Previously, on CS4410...

What are the implications?

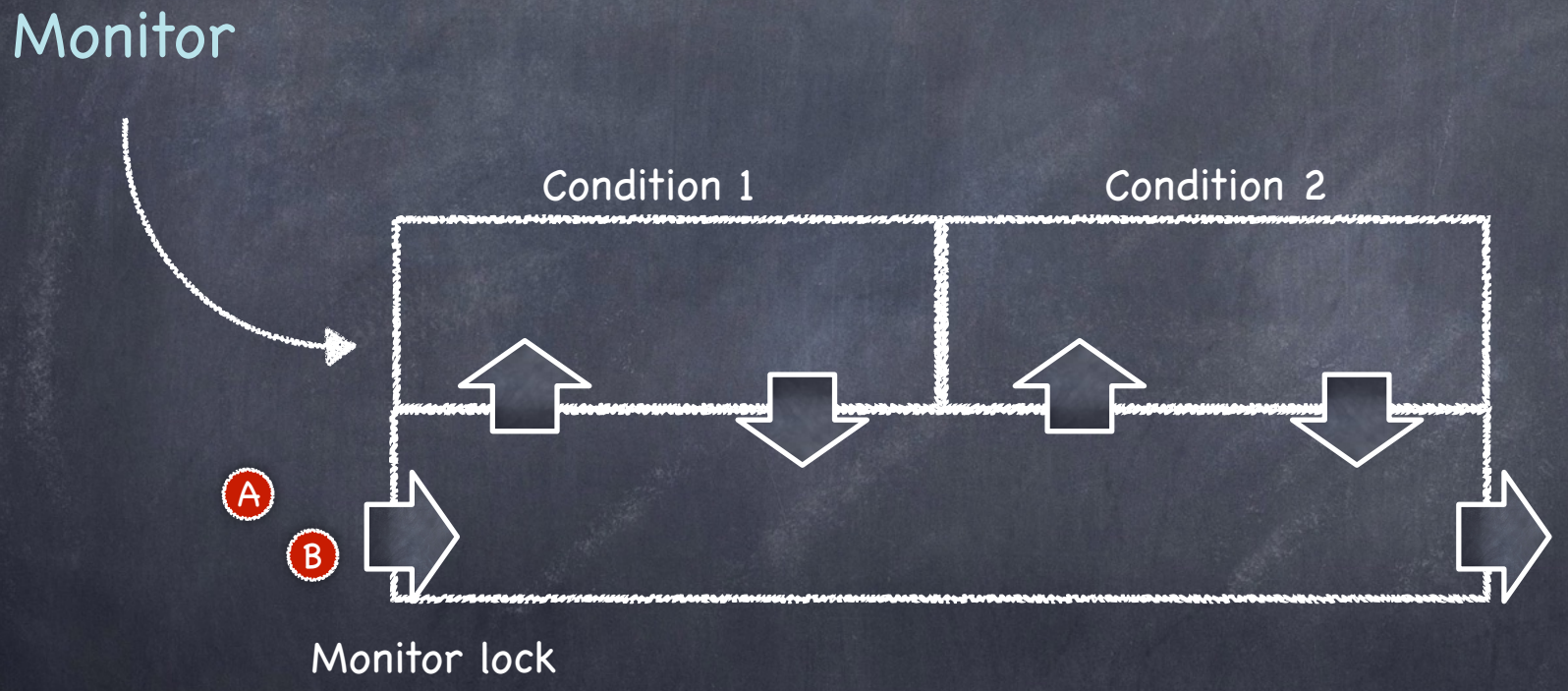
Hoare

- Signaling is atomic with the resumption of waiting thread
  - shared state cannot change before waiting thread is resumed!
  - safety requires to signal only when condition holds
- Shared state can be checked using an if statement
- Makes it easier to prove liveness
- Tricky to implement

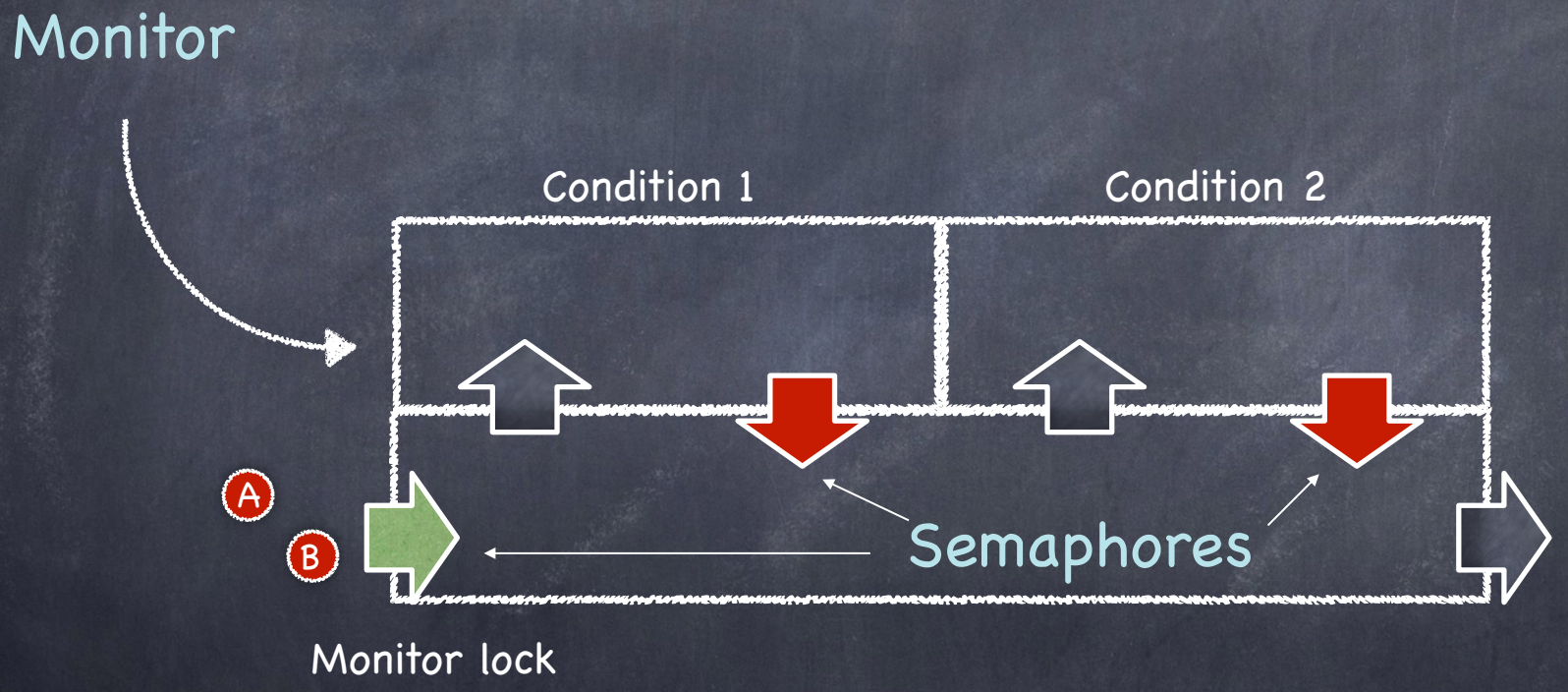
Mesa

- notify() and notifyAll() are hints
  - adding them affects performance, never safety
- Shared state must be checked in a loop (the condition could have changed since the thread was notified!)
- Simple implementation
- Resilient to spurious wakeup

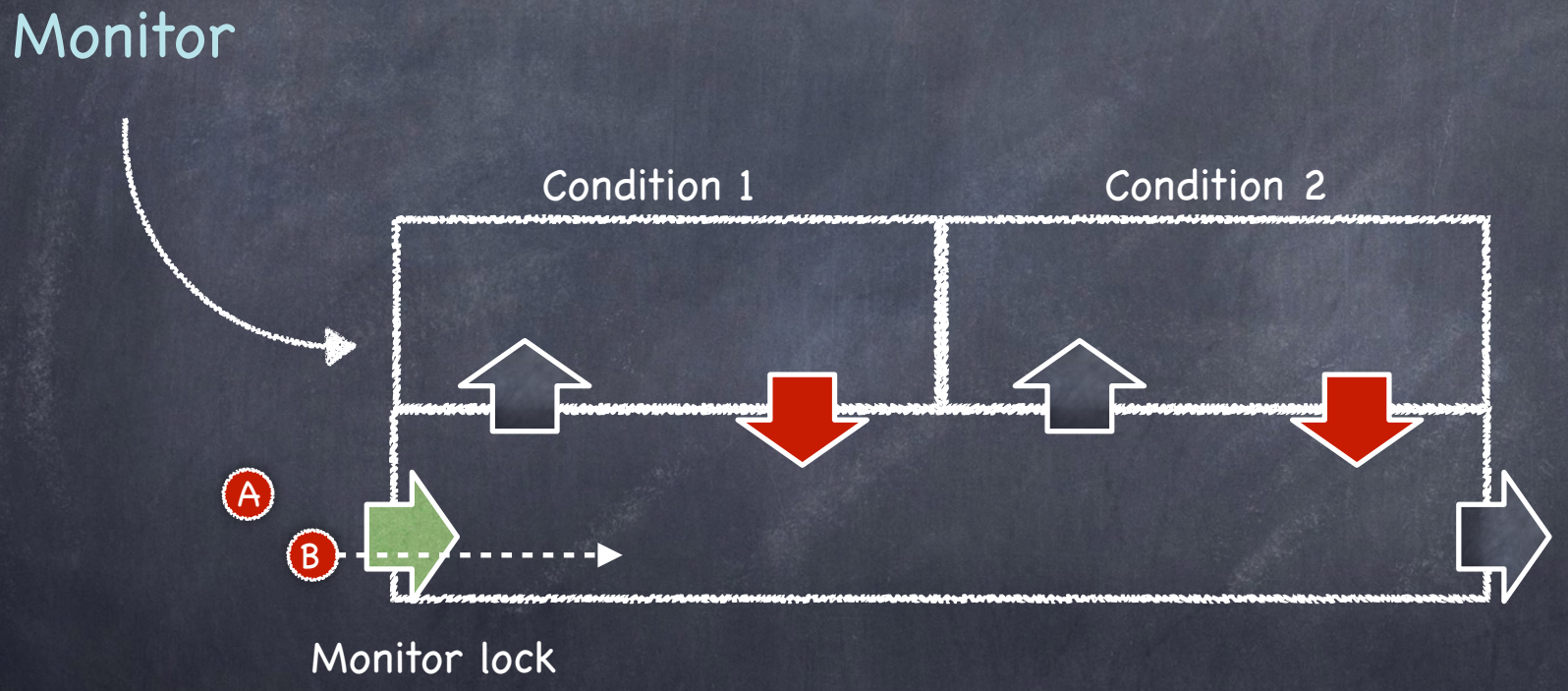
How does Hoare pass the monitor lock ?



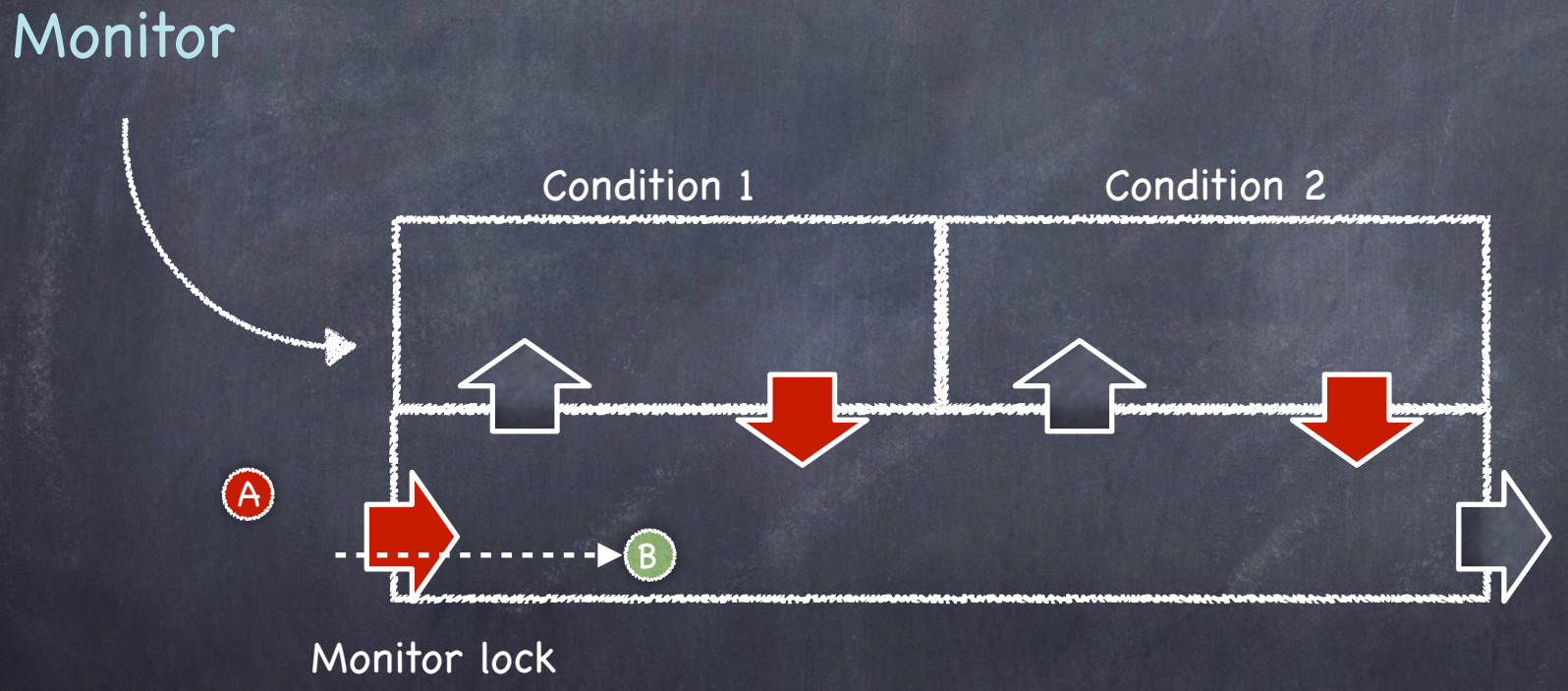
How does Hoare pass the monitor lock ?



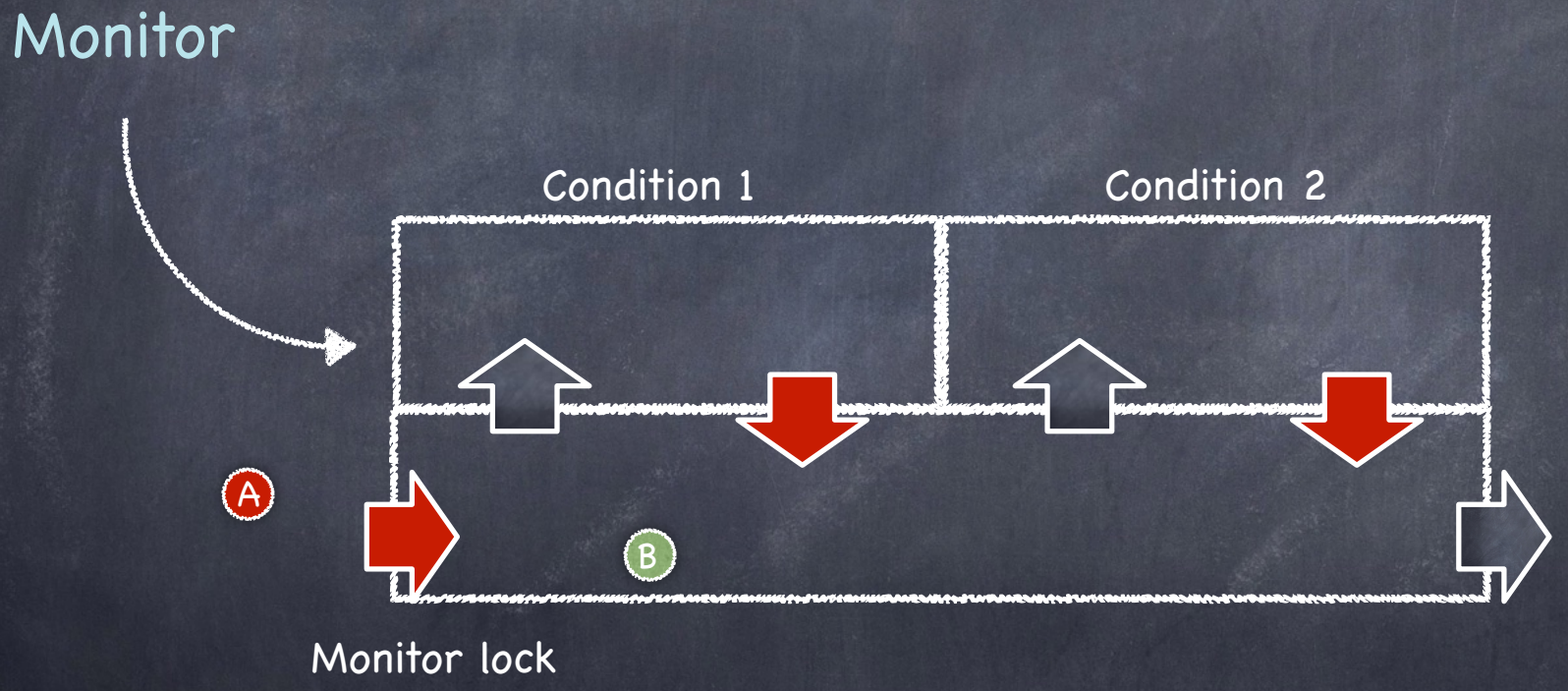
How does Hoare pass the monitor lock ?



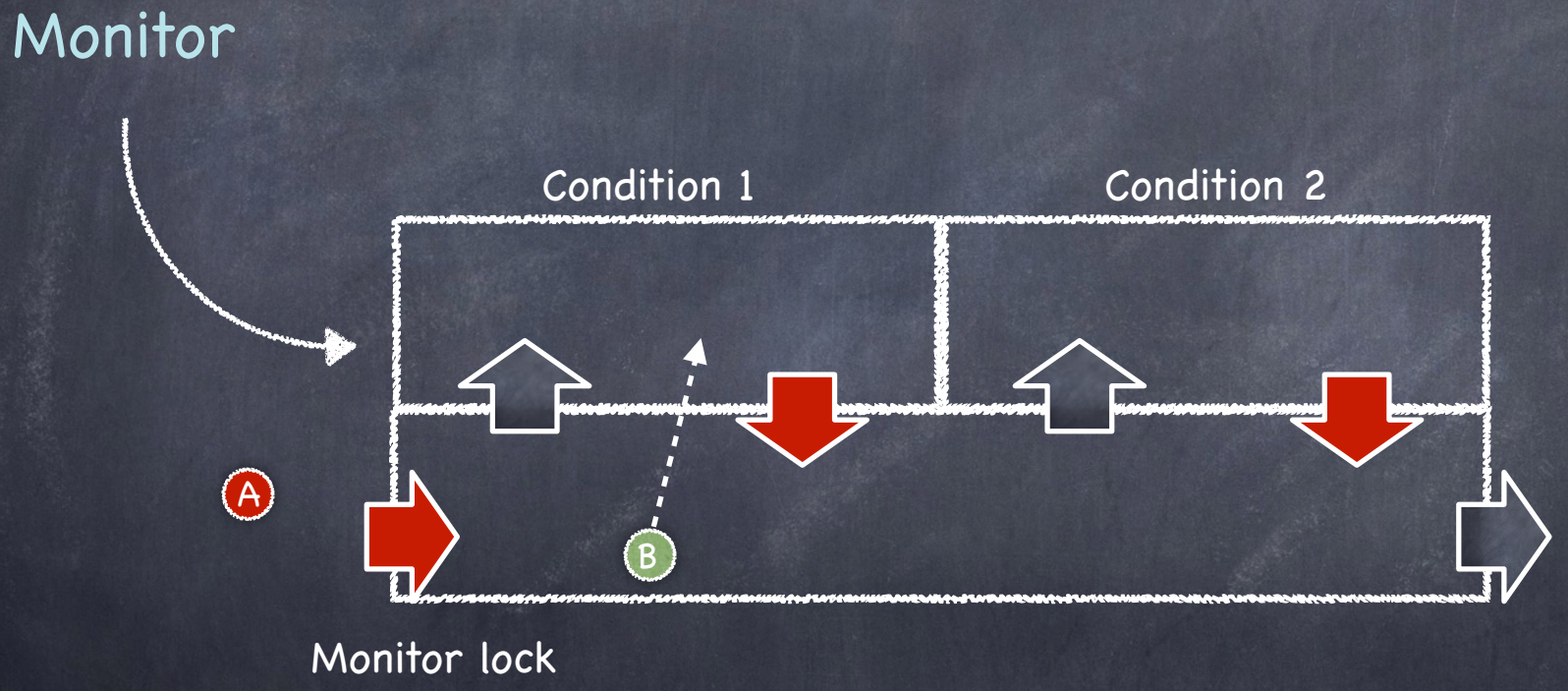
How does Hoare pass the monitor lock ?



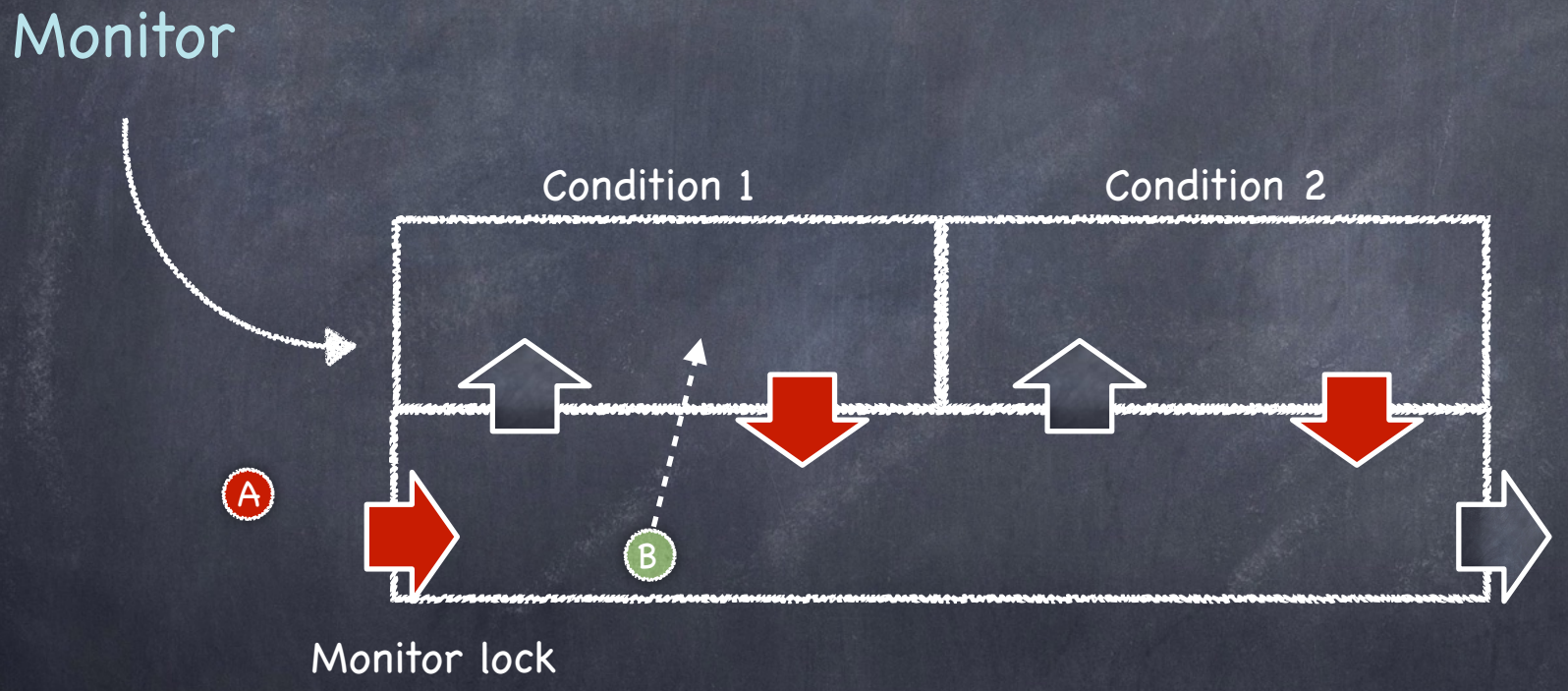
How does Hoare pass the monitor lock ?



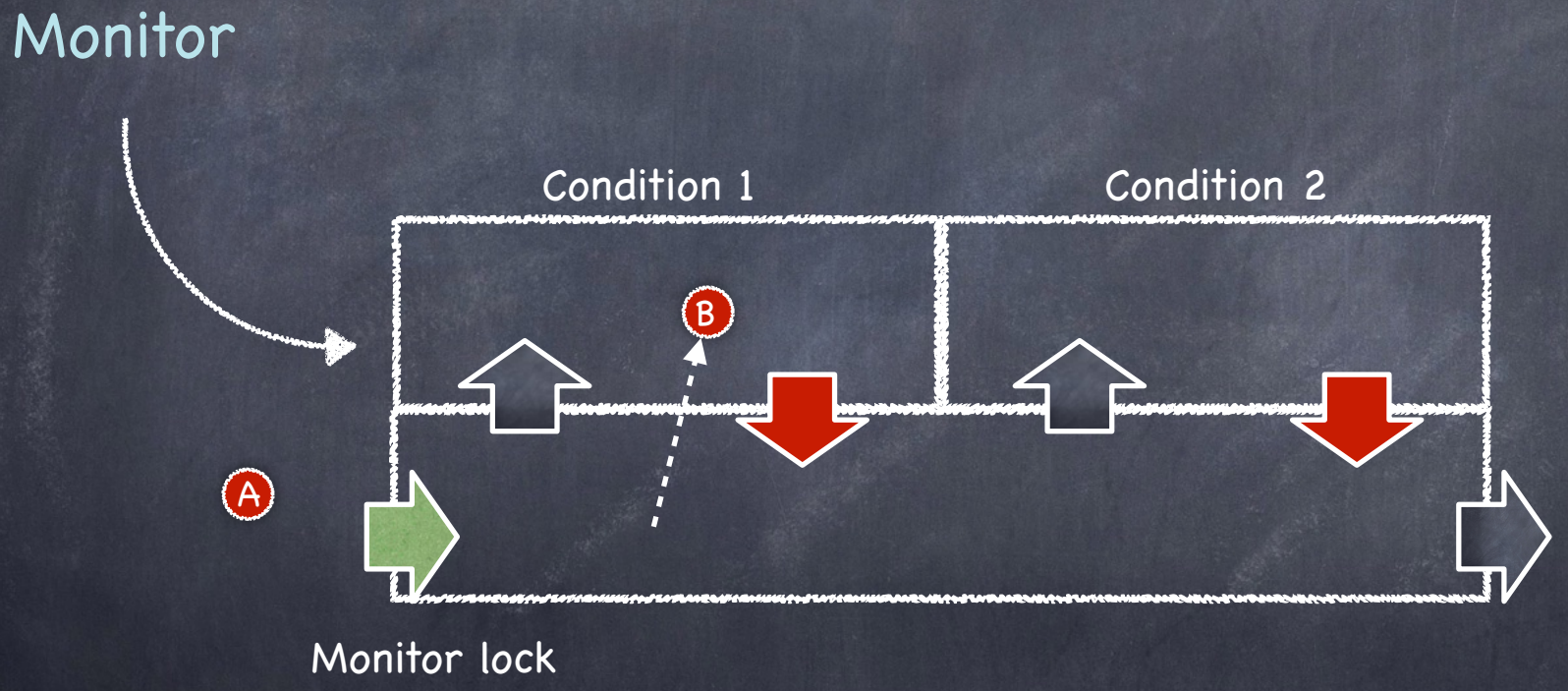
How does Hoare pass the monitor lock ?



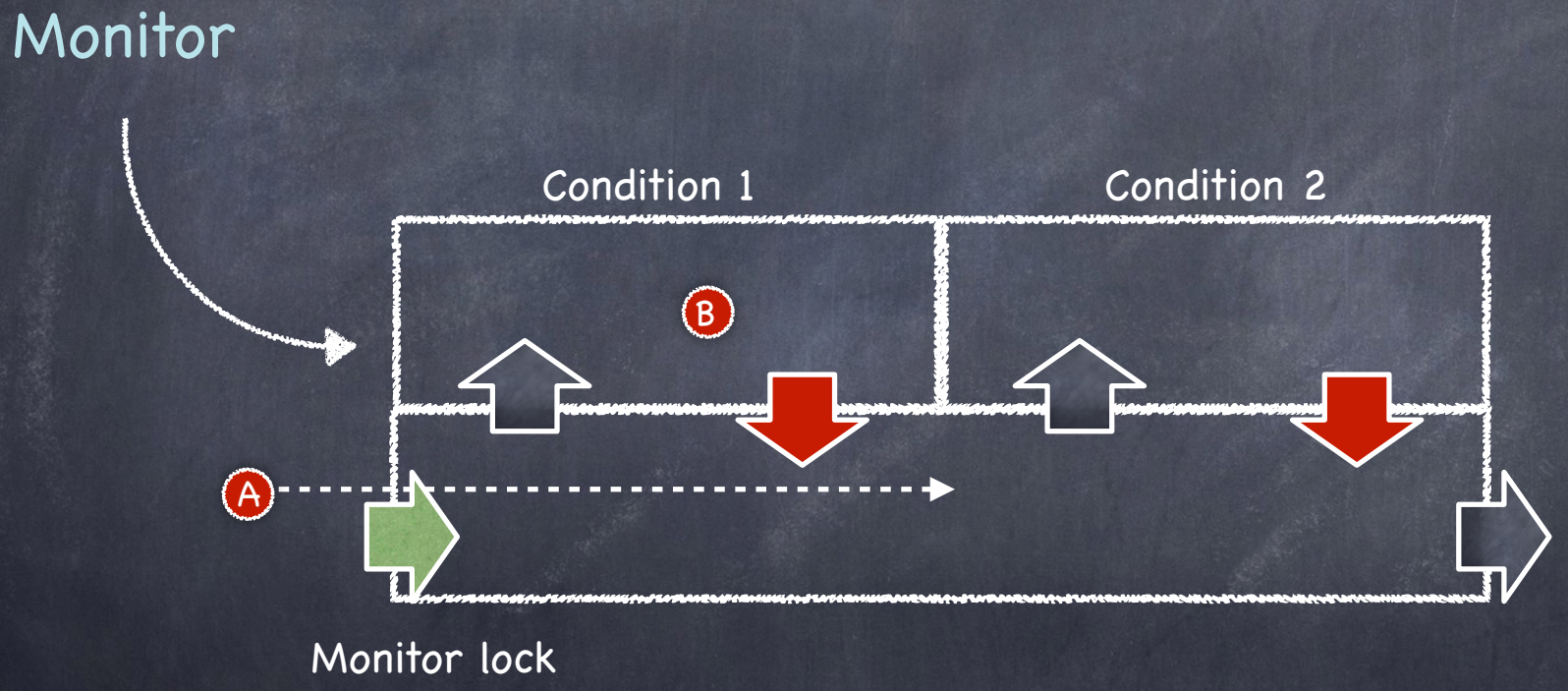
How does Hoare pass the monitor lock ?



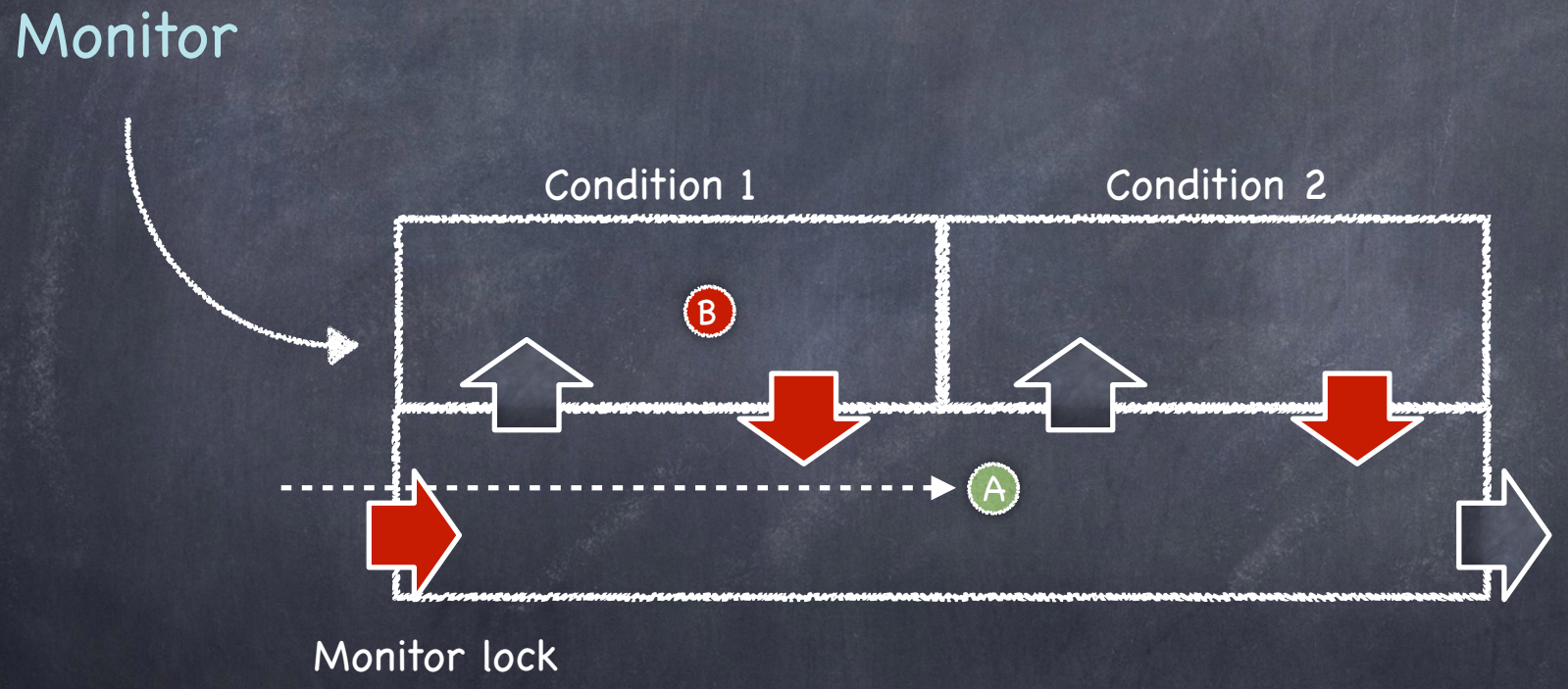
How does Hoare pass the monitor lock ?



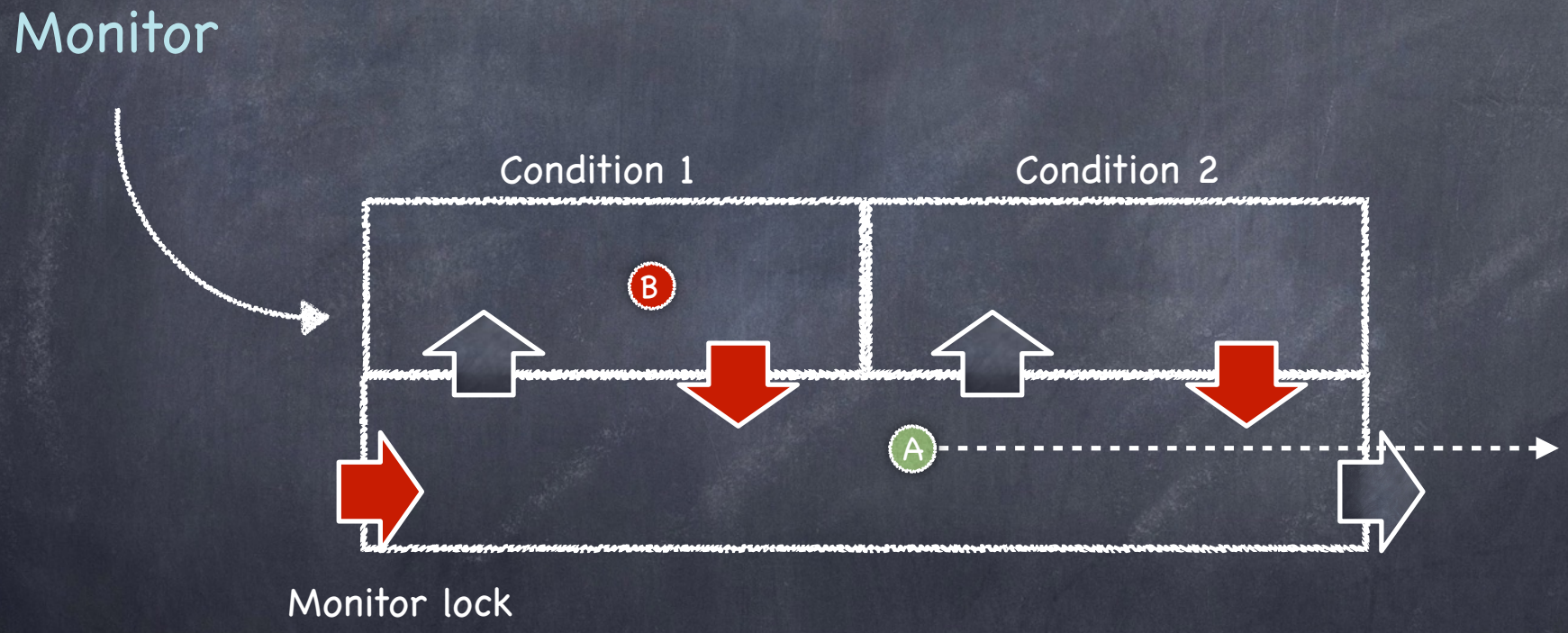
How does Hoare pass the monitor lock ?



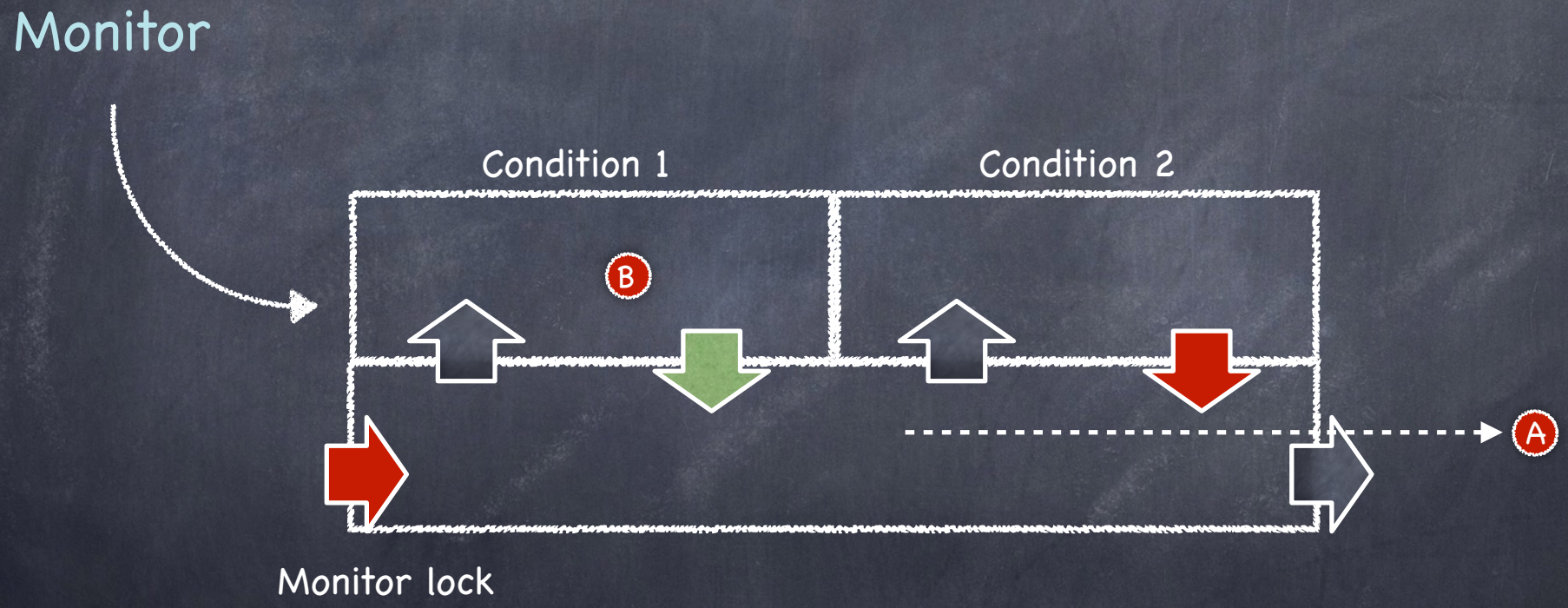
How does Hoare pass the monitor lock ?



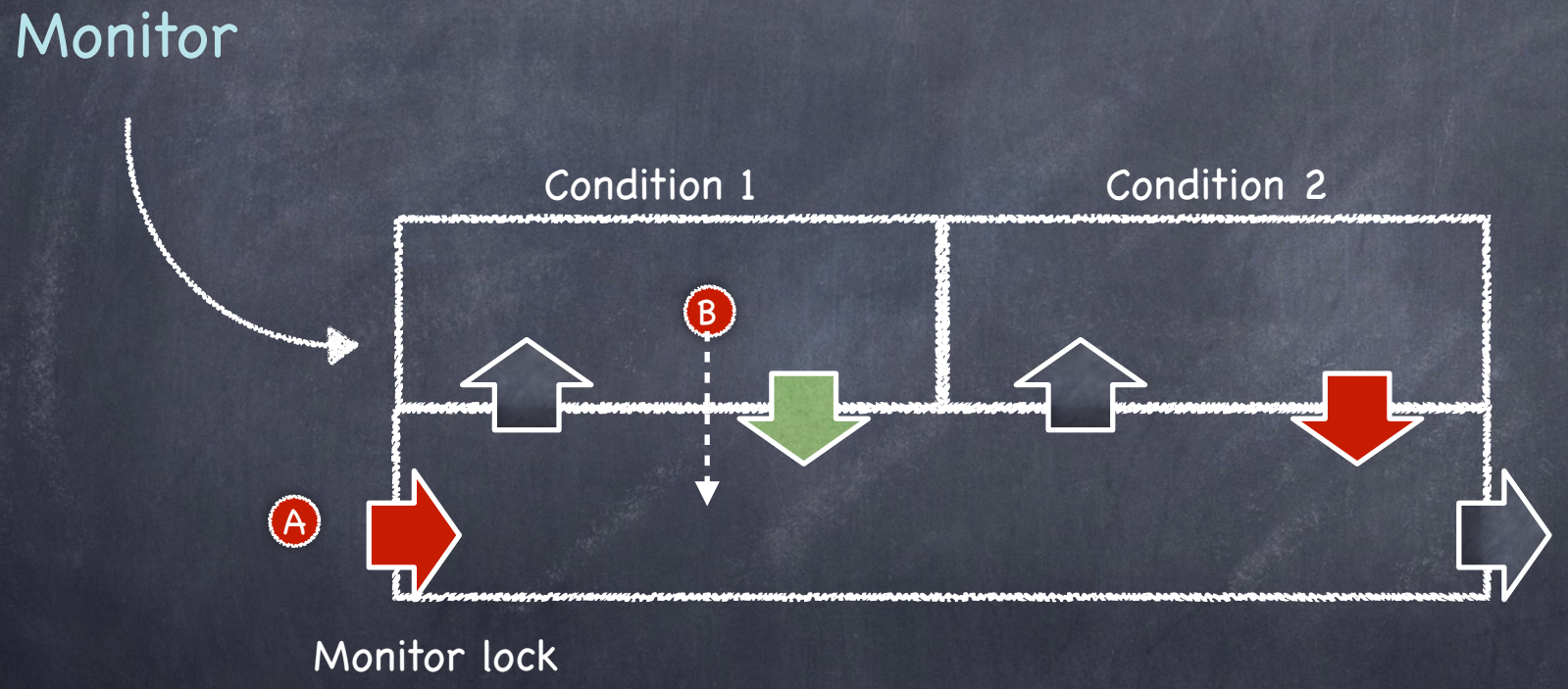
How does Hoare pass the monitor lock ?



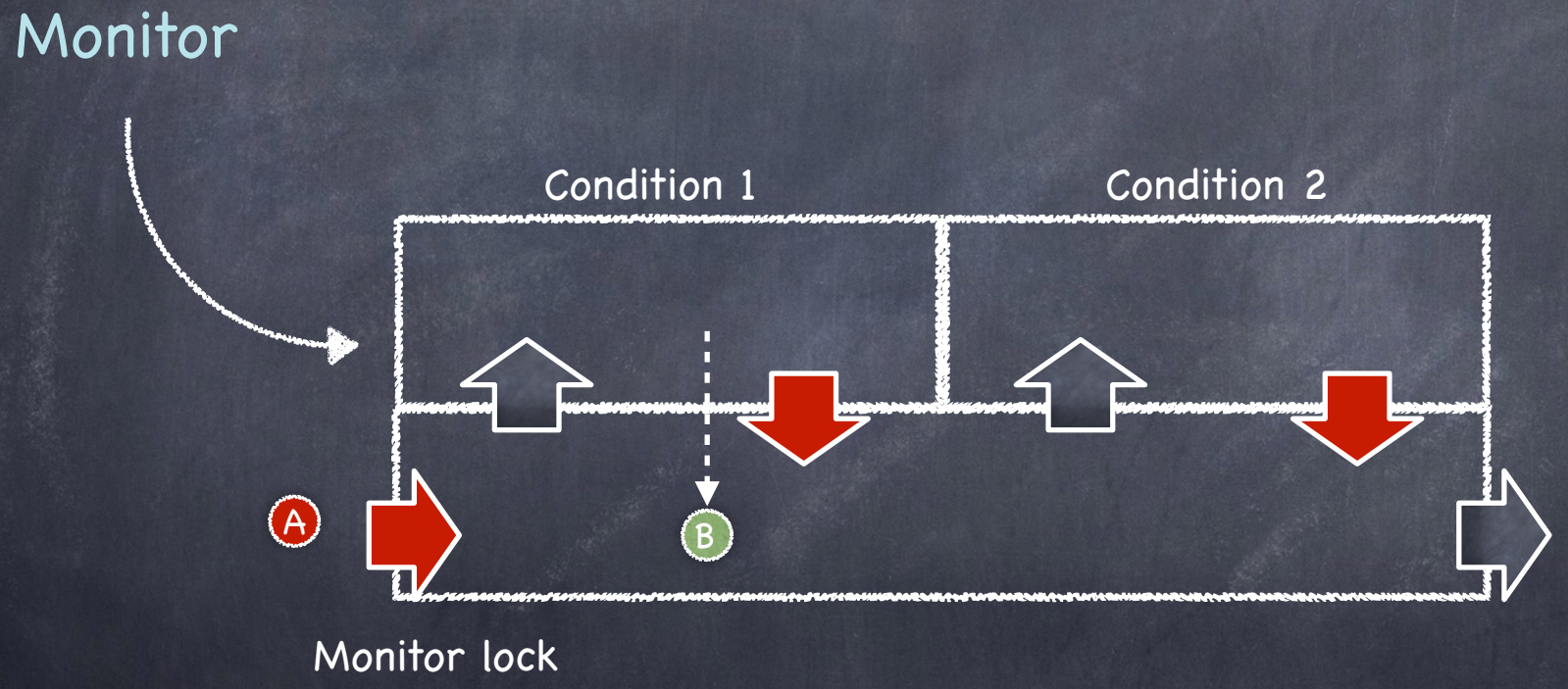
How does Hoare pass the monitor lock ?



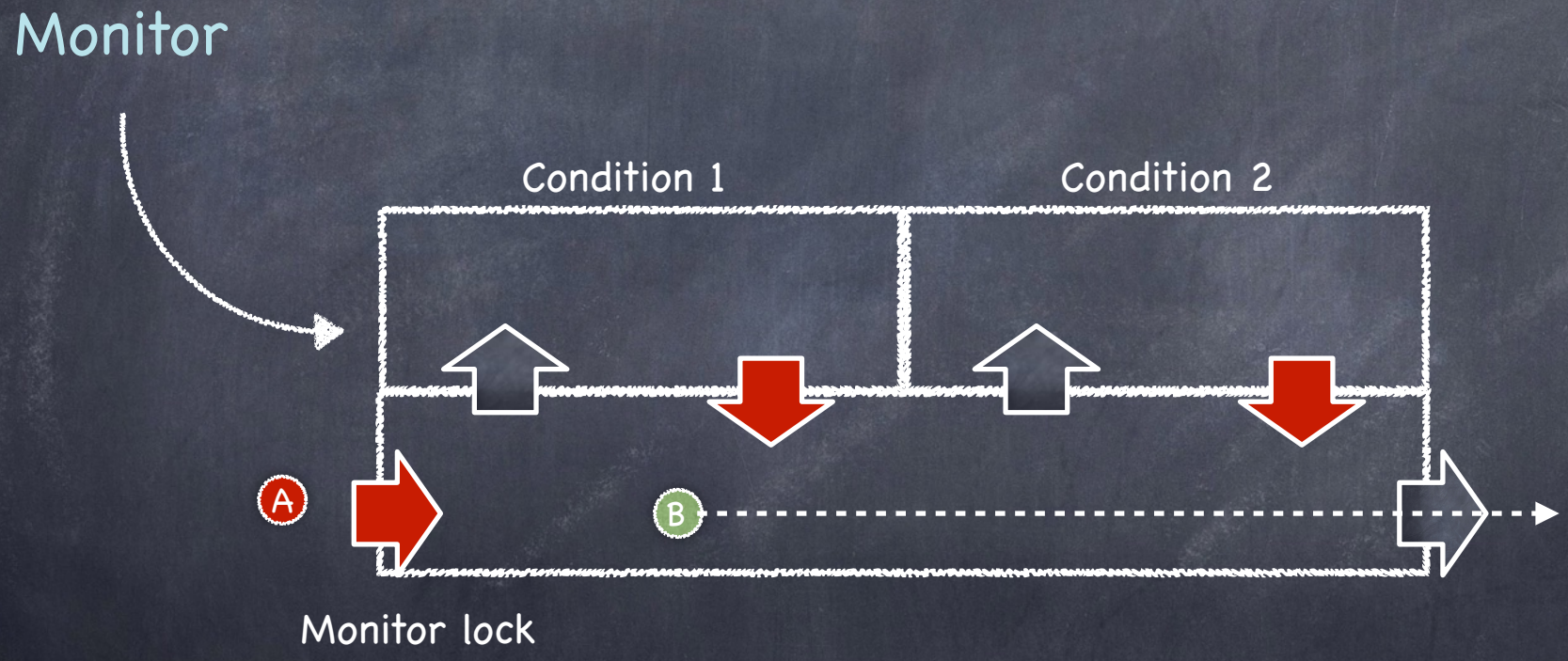
How does Hoare pass the monitor lock ?



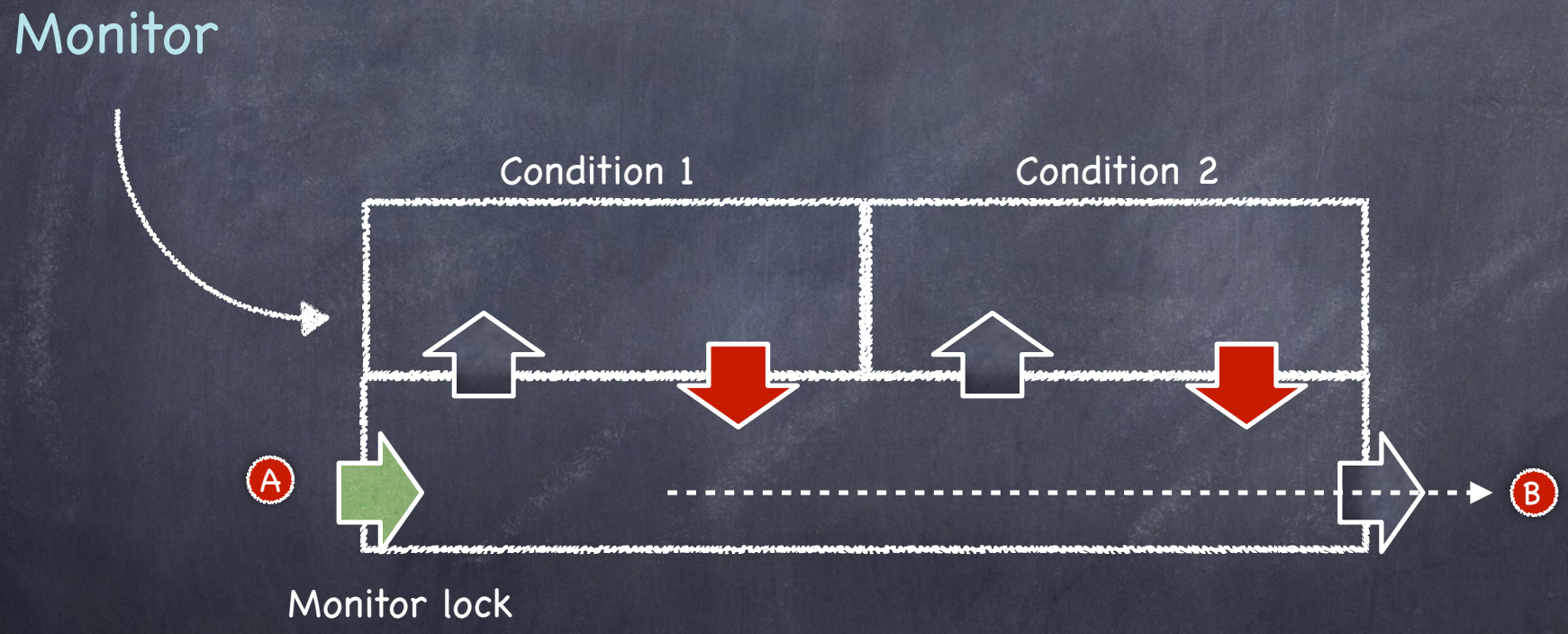
How does Hoare pass the monitor lock ?



How does Hoare pass the monitor lock ?



How does Hoare pass the monitor lock ?



Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Running

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl executes

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

girl back on ready Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

cook gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

cook executes

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

cook back on ready Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

boy gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

boy tries to enter monitor

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

boy tries to enter monitor

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

boy back on ready Q

Running

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
    BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

cook gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
          hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl tries to enter monitor

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
  }
}
```

Q: 0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

monitor has lock Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  lo mlock;
  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Waiting

Q:

0

Running

girl placed on lock Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  lo mlock;
  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Waiting

Q:

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

cook gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  lo mlock;

  int numburgers = 0;
  condition hungrykid;


  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
          hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
  }
}
```

Waiting

Q:

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
     BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

cook executes

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  lock mlock;
  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Waiting

Q: 0

Ready

Running

cook tries to enter monitor

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  lo mlock;
  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Waiting

Q:

0

Ready

Running

cook placed on lock Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  lo mlock;
  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Waiting

Q:

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Running

Ready

boy (with lock) gets the CPU

Kid and Cook Threads

Waiting

Q:

0

```
Monitor BurgerKing {
  lock mlock;
  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

no burgers

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  lock mlock;
  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
     mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
  }
}
```

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Waiting

Q:

0

Waiting

Q:

Ready

Running

boy releases monitor lock & waits for hungrykid signal

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  lo mlock;
  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
          hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
     mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
  }
}
```

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Waiting

Q:

0

Waiting

Q:

Ready

Running

cook joins ready Q with monitor lock that boy released

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
          hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
     mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
  }
}
```

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Waiting

Q:

0

Waiting

Q:

Ready

Running

cook gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  lo mlock;
  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
  }
}
```

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Waiting

Q:

0

Waiting

Q:

Ready

Running

cook acquires monitor lock

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  lock mlock;
  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
          hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
  }
}
```

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Waiting

Q:

Waiting

Q:

1

Ready

Running

cook makes a burger

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  lo  mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
          hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
  }
}
```

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Waiting

Q:

Q:

1

Ready

Running

cook signals a hungry kid

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
}
```

Q:

Q:

1

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

cook releases monitor lock; girl goes back to ready Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
  }
}
```

Q:

Q:

1

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

cook leaves monitor

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
  }
}
```

Q:

Q:

1

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
     BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

cook executes

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
  }
}
```

Q:

Q:

1

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
     BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Running

cook moved to ready Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
      while (numburgers==0)
         hungrykid.wait()
      numburgers -= 1
      mlock.release()
  }

  void makeburger() {
      mlock.acquire()
      ++numburger;
      hungrykid.notify();
      mlock.release()
  }
}
```

Q:

Q:

1

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
     BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Q:

Q:

1

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl acquires monitor lock

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Q:

Q:

1

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
     BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl executes

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Q:

Q:

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
     BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Mmmm… burgers…

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()


  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Q:

0

Q:

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
     BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl releases monitor lock

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Q:

Q:

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
     BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl leaves monitor

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Q:

Q:

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

girl executes

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Q:

Q:

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Running

girl moved to ready Q

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Q: 0

Q:

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

boy gets the CPU

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Q:

Q:

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

boy acquires monitor lock

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Q:

0

Q:

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

boy returns from wait

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Q:

0

Q:

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

boy executes

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Q:

Q:

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
    BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

no burgers

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

Waiting

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
     BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Running

boy releases monitor lock and waits for hungrykid signal

Kid and Cook Threads

```
kid_main() {

    dig_in_mud();
   BK.kid_eat();
    bathe();
    draw_on_walls();
    BK.kid_eat();
   facetime_Karthik();
   facetime_Gma();
   BK.kid_eat();

}
```

Waiting

```
Monitor BurgerKing {
  Lock mlock;

  int numburgers = 0;
  condition hungrykid;

  void kid_eat() {
     mlock.acquire()
     while (numburgers==0)
        hungrykid.wait()
     numburgers -= 1
     mlock.release()
  }

  void makeburger() {
     mlock.acquire()
     ++numburger;
     hungrykid.notify();
     mlock.release()
  }
}
```

Q: 0

```
cook_main() {

  wake();
  shower();
  drive_to_work();
  while(not_5pm)
     BK.makeburger();
  drive_to_home();
  watch_got();
  sleep();

}
```

Ready

Running

cook gets the CPU

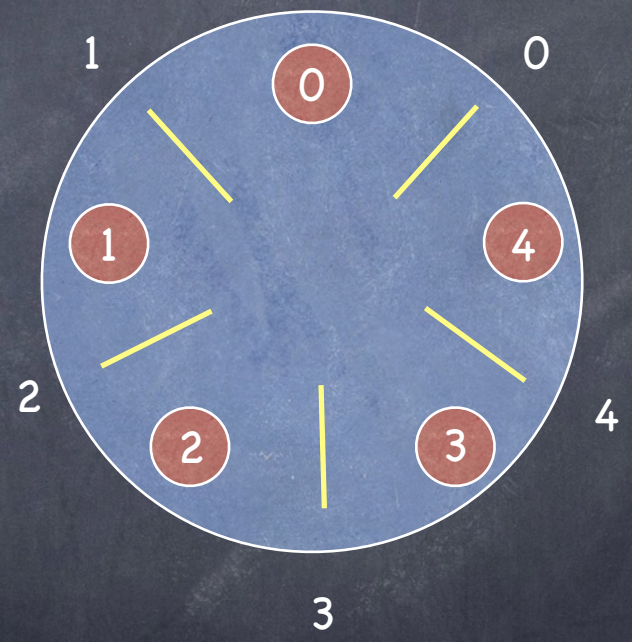
and so forth...

Deadlock

Chapter 32 in "Three Easy Steps"
Chapter 19 in the Harmony Book

Dining Philosophers

```
Pi: do forever
        acquire( left(i) );
        acquire( right(i) );
        eat;
        release( left(i) );
        release( right(i) );
    end
```



left(i): i

right(i): (i+1) mod 5

Dining Philosophers in Harmony

```
1   from synch import Lock, acquire, release
2
3   const N = 5
4
5   forks = [Lock(),] * N
6
7   def diner(which):
8       let left, right = (which, (which + 1) % N):
9           while choose({ False, True }):
10              acquire(?forks[left])
11              acquire(?forks[right])
12              # dine
13              release(?forks[left])
14              release(?forks[right])
15              # think
16
17  for i in {0..N-1}:
18      spawn diner(i)
```

Dining Philosophers in Harmony

```
**Final state** (all threads have terminated or are blocked):

* Threads:
    * T1: (blocked) diner(0) --> acquire(?forks[1])
        * about to execute atomic section in line synch/35
    * T2: (blocked) diner(1) --> acquire(?forks[2])
        * about to execute atomic section in line synch/35
    * T3: (blocked) diner(2) --> acquire(?forks[3])
        * about to execute atomic section in line synch/35
    * T4: (blocked) diner(3) --> acquire(?forks[4])
        * about to execute atomic section in line synch/35
    * T5: (blocked) diner(4) --> acquire(?forks[0])
        * about to execute atomic section in line synch/35
* Variables:
    * forks: [ True, True, True, True, True ]
```

Problematic Emergent Properties

- Starvation: Process waits forever
- Deadlock: a set of processes exist, where each is blocked and can become unblocked only by the action of another process in the same set
  - Deadlock implies Starvation (but not viceversa)
  - Starvation often tied to fairness — which requires that a process be not forever blocked on a condition that becomes (i) continuously true or (ii) infinitely-often true

Testing for starvation or deadlock is difficult in practice

More Examples of Deadlock

- Example 1 (initially in1 = in2 = False):

```
in1 = True;  await not in2;  in1 = False
                    //
in2 = True;  await not in1;  in2 = False
```

- Example 2 (initially lk1 = lk2 = released):

```
acquire(lk1); acquire(lk2);  release(lk2); release(lk1)
                    //
acquire(lk2); acquire(lk1);  release(lk1); release(lk2)
```

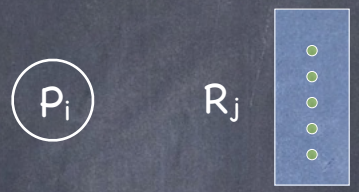
System Model

- Set of resources requiring "exclusive" access
  - Might be "k exclusive access" if k instances of resource are available
  - Examples: buffers, packets, I/O devices, processors
- Protocol to access a resource causes blocking
  - If resource is free, access is granted and process proceeds
    - Uses resource
    - Releases resource
  - If resource is in use, process blocks

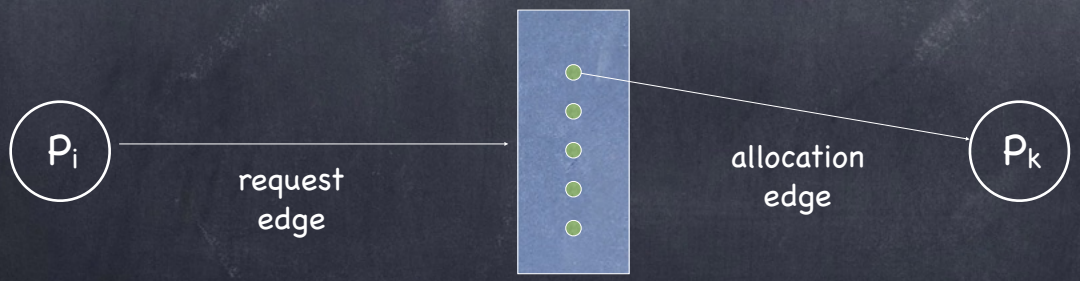
A Graph Theoretic Model of Deadlock

Resource Allocation Graph

- Computer system modeled as a RAG, a directed graph G(V, E)
  - $V = \{P_1,\ldots,P_n\} \cup \{R_1,\ldots,R_n\}$



  - E = {edges from a resource to a process} ∪ {edges from a process to a resource}



Necessary conditions for deadlock

Deadlock only if they all hold

1. **Bounded resources**

   Acquire can block invoker

2. **No preemption**

   the resource is mine, MINE! (until I release it)

3. **Wait while holding**

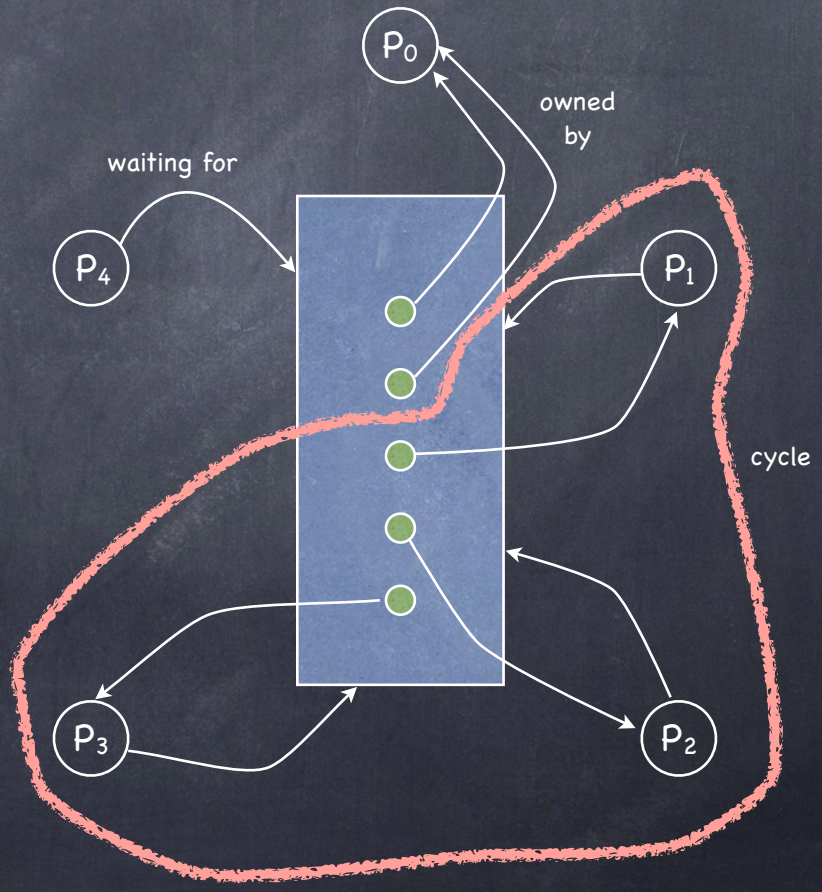
   holds one resource while waiting for another

4. **Circular waiting**

   P_i waits for P_{i+1} and holds a resource requested by P_{i-1}

   sufficient if one instance of each resource

Not sufficient in general



Deadlock is Undesirable!

- Deadlock prevention: Ensure that a necessary condition cannot hold
- Deadlock avoidance: System does not allocate resources that may lead to a deadlock
- Deadlock detection: Allow system to deadlock; detect it; recover

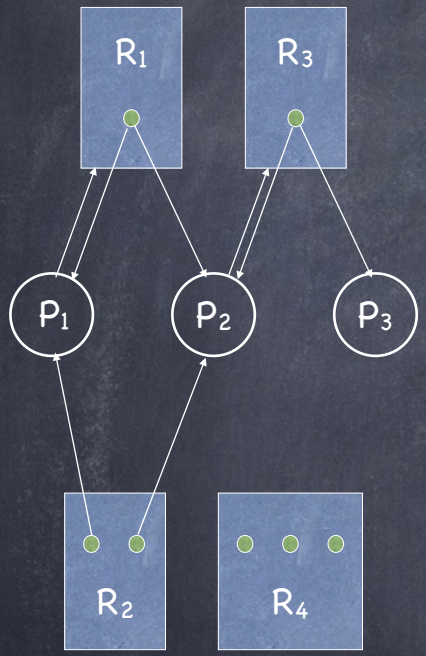
Testing for cycles

- Reduction Algorithm
  - Find a node with no outgoing edges
    - Erase any edges coming into it
    - Repeat until no such node
- Intuition: Node with no outgoing edges is not waiting on any resource
  - It will eventually finish and release its resources
  - Processes waiting for those resources will be able to acquire them and will no longer be waiting!

Erase all edges $\iff$ Graph has no cycles

Edges remain $\iff$ Deadlock

RAG Reduction



Deadlock?

NO! (no cycles)

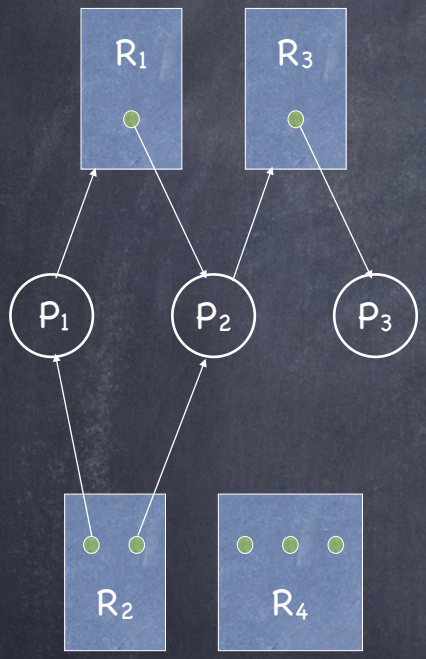
Step 1: Satisfy P_3's requests

Step 2: Satisfy P_2's requests

Step 3: Satisfy P_1's requests

Schedule [P_3 P_2 P_1] completely eliminates edges!

RAG Reduction



Deadlock?

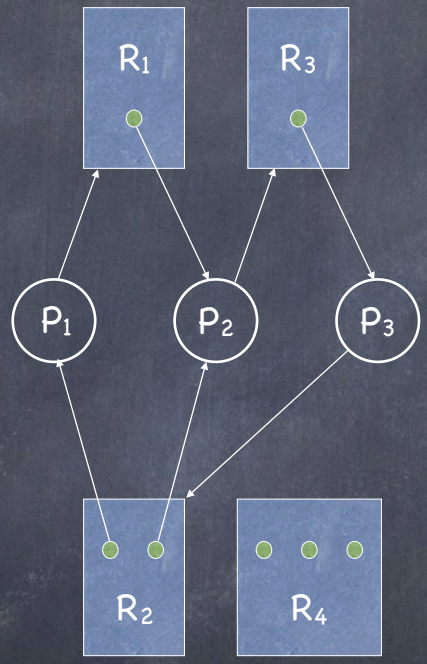
NO! (no cycles)

Step 1: Satisfy P_3's requests

Step 2: Satisfy P_2's requests

Step 3: Satisfy P_1's requests

Schedule [P_3 P_2 P_1] completely eliminates edges!



Deadlock?

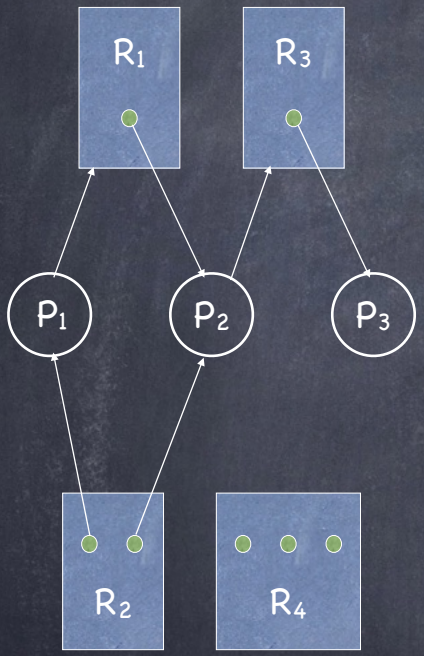
Yes!

RAG has a cycle

Every node has some outgoing edge

Cannot satisfy any of P_1, P_2, P_3 requests!

RAG Reduction



Deadlock?

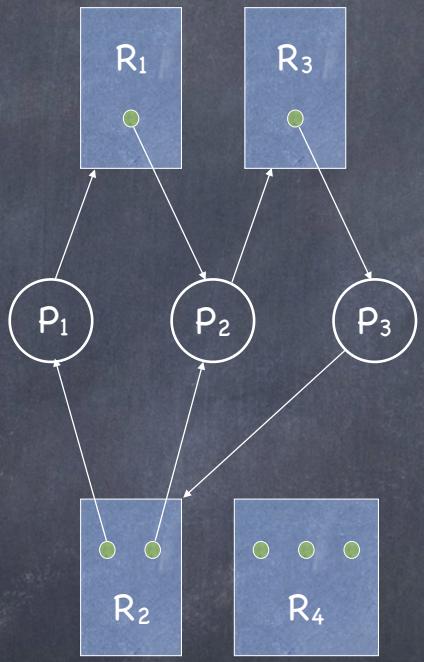
NO! (no cycles)

Step 1: Satisfy P_3's requests

Step 2: Satisfy P_2's requests

Step 3: Satisfy P_1's requests

Schedule [P_3 P_2 P_1] completely eliminates edges!



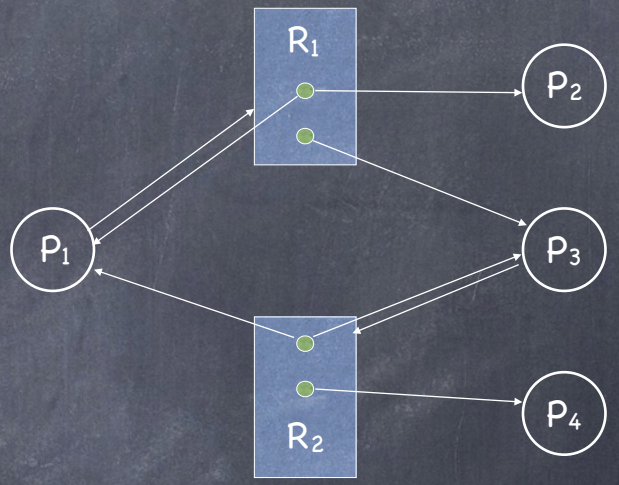
Deadlock?

Yes!

RAG has a cycle

Every node has some outgoing edge

Cannot satisfy any of P_1, P_2, P_3 requests!



Deadlock?

NO!

RAG has a cycle

Schedule [P_2 P_1 P_3 P_4] completely eliminates edges!

More Musings on Deadlock

- Does the order of RAG reduction matter?
  - No. If P_i and P_j can both be reduced, reducing P_i does not affect the reducibility of P_j
- Does a deadlock disappear on its own?
  - No. Unless a process is killed or forced to release a resource, we are stuck!
- If a system is not deadlocked at time T, is it guaranteed to be deadlock-free at T+1?
  - No. Just by requesting a resource (never mind being granted one) a process can create a circular wait!