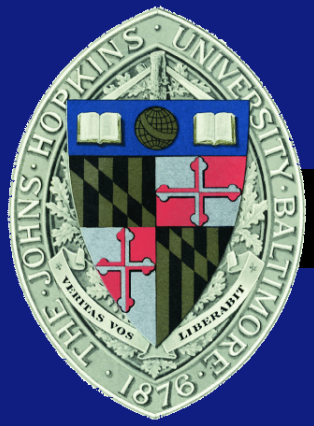
Lecture 12.1
Introduction to Synchronization

EN 600.320/420

Instructor: Randal Burns

13 March 2017



Department of Computer Science, *Johns Hopkins University*

Synchronization

A look inside the critical section

Two common goals for synchronization

- Contention:
  - How to resolve the conflicts that result from multiple processes trying to access shared resources?
- Cooperation:
  - An action by one process may enable another action by another process
  - In such cases, processes should coordinate their actions

Why is synchronization hard?

- Design an algorithm for purchasing milk between two roomates Alice and Bob
- Steps:
  - Arrive home
  - Look in fridge for milk
  - Leave for grocery
  - Buy milk
  - Arrive home with purchased milk

Alice

- Arrive home
- Look in fridge for milk
- Leave for grocery
- Buy milk
- Arrive home with purchased milk

Bob

- Arrive home
- Look in fridge for milk
- Leave for grocery
- Buy milk
- Arrive home with purchased milk

Why is synchronization hard?

- Design an algorithm for purchasing milk between two roomates Alice and Bob
- Steps:
  - Arrive home
  - Look in fridge for milk
  - Leave for grocery
  - Buy milk
  - Arrive home with purchased milk
- Too much milk!
- Problem is impossible without communication between parties

Let's Try Using Notes

- *Algorithm #1:* If you find that there is no milk in fridge, leave a note on the door, go to store and purchase milk, on return home remove note

```
if (no note) then
  if (no milk) then
    leave note
    buy milk
    remove note
  fi
fi
```

They can't see each other

Alice

```
if (no note) then
  if (no milk) then
    leave note
    buy milk
    remove note
  fi
fi
```

Bob

```
if (no note) then
  if (no milk) then
    leave note
    buy milk
    remove note
  fi
fi
```

Let's Try Using Notes

- *Algorithm #2:* Based on leaving a note (with one's name) before checking fridge

```
leave note A
if (no note B) then
  if (no milk) then
    buy milk
  fi
fi
remove note
```

Alice

```
leave note A
if (no note B) then
  if (no milk) then
    buy milk
  fi
fi
remove note
```

Bob

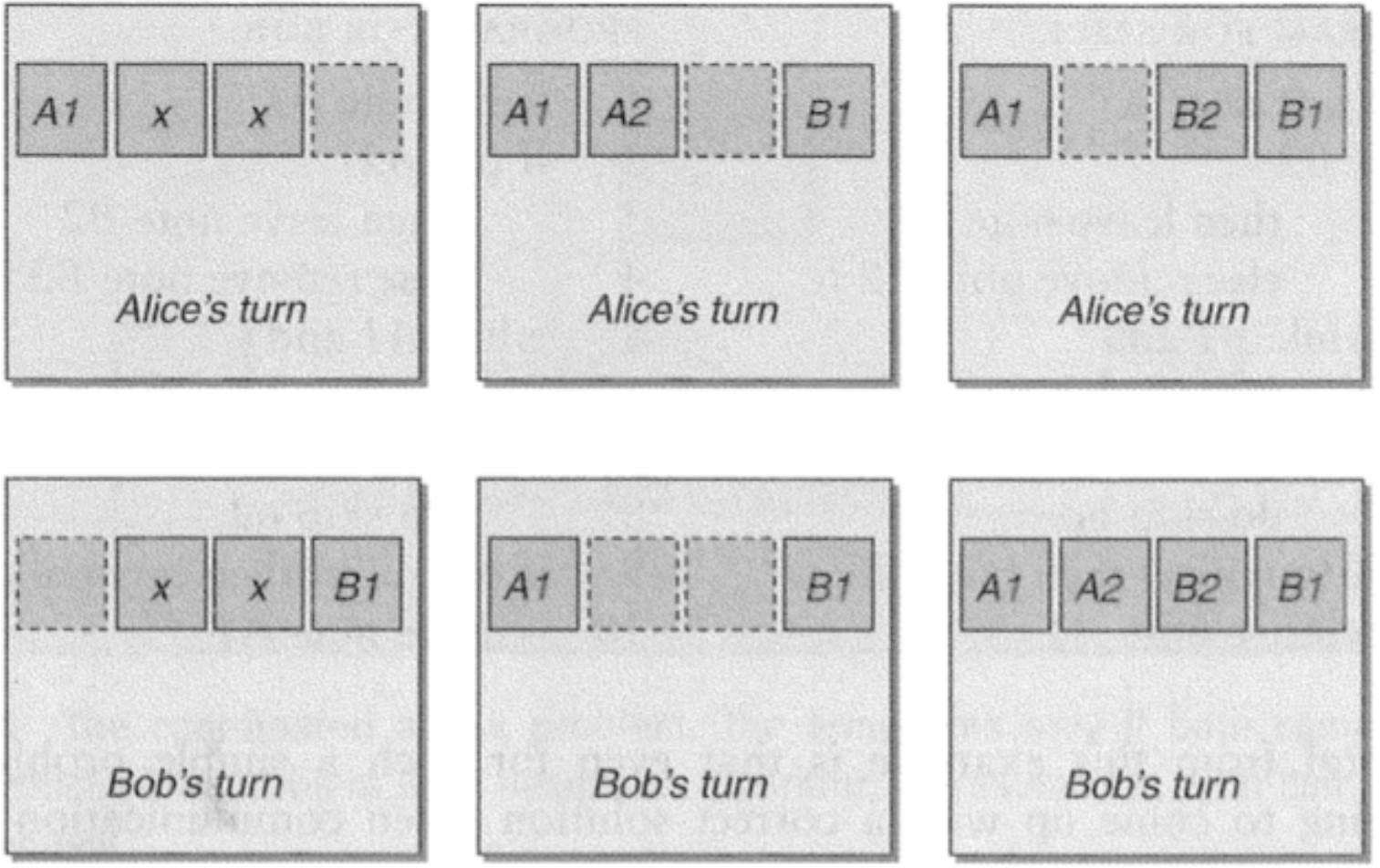
```
leave note B
if (no note A) then
  if (no milk) then
    buy milk
  fi
fi
remove note
```

A Correct Algorithm

```
leave note A1
if (B2)
  then leave note A2
  else remove note A2 fi
while B1 and
  ((A2 and B2) or
   (no A2 and no B2))
  do skip do
if (no milk)
  then buy milk fi
remove note A1
```

```
leave note B1
if (no A2)
  then leave note B2
  else remove note B2 fi
while A1 and
  ((A2 and no B2) or
   (no A2 and B2))
  do skip do
if (no milk)
  then buy milk fi
remove note B1
```

Possible Configurations



Two Notes

- First one to identify contention
  - Are two parties vying for this resource
- Second one to break ties during contention
  - Essentially even and odd configurations

- These notes are the analogies of atomic shared registers in computing
  - Essentially a volatile variable of basic type

Some Properties

- Correct
- Asynchronous: doesn't depend on timing
- Symmetric: equal chance of A/B buying milk
  - Notably steps aren't symmetric
- Two parties

- Even simple synchronization is hard and subtle

The Point Again

- Going to take a somewhat more formal look at synchronization
  - Not just present the constructs
- Synchronization issues are the major bug in parallel programs
  - Deadlock
  - Incorrect results
- The constructs/algorithms underlying critical sections, locks, atomic variables are complex
  - Understanding them will help you use them well