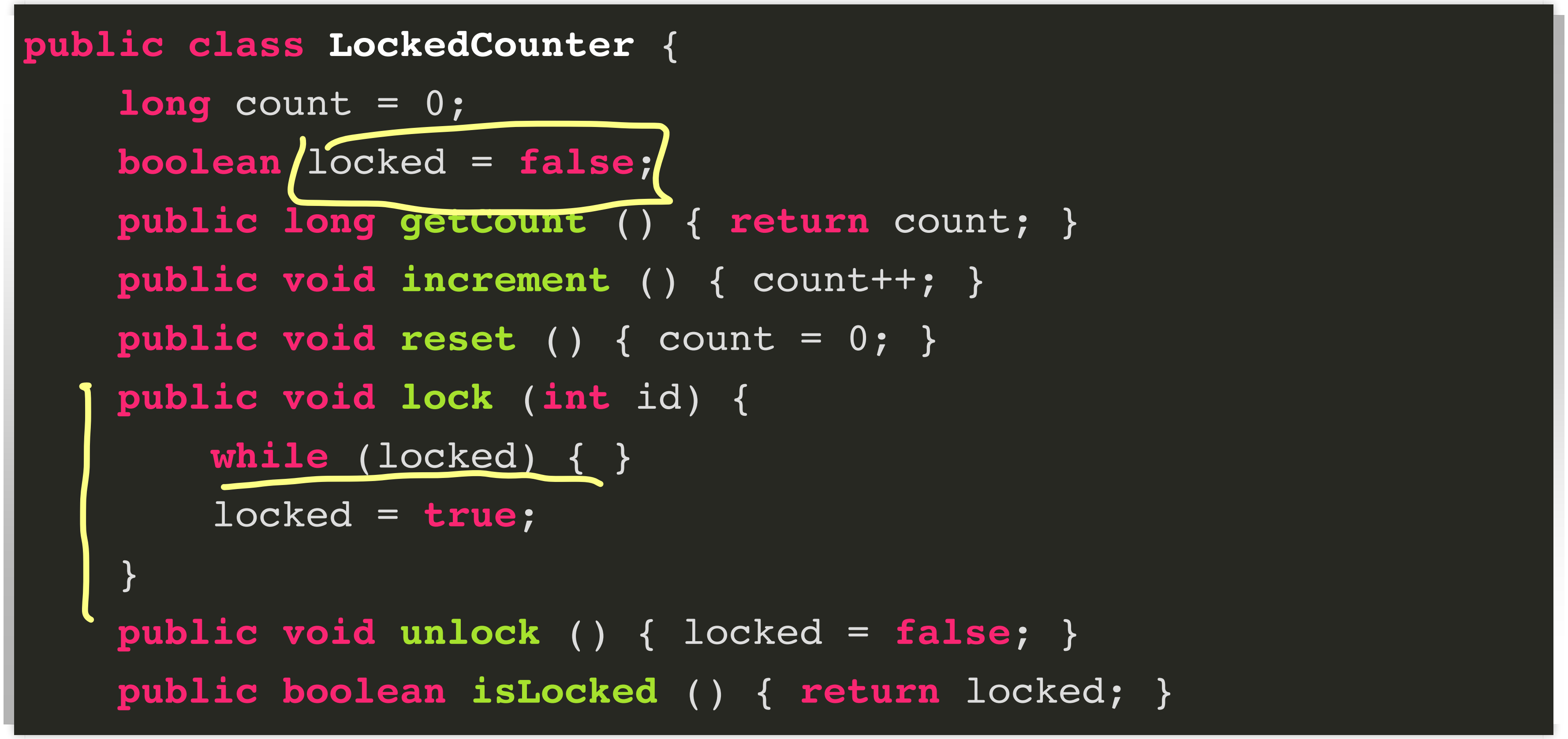
Proposed Last Time

Fix `Counter` issue by *locking* the `count`

To increment the `Counter`:

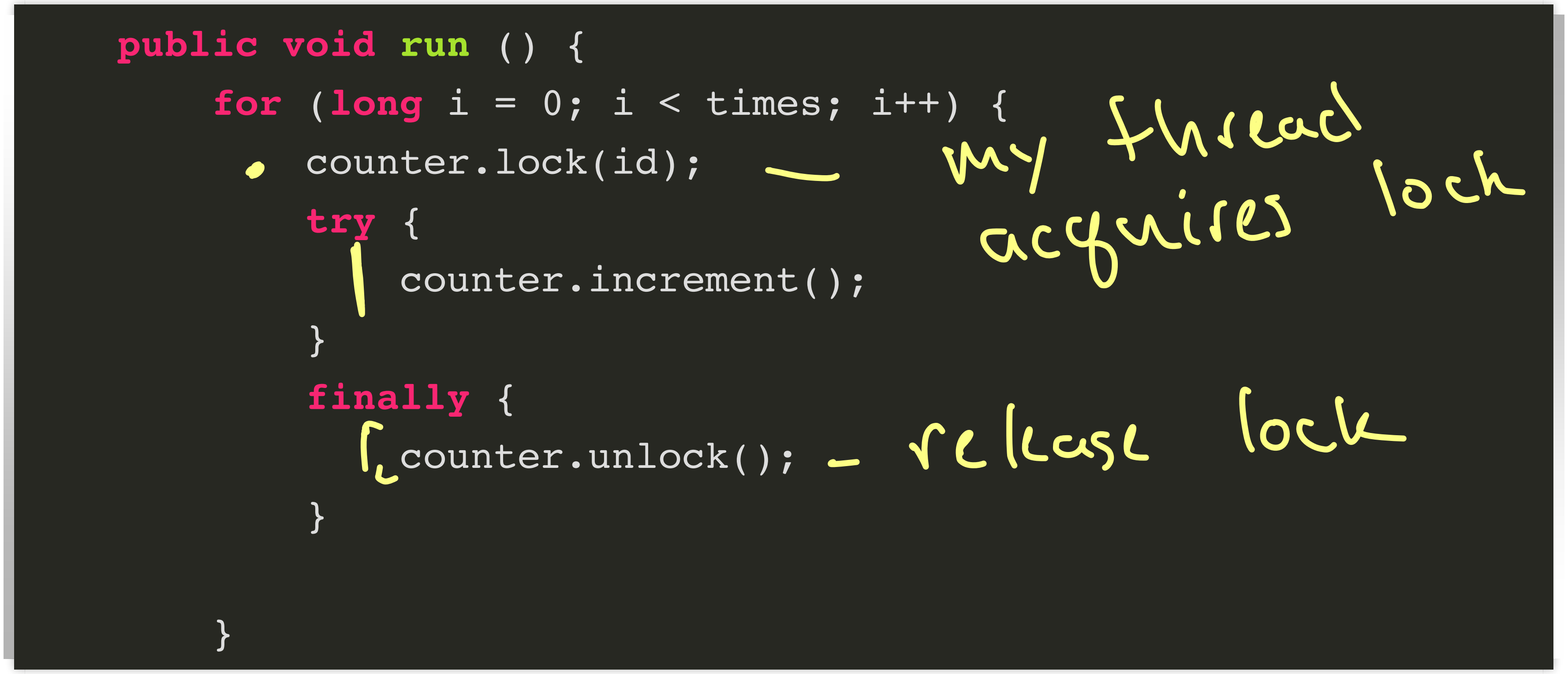
1. check if `Counter` is locked
   - if so, wait until it is unlocked
2. lock the `Counter`
   - no other thread can modify while locked
3. increment the counter
4. unlock the `Counter`

An Attempt



```
public class LockedCounter {
    long count = 0;
    boolean locked = false;
    public long getCount () { return count; }
    public void increment () { count++; }
    public void reset () { count = 0; }
    public void lock (int id) {
        while (locked) { }
        locked = true;
    }
    public void unlock () { locked = false; }
    public boolean isLocked () { return locked; }
```

Running the Locked Counter



```
public void run () {
    for (long i = 0; i < times; i++) {
        counter.lock(id);
        try {
            counter.increment();
        }
        finally {
            counter.unlock();
        }

    }
```

my thread acquires lock

release lock

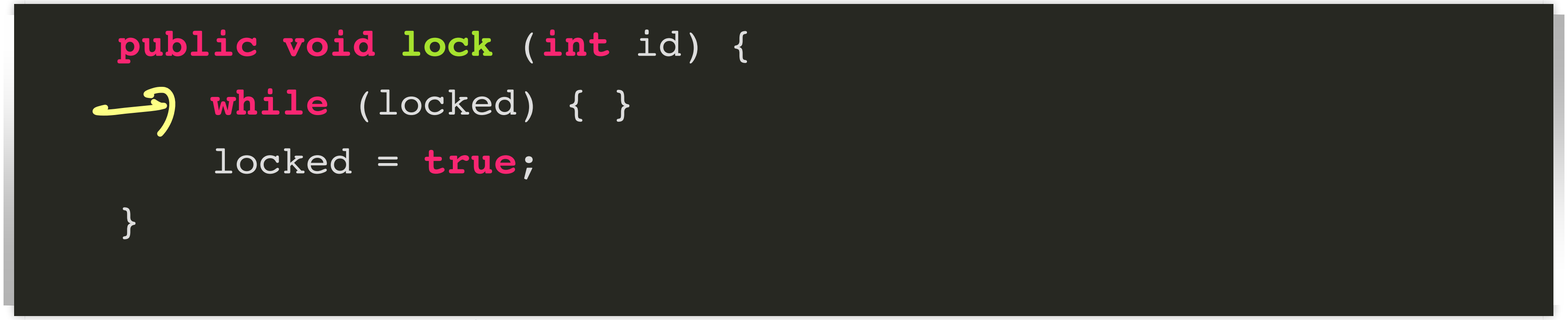
Will It Work?

`LockedCounterTester` Demo!

Question

What happened?

Init: locked = false



```
public void lock (int id) {
    while (locked) { }
    locked = true;
}
```

	T0	T1
false ←	read locked	
		read locked → false
	write true	
		write true

Morals

1. Empirical testing is not enough!
2. Must understand correctness **formally**

Correct Behavior

If multiple threads try to increment at a time:

- exactly one thread gets to increment at a time
- other threads wait until increment completed

Terminology

We want our `Counter` to satisfy **mutual exclusion**.

A Parable

A Shared Resource

- Professor (Scott) Alfeld and I are neighbors
- For purposes of today’s lecture, say we share a backyard
- We have dogs: Finnnegan (my dog), Ruple (Scott’s dog)
- Sadly, our dogs don’t get along
  - they used to, but not anymore
  - we don’t know why

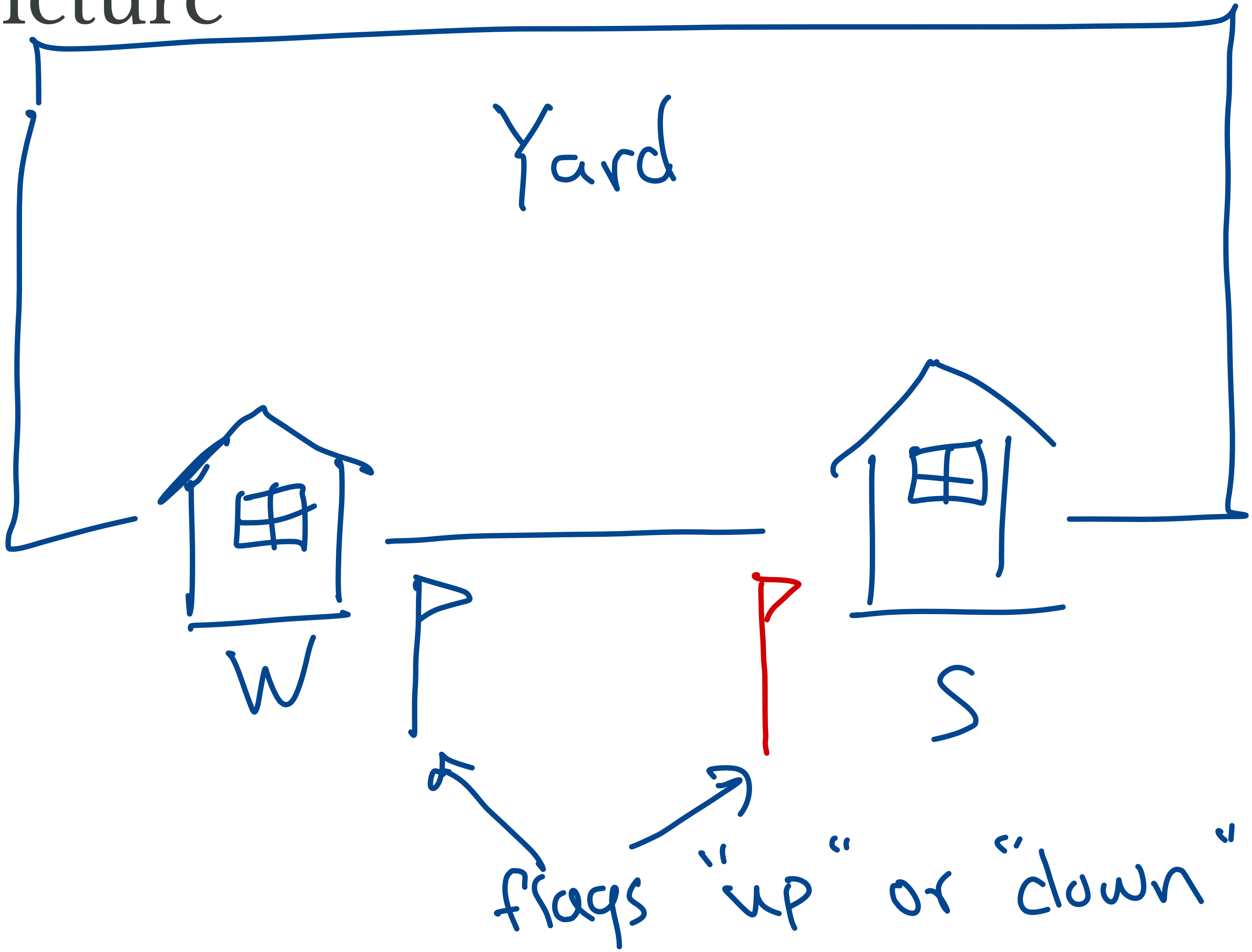
Finn and Ruple

A Question

How can Scott and Will ensure that we don’t let Finn and Ruple out in the yard at the same time?

- don’t like bothering each other with a text/phone call
- shouldn’t have to “actively” communicate unless both dogs need to go out
- have a way of passively signaling intent
  - use flags!
  - each has a flag that can be raised or lowered
  - can see the state of each other’s flags

A Picture



Our Goals

Safety Goal:

- Both dogs are not simultaneously out in the yard
  - *mutual exclusion* property

Liveness Goal:

- If ~~both~~ a dog~~s~~ need to go outside, eventually one does
  - *deadlock-freedom* property

Note: getting mutual exclusion and deadlock-freedom separately is easy!

A First Protocol: Flag if Out

1. Look to see if other flag is raised.
   - if so, wait until not raised
2. If not, raise flag then let dog out
3. When dog comes in, lower flag

Question. Does this work?

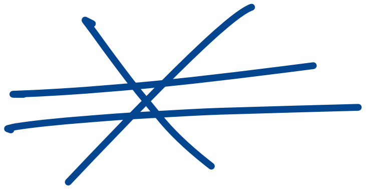
No :

(1) both look at flags

- see flag down

(2) both raise flags

(3) both let dogs out



no <u>Mutex</u>

A Bad Execution:

1. We both look at (approximately) the same time and see others’ flag is down.
2. We both raise flags as (approximately) the same time.
3. We both let dogs out at the same time.

A Second Protocol: Flag Intent

1. Raise flag.
2. Check if other flag is up
   - if so, wait until not raised
3. If other flag is down, let dog out!
4. When dog returns, lower flag.

Question. Does this work?

(1) Both raise

(2) Both wait .... indefinitely

No deadlock freedom

(deadlock!)

Another Bad Execution

1. Both raise flag at (approximately) same time.
2. Both see other’s flag raised.
3. Both wait... neither dog ever goes outside!

More Generally

Both protocols are **symmetric**

- Scott and Will behave the same way according to what we see

Can a symmetric protocol possibly work?

For Any Symmetric Protocol

Suppose we act simultaneously:

1. start in same state
2. perform same action
3. see that other performed same action
4. respond in same manner
5. ...

This continues indefinitely, so either

- we both let dogs out at same time, or
- neither dog goes out ever

Apparently

We need an asymmetric protocol

- Under contention, give Finn priority
    - Scott agrees with this

Note. *Symmetry breaking* is a common theme in parallel/distributed computing.

Third Protocol

Separate protocols for Will and Scott

Will’s Protocol

When Finn needs to go out:

1. Raise flag
2. While Scott’s flag is raised, wait
3. Let Finn out
4. When Finn comes in, lower flag

Scott’s Protocol

When Ru needs to go out:

1. Raise flag
2. While Will’s flag is raised:
    1. lower flag
    2. wait until Will’s flag is lowered
    3. raise flag
3. When Scott’s flag is up and Will’s is down, release Ru
4. When Ru returns, lower flag

Does Third Protocol Work?

- Do we get mutual exclusion?
- Do we get get deadlock-freedom?

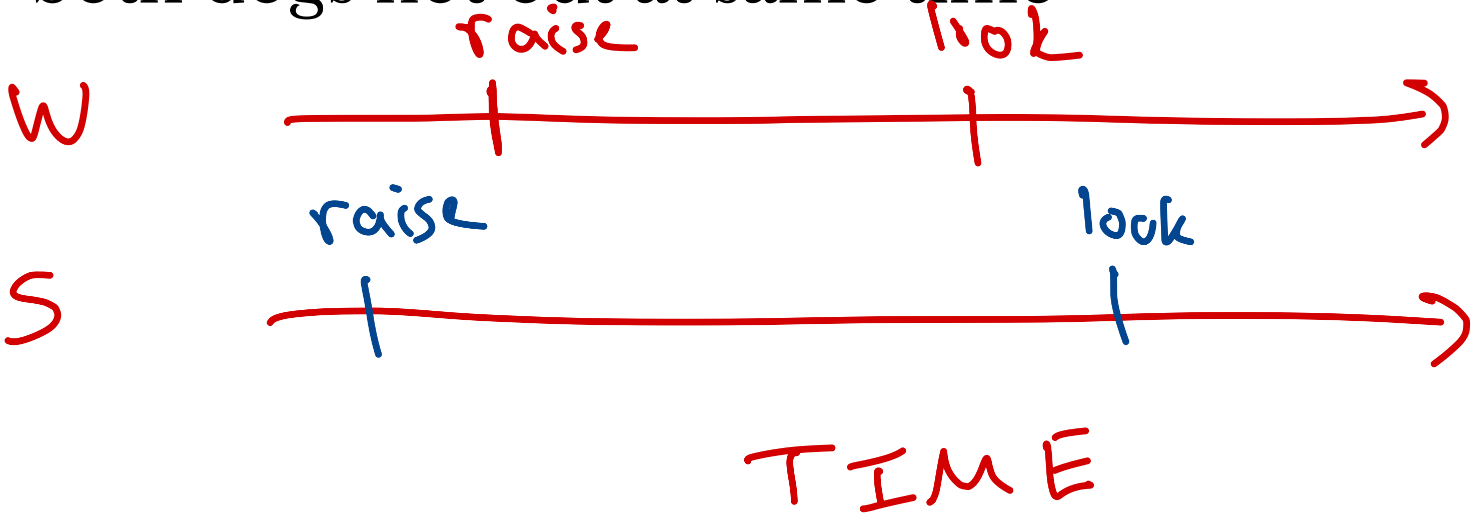
Crucial Insight

If both Scott and Will:

1. raise flag, then
2. look at other's flag

then at least one of us will see other flag raised

- always check other's flag *before* letting dog out
- both dogs not out at same time



More Formally

If we want to *prove* mutual exclusion property

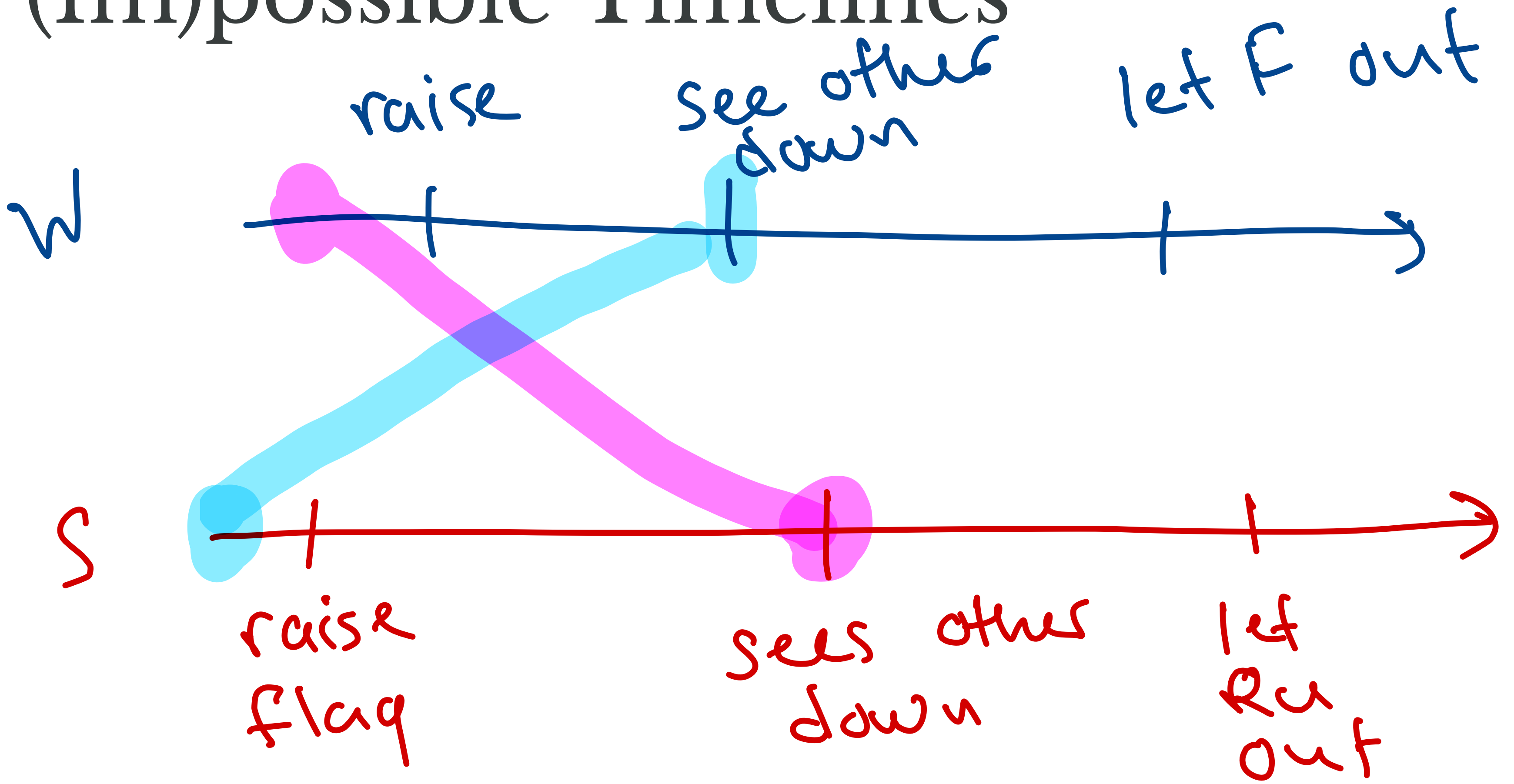
- argue by contradiction
- suppose that at some time, both dogs were out
- what could have led us there?

Property of Both Protocols

Before letting a dog out, both Scott and Will do:

1. raise flag
2. see other's flag down
3. let dog out

(Im)possible Timelines



Conclusion

If both Finn and Ru are in yard at same time, Will or Scott must not have followed the protocol!

- This establishes mutual exclusion property.

What About Deadlock-Freedom?

If both Finn and Ruple want to go out

1. Both Will and Scott raise flags
2. Eventually, Scott sees Will's flag
    - lowers his flag (sorry Ru)
3. Eventually, Will sees Scott's flag down
4. Finn goes out!

Nice!

This protocol gives mutual exclusion and deadlock-freedom...