# CSCE 313-200
# Introduction to Computer Systems
# Spring 2024

## Deadlocks

Dmitri Loguinov
Texas A&M University

March 8, 2024

# Chapter 6: Roadmap

6.1 Principles

6.6 Dining philosophers

6.2 Prevention

6.3 Avoidance

6.4 Detection

6.5 Integrated strategies

6.7 Unix

6.8 Linux

6.9 Solaris

6.10 Windows

Part II

- Chapter 3: Processes
- Chapter 4: Threads
- Chapter 5: Concurrency
- Chapter 6: Deadlocks

# Principles

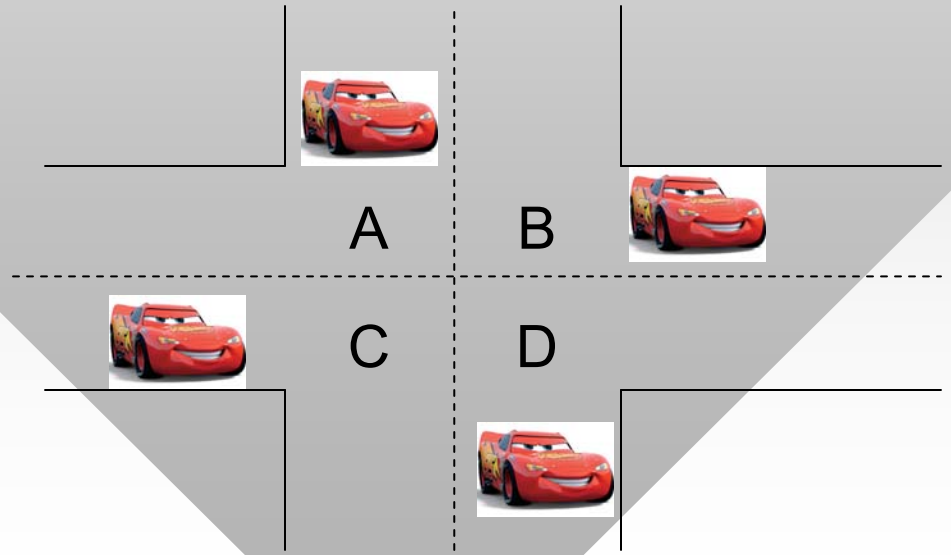
```
ThreadP () {
    mutexA.Lock();
    mutexB.Lock();
    // critical section
    mutexA.Unlock();
    mutexB.Unlock();
}
```

```
ThreadQ () {
    mutexB.Lock();
    mutexA.Lock();
    // critical section
    mutexB.Unlock();
    mutexA.Unlock();
}
```

- Deadlock is a permanent (infinite) wait for resources
  - Important problem in the field of synchronization
- Typical example with threads P and Q:
  - Two mutexes locked in different order
  - Common source of deadlocks
- Another example:

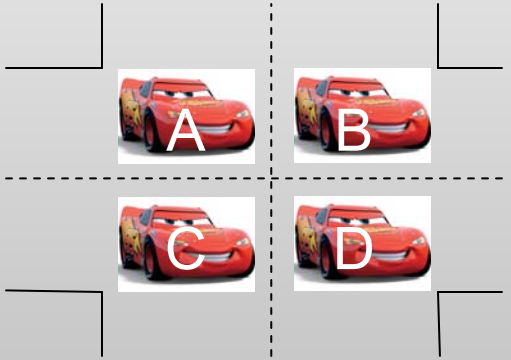
```
CarNorth () {
    mutexA.Lock();
    mutexC.Lock();
    // drive
    mutexA.Unlock();
    mutexC.Unlock();
}
```

```
CarWest () {
    mutexC.Lock();
    mutexD.Lock();
    // drive
    mutexC.Unlock();
    mutexD.Unlock();
}
```



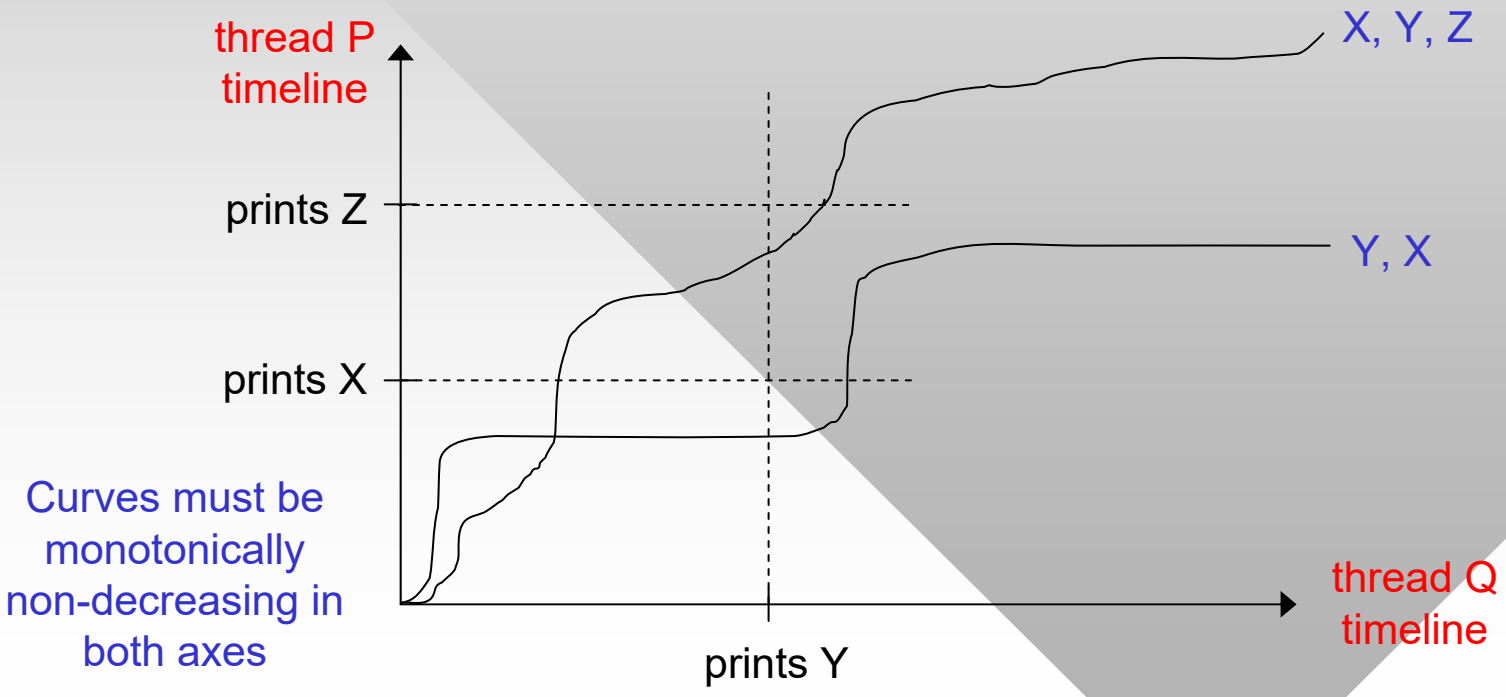
# Principles

- Example (cont'd): deadlock possible in general and...
  - Certain when each grabs their first mutex:
- Conditions for a deadlock to be possible
  - 1) Mutual exclusion (no sharing)
  - 2) Hold and wait (allowed to hold one resource and wait for another, i.e., acquisition of multiple mutexes is *not* atomic)
  - 3) No preemption (held resources not released until critical section has been successfully completed)
- Conditions for it to be certain
  - 1)-3) plus 4) circular wait



# Progress Diagram

- Assume two threads P and Q in parallel execution
  - Denote by t the absolute time
  - Progress diagram is a 2D parametric curve (x(t),y(t)) where x(t) is the number of instructions executed by Q and y(t) by P

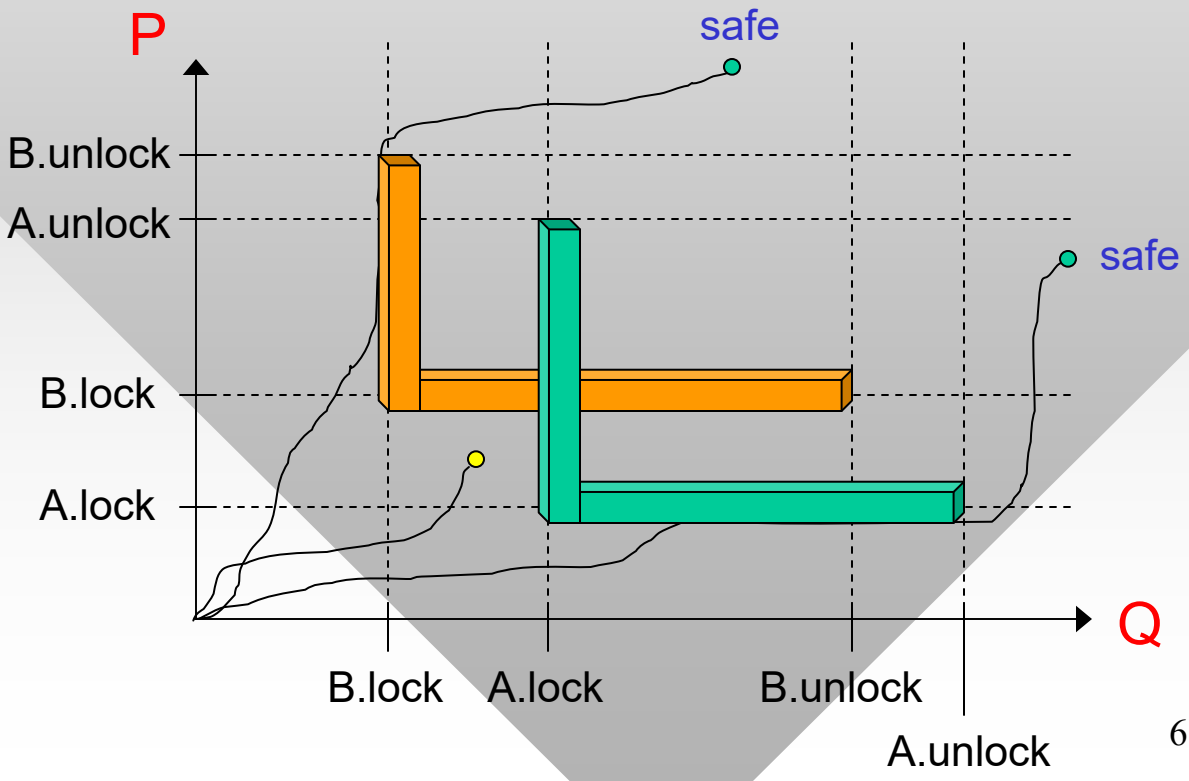


# Progress Diagram

- Back to our example with P and Q
- Mutexes place L-shaped obstacles/barriers on the progress diagram that cannot be crossed

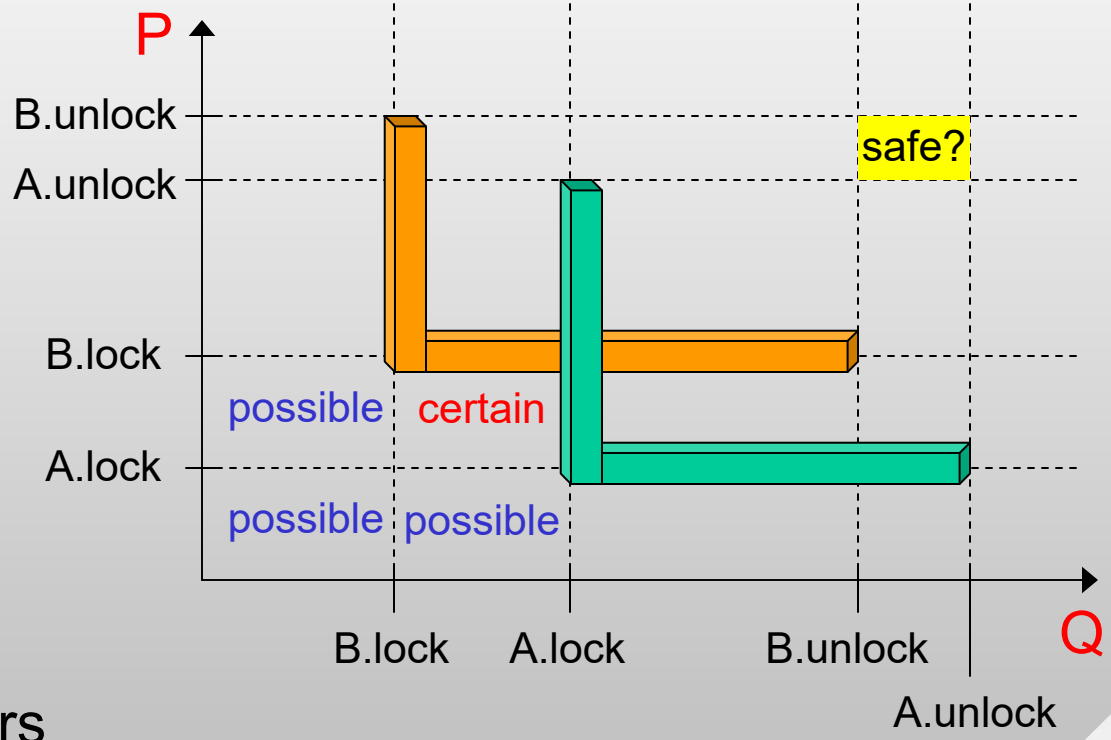
```
ThreadP () {
    mutexA.Lock();
    mutexB.Lock();
    // critical section
    mutexA.Unlock();
    mutexB.Unlock();
}
```

```
ThreadQ () {
    mutexB.Lock();
    mutexA.Lock();
    // critical section
    mutexB.Unlock();
    mutexA.Unlock();
}
```



# Progress Diagram

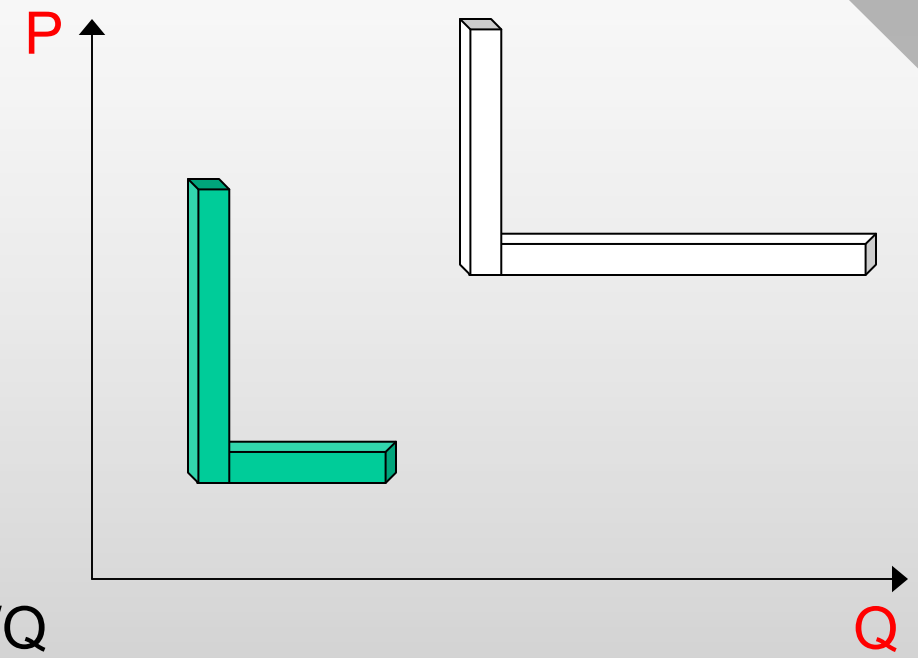
- In three quadrants near the origin, deadlock possible
  - In one, it is certain
- All other sections are safe
  - Except impossible states behind barriers

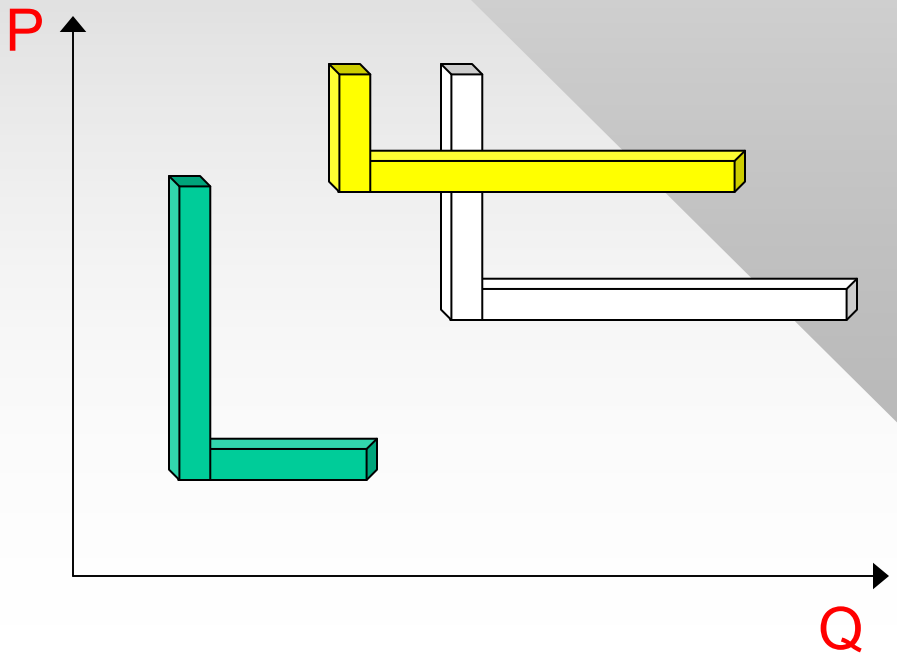


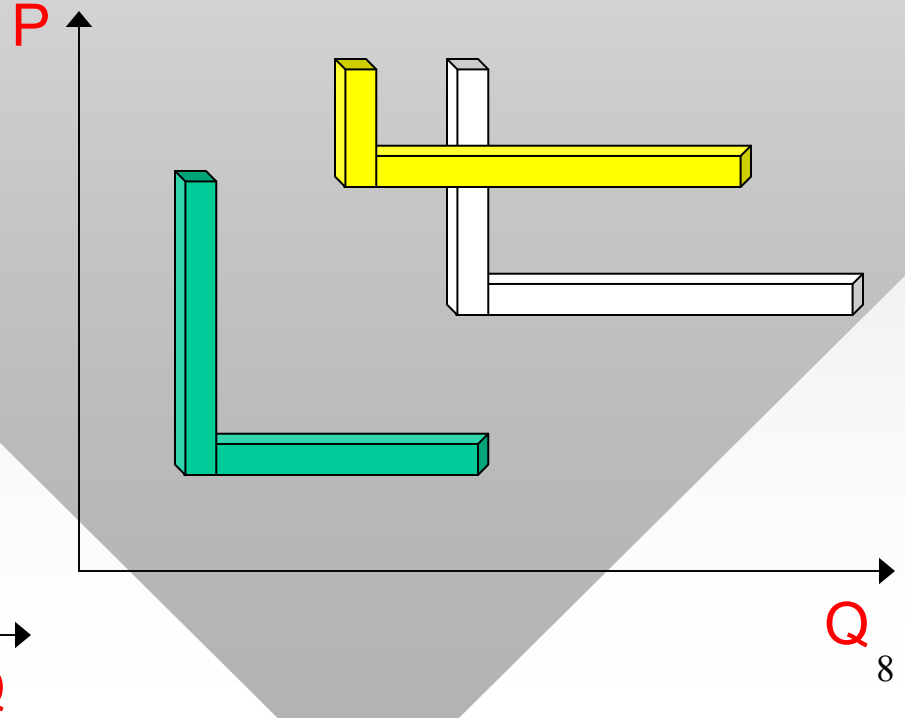

- Static or dynamic analysis to detect deadlocks
- What happens with N threads?
  - N-dimensional diagram

# Progress Diagram

- How about these diagrams?
- In what order are mutexes acquired?
  - Write pseudo code for P/Q



# Resource Allocation Graph

- To visualize deadlocks, often a graph is drawn between all threads and resources
  - Edges of this bipartite graph are labeled with “held by” (resources → threads) and “wants” (threads → resources)
- If this directed graph has a cycle, there is a deadlock
  - Car labels (N, E, W, S) map to North/East/West/South position