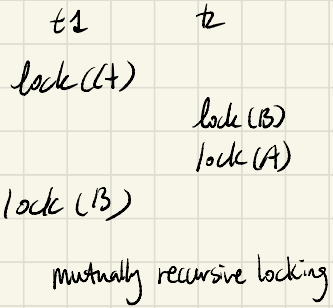
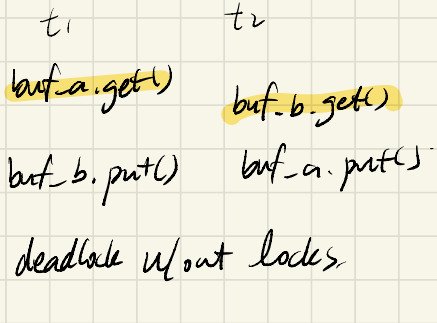
10/25/23 Deadlocks

→ cycle of waiting threads blocked on each other





resource = producing & consuming power

→ Necessary Conditions For Deadlock

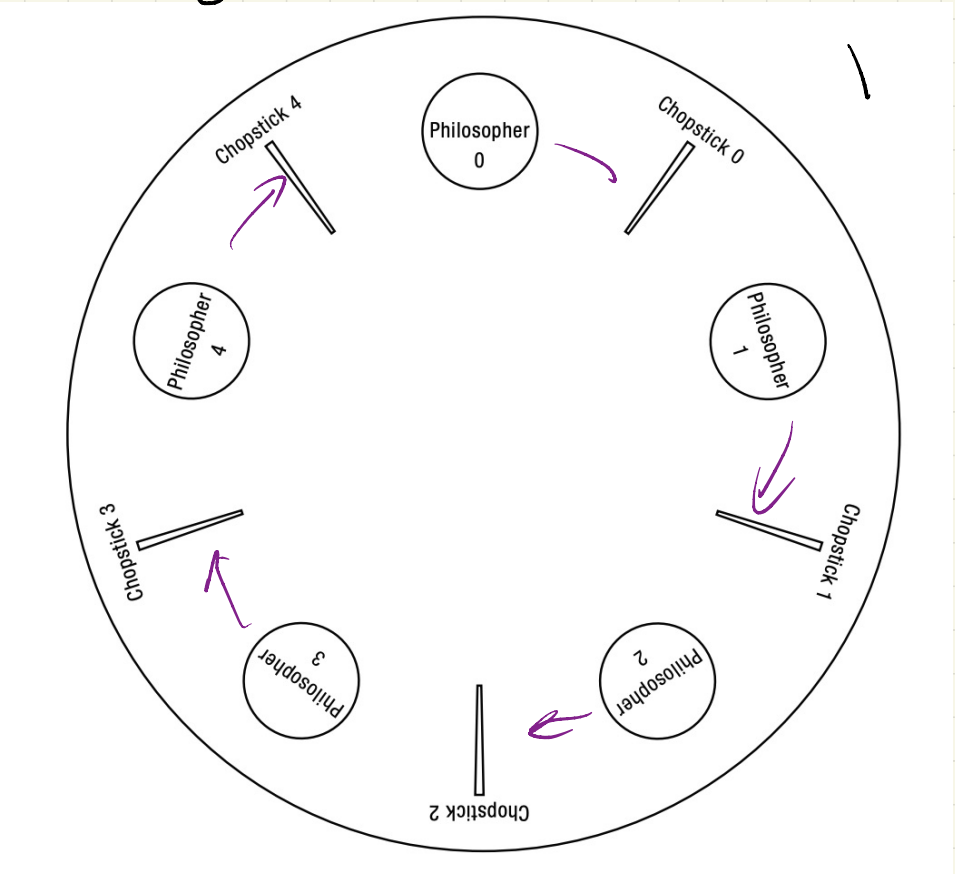
① Bounded Resources : finite instances of each resource

② No Preemption : resource can't be forcibly taken away

③ Hold & wait : hold on to resource while waiting

④ Circular wait : cycle of waiting

Dining Philosophers



5 philosophers

5 chopsticks (resource)

each needs 2 chopsticks to eat

→ grab one from left & grab one from right

→ Necessary condition for Deadlock but not sufficient for deadlock.

- single instance ⇒ sufficient
- multi instance ⇒ not sufficient (+1 to chopstick)

What to do? (limit code behavior)

① Prevention: when writing code, break one of the 4 conditions for deadlock.

② Avoidance: system controls resource access & scheduling to avoid ↑

③ Detection: let things be, detect & recover when there's deadlock.

Deadlock Prevention

Bounded resources: provide sufficient resources (reserve some resources to deal w/ cases before running out of resources)

No Preemption: have preemption

Hold & wait: release while wait

Circular wait: ordering.

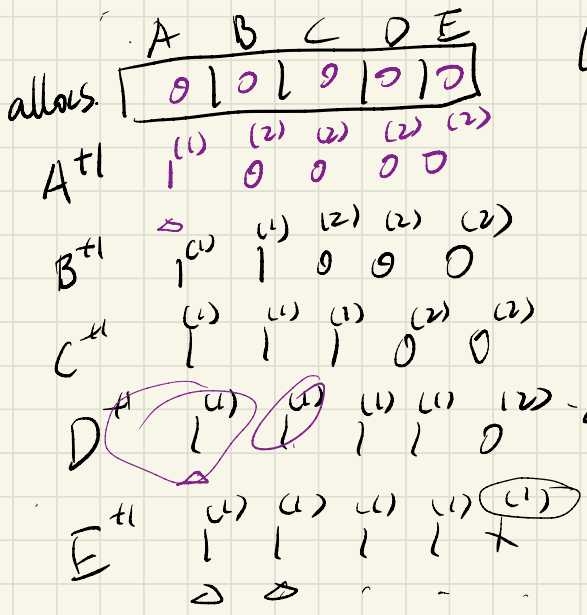
Locking ordering

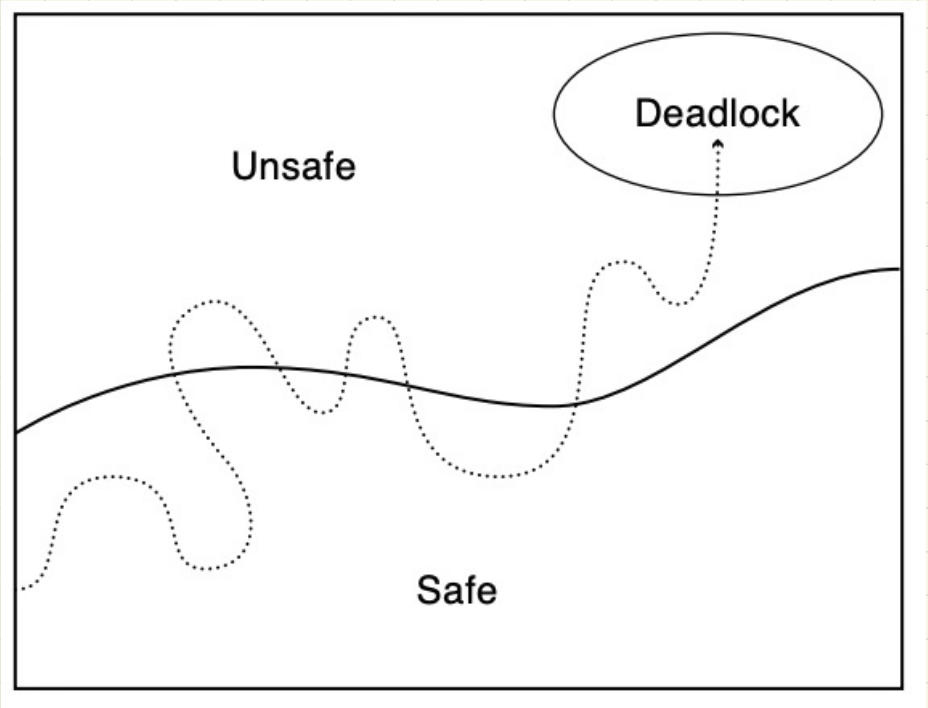
Deadlock Avoidance

→ system grants access to resource, can delay fulfilling requests

→ needs to know maximum resources each thread needs.

★ have a way to grant all max requests = Safe.




Deadlock
Unsafe
Safe