Mutual Exclusion and Fairness

October 2026

Setup

Two processes P_0 and P_1 run concurrently. Each repeatedly executes:

Setup

Two processes P_0 and P_1 run concurrently. Each repeatedly executes:

1. a non-critical section

Setup

Two processes P_0 and P_1 run concurrently. Each repeatedly executes:

1. a non-critical section
2. an entry protocol

Setup

Two processes P_0 and P_1 run concurrently. Each repeatedly executes:

1. a non-critical section
2. an entry protocol
3. a critical section

Setup

Two processes P_0 and P_1 run concurrently. Each repeatedly executes:

1. a non-critical section
2. an entry protocol
3. a critical section
4. an exit protocol

Setup

Two processes P_0 and P_1 run concurrently. Each repeatedly executes:

1. a non-critical section
2. an entry protocol
3. a critical section
4. an exit protocol

A run interleaves each process's steps in some arbitrary order.

Key Properties

Call cs_i and try_i the propositions signalling that P_i is in its critical section or in its entry protocol, respectively.

Key Properties

Call cs_i and try_i the propositions signalling that P_i is in its critical section or in its entry protocol, respectively.

- Mutual Exclusion: the two processes are never both in their critical section.

Key Properties

Call cs_i and try_i the propositions signalling that P_i is in its critical section or in its entry protocol, respectively.

- Mutual Exclusion: the two processes are never both in their critical section.

$$\Box\neg(cs_0 \wedge cs_1)$$

Key Properties

Call cs_i and try_i the propositions signalling that P_i is in its critical section or in its entry protocol, respectively.

- Mutual Exclusion: the two processes are never both in their critical section.

$$\Box\neg(cs_0 \wedge cs_1)$$

- Deadlock freedom: if a process tries to enter its critical section, then a process will eventually enter its critical section (or is there already).

Key Properties

Call cs_i and try_i the propositions signalling that P_i is in its critical section or in its entry protocol, respectively.

- Mutual Exclusion: the two processes are never both in their critical section.

$$\Box\neg(cs_0 \wedge cs_1)$$

- Deadlock freedom: if a process tries to enter its critical section, then a process will eventually enter its critical section (or is there already).

$$\Box((try_0 \vee try_1) \rightarrow \Diamond(cs_0 \vee cs_1))$$

Key Properties

Call cs_i and try_i the propositions signalling that P_i is in its critical section or in its entry protocol, respectively.

- Mutual Exclusion: the two processes are never both in their critical section.

$$\Box\neg(cs_0 \wedge cs_1)$$

- Deadlock freedom: if a process tries to enter its critical section, then a process will eventually enter its critical section (or is there already).

$$\Box((try_0 \vee try_1) \rightarrow \Diamond(cs_0 \vee cs_1))$$

- Starvation freedom: if a process tries to enter its critical section, it will eventually enter its critical section.

Key Properties

Call cs_i and try_i the propositions signalling that P_i is in its critical section or in its entry protocol, respectively.

- Mutual Exclusion: the two processes are never both in their critical section.

$$\Box\neg(cs_0 \wedge cs_1)$$

- Deadlock freedom: if a process tries to enter its critical section, then a process will eventually enter its critical section (or is there already).

$$\Box((try_0 \vee try_1) \rightarrow \Diamond(cs_0 \vee cs_1))$$

- Starvation freedom: if a process tries to enter its critical section, it will eventually enter its critical section.

$$\Box(try_i \rightarrow \Diamond cs_i)$$

Fairness

Technically, a scheduler could simply choose to never run P_0 at all!

Fairness

Technically, a scheduler could simply choose to never run P_0 at all!

Call ub_i the proposition "P_i is unblocked,"
and ex_i the proposition "P_i takes the current step."

How might you read these properties?

1. $\Diamond\Box ub_i \rightarrow \Box\Diamond ex_i$

Fairness

Technically, a scheduler could simply choose to never run P_0 at all!

Call ub_i the proposition "P_i is unblocked,"
and ex_i the proposition "P_i takes the current step."

How might you read these properties?

1. $\Diamond\Box ub_i \rightarrow \Box\Diamond ex_i$
2. $\Box\Diamond ub_i \rightarrow \Box\Diamond ex_i$

Fairness

Technically, a scheduler could simply choose to never run P_0 at all!

Call ub_i the proposition "P_i is unblocked,"
and ex_i the proposition "P_i takes the current step."

How might you read these properties?

1. $\Diamond\Box ub_i \rightarrow \Box\Diamond ex_i$
2. $\Box\Diamond ub_i \rightarrow \Box\Diamond ex_i$

The first property says that a process that is *eventually constantly* unblocked will always eventually be scheduled.

Fairness

Technically, a scheduler could simply choose to never run P_0 at all!

Call ub_i the proposition "P_i is unblocked,"
and ex_i the proposition "P_i takes the current step."

How might you read these properties?

1. $\Diamond\Box ub_i \rightarrow \Box\Diamond ex_i$
2. $\Box\Diamond ub_i \rightarrow \Box\Diamond ex_i$

The first property says that a process that is *eventually constantly* unblocked will always eventually be scheduled.

The second property says that a process that is *always eventually* unblocked will always eventually be scheduled.

Fairness Cont'd

1. $\Diamond\Box ub_i \rightarrow \Box\Diamond ex_i$
2. $\Box\Diamond ub_i \rightarrow \Box\Diamond ex_i$

How are the first and second propositions related?

Fairness Cont'd

1. $\Diamond\Box ub_i \rightarrow \Box\Diamond ex_i$
2. $\Box\Diamond ub_i \rightarrow \Box\Diamond ex_i$

How are the first and second propositions related?

We call them "weak fairness," and "strong fairness" respectively.

Fairness Cont'd

1. $\Diamond\Box ub_i \rightarrow \Box\Diamond ex_i$
2. $\Box\Diamond ub_i \rightarrow \Box\Diamond ex_i$

How are the first and second propositions related?

We call them "weak fairness," and "strong fairness" respectively. Note that strong fairness implies weak fairness.

Fairness Cont'd

1. $\Diamond\Box ub_i \rightarrow \Box\Diamond ex_i$
2. $\Box\Diamond ub_i \rightarrow \Box\Diamond ex_i$

How are the first and second propositions related?

We call them "weak fairness," and "strong fairness" respectively. Note that strong fairness implies weak fairness.

We say a property φ holds under fairness if:

$$(Fair_0 \wedge Fair_1) \rightarrow \varphi$$

Fairness Cont'd

1. $\Diamond\Box ub_i \rightarrow \Box\Diamond ex_i$
2. $\Box\Diamond ub_i \rightarrow \Box\Diamond ex_i$

How are the first and second propositions related?

We call them "weak fairness," and "strong fairness" respectively. Note that strong fairness implies weak fairness.

We say a property φ holds under fairness if:

$$(Fair_0 \wedge Fair_1) \rightarrow \varphi$$

Check-and-set

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      await ¬flag[j]
4      flag[i] := true
5      # critical section
6      flag[i] := false
```

- Mutual exclusion?

Check-and-set

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      await ¬flag[j]
4      flag[i] := true
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **No!**

Check-and-set

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      await ¬flag[j]
4      flag[i] := true
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **No!**
- Deadlock freedom?

Check-and-set

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      await ¬flag[j]
4      flag[i] := true
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **No!**
- Deadlock freedom? **Yes!**

Check-and-set

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      await ¬flag[j]
4      flag[i] := true
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **No!**
- Deadlock freedom? **Yes!**
- Starvation freedom...

Check-and-set

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      await ¬flag[j]
4      flag[i] := true
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **No!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness?

Check-and-set

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      await ¬flag[j]
4      flag[i] := true
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **No!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**

Check-and-set

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      await ¬flag[j]
4      flag[i] := true
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **No!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness?

Check-and-set

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      await ¬flag[j]
4      flag[i] := true
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **No!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**

Check-and-set

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      await ¬flag[j]
4      flag[i] := true
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **No!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**
  - Under strong fairness?

Check-and-set

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      await ¬flag[j]
4      flag[i] := true
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **No!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**
  - Under strong fairness? **Yes!**

Check-and-set

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      await ¬flag[j]
4      flag[i] := true
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **No!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**
  - Under strong fairness? **Yes!** (not that it matters)

Set-then-check

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      await ¬flag[j]
5      # critical section
6      flag[i] := false
```

- Mutual exclusion?

Set-then-check

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      await ¬flag[j]
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **Yes!**

Set-then-check

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      await ¬flag[j]
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom?

Set-then-check

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      await ¬flag[j]
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**

Set-then-check

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      await ¬flag[j]
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...

Set-then-check

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      await ¬flag[j]
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...
  - Without fairness?

Set-then-check

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      await ¬flag[j]
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...
  - Without fairness? **No!**

Set-then-check

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      await ¬flag[j]
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness?

Set-then-check

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      await ¬flag[j]
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**

Set-then-check

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      await ¬flag[j]
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**
  - Under strong fairness?

Set-then-check

Sharing array initialized by $flag[0..1] = \bot$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      await ¬flag[j]
5      # critical section
6      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**
  - Under strong fairness? **No!**

Strict alternation

Sharing variable initialized by $turn = 0$

```
1  loop
2      # non-critical section (arbitrary code!)
3      await turn = i
4      # critical section
5      turn := j
```

- Mutual exclusion?

Strict alternation

Sharing variable initialized by $turn = 0$

```
1  loop
2      # non-critical section (arbitrary code!)
3      await turn = i
4      # critical section
5      turn := j
```

- Mutual exclusion? **Yes!**

Strict alternation

Sharing variable initialized by $turn = 0$

```
1  loop
2      # non-critical section (arbitrary code!)
3      await turn = i
4      # critical section
5      turn := j
```

- Mutual exclusion? **Yes!**
- Deadlock freedom?

Strict alternation

Sharing variable initialized by $turn = 0$

```
1  loop
2      # non-critical section (arbitrary code!)
3      await turn = i
4      # critical section
5      turn := j
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**

Strict alternation

Sharing variable initialized by $turn = 0$

```
1  loop
2      # non-critical section (arbitrary code!)
3      await turn = i
4      # critical section
5      turn := j
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...

Strict alternation

Sharing variable initialized by $turn = 0$

```
1  loop
2      # non-critical section (arbitrary code!)
3      await turn = i
4      # critical section
5      turn := j
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...
  - Without fairness?

Strict alternation

Sharing variable initialized by $turn = 0$

```
1  loop
2      # non-critical section (arbitrary code!)
3      await turn = i
4      # critical section
5      turn := j
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...
  - Without fairness? **No!**

Strict alternation

Sharing variable initialized by $turn = 0$

```
1  loop
2      # non-critical section (arbitrary code!)
3      await turn = i
4      # critical section
5      turn := j
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness?

Strict alternation

Sharing variable initialized by $turn = 0$

```
1  loop
2      # non-critical section (arbitrary code!)
3      await turn = i
4      # critical section
5      turn := j
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**

Strict alternation

Sharing variable initialized by $turn = 0$

```
1  loop
2      # non-critical section (arbitrary code!)
3      await turn = i
4      # critical section
5      turn := j
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**
  - Under strong fairness?

Strict alternation

Sharing variable initialized by $turn = 0$

```
1  loop
2      # non-critical section (arbitrary code!)
3      await turn = i
4      # critical section
5      turn := j
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **No!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**
  - Under strong fairness? **No!**

Test-and-set lock (blocking)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      ⟨ await ¬lock ; lock := true ⟩
4      # critical section
5      lock := false
```

- Mutual exclusion?

Test-and-set lock (blocking)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      ⟨ await ¬lock ; lock := true ⟩
4      # critical section
5      lock := false
```

- Mutual exclusion? **Yes!**

Test-and-set lock (blocking)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      ⟨ await ¬lock ; lock := true ⟩
4      # critical section
5      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom?

Test-and-set lock (blocking)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      ⟨ await ¬lock ; lock := true ⟩
4      # critical section
5      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**

Test-and-set lock (blocking)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      ⟨ await ¬lock ; lock := true ⟩
4      # critical section
5      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...

Test-and-set lock (blocking)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      ⟨ await ¬lock ; lock := true ⟩
4      # critical section
5      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness?

Test-and-set lock (blocking)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      ⟨ await ¬lock ; lock := true ⟩
4      # critical section
5      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**

Test-and-set lock (blocking)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      ⟨ await ¬lock ; lock := true ⟩
4      # critical section
5      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness?

Test-and-set lock (blocking)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      ⟨ await ¬lock ; lock := true ⟩
4      # critical section
5      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**

Test-and-set lock (blocking)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      ⟨ await ¬lock ; lock := true ⟩
4      # critical section
5      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**
  - Under strong fairness?

Test-and-set lock (blocking)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      ⟨ await ¬lock ; lock := true ⟩
4      # critical section
5      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**
  - Under strong fairness? **Yes!**

Test-and-set lock (spinning)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      do
4          ⟨ b := lock ; lock := true ⟩
5      until ¬b
6      # critical section
7      lock := false
```

- Mutual exclusion?

Test-and-set lock (spinning)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      do
4          ⟨ b := lock ; lock := true ⟩
5      until ¬b
6      # critical section
7      lock := false
```

- Mutual exclusion? **Yes!**

Test-and-set lock (spinning)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      do
4          ⟨ b := lock ; lock := true ⟩
5      until ¬b
6      # critical section
7      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom?

Test-and-set lock (spinning)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      do
4          ⟨ b := lock ; lock := true ⟩
5      until ¬b
6      # critical section
7      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**

Test-and-set lock (spinning)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      do
4          ⟨ b := lock ; lock := true ⟩
5      until ¬b
6      # critical section
7      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...

Test-and-set lock (spinning)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      do
4          ⟨ b := lock ; lock := true ⟩
5      until ¬b
6      # critical section
7      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness?

Test-and-set lock (spinning)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      do
4          ⟨ b := lock ; lock := true ⟩
5      until ¬b
6      # critical section
7      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**

Test-and-set lock (spinning)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      do
4          ⟨ b := lock ; lock := true ⟩
5      until ¬b
6      # critical section
7      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness?

Test-and-set lock (spinning)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      do
4          ⟨ b := lock ; lock := true ⟩
5      until ¬b
6      # critical section
7      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**

Test-and-set lock (spinning)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      do
4          ⟨ b := lock ; lock := true ⟩
5      until ¬b
6      # critical section
7      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**
  - Under strong fairness?

Test-and-set lock (spinning)

Sharing variable initialized by $lock = \bot$

```
1  loop
2      # non-critical section
3      do
4          ⟨ b := lock ; lock := true ⟩
5      until ¬b
6      # critical section
7      lock := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **No!**
  - Under strong fairness? **No!**

Peterson's algorithm

Sharing array initialized by $flag[0..1] = \bot$ and variable $turn = 0$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      turn := j
5      await ¬flag[j] ∨ turn = i
6      # critical section
7      flag[i] := false
```

- Mutual exclusion?

Peterson's algorithm

Sharing array initialized by $flag[0..1] = \bot$ and variable $turn = 0$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      turn := j
5      await ¬flag[j] ∨ turn = i
6      # critical section
7      flag[i] := false
```

- Mutual exclusion? **Yes!**

Peterson's algorithm

Sharing array initialized by $flag[0..1] = \bot$ and variable $turn = 0$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      turn := j
5      await ¬flag[j] ∨ turn = i
6      # critical section
7      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom?

Peterson's algorithm

Sharing array initialized by $flag[0..1] = \bot$ and variable $turn = 0$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      turn := j
5      await ¬flag[j] ∨ turn = i
6      # critical section
7      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**

Peterson's algorithm

Sharing array initialized by $flag[0..1] = \bot$ and variable $turn = 0$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      turn := j
5      await ¬flag[j] ∨ turn = i
6      # critical section
7      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...

Peterson's algorithm

Sharing array initialized by $flag[0..1] = \bot$ and variable $turn = 0$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      turn := j
5      await ¬flag[j] ∨ turn = i
6      # critical section
7      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness?

Peterson's algorithm

Sharing array initialized by $flag[0..1] = \bot$ and variable $turn = 0$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      turn := j
5      await ¬flag[j] ∨ turn = i
6      # critical section
7      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**

Peterson's algorithm

Sharing array initialized by $flag[0..1] = \bot$ and variable $turn = 0$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      turn := j
5      await ¬flag[j] ∨ turn = i
6      # critical section
7      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness?

Peterson's algorithm

Sharing array initialized by $flag[0..1] = \bot$ and variable $turn = 0$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      turn := j
5      await ¬flag[j] ∨ turn = i
6      # critical section
7      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **Yes!**

Peterson's algorithm

Sharing array initialized by $flag[0..1] = \bot$ and variable $turn = 0$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      turn := j
5      await ¬flag[j] ∨ turn = i
6      # critical section
7      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **Yes!**
  - Under strong fairness?

Peterson's algorithm

Sharing array initialized by $flag[0..1] = \bot$ and variable $turn = 0$

```
1  loop
2      # non-critical section
3      flag[i] := true
4      turn := j
5      await ¬flag[j] ∨ turn = i
6      # critical section
7      flag[i] := false
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **Yes!**
  - Under strong fairness? **Yes!**

Ticket lock

Sharing variables $next = 0$ and $serving = 0$

```
1  loop
2      # non-critical section
3      ⟨ my := next; next := next + 1 ⟩
4      await serving = my
5      # critical section
6      serving := serving + 1
```

- Mutual exclusion?

Ticket lock

Sharing variables $next = 0$ and $serving = 0$

```
1  loop
2      # non-critical section
3      ⟨ my := next; next := next + 1 ⟩
4      await serving = my
5      # critical section
6      serving := serving + 1
```

- Mutual exclusion? **Yes!**

Ticket lock

Sharing variables $next = 0$ and $serving = 0$

```
1  loop
2      # non-critical section
3      ⟨ my := next; next := next + 1 ⟩
4      await serving = my
5      # critical section
6      serving := serving + 1
```

- Mutual exclusion? **Yes!**
- Deadlock freedom?

Ticket lock

Sharing variables $next = 0$ and $serving = 0$

```
1  loop
2      # non-critical section
3      ⟨ my := next; next := next + 1 ⟩
4      await serving = my
5      # critical section
6      serving := serving + 1
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**

Ticket lock

Sharing variables $next = 0$ and $serving = 0$

```
1  loop
2      # non-critical section
3      ⟨ my := next; next := next + 1 ⟩
4      await serving = my
5      # critical section
6      serving := serving + 1
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...

Ticket lock

Sharing variables $next = 0$ and $serving = 0$

```
1  loop
2      # non-critical section
3      ⟨ my := next; next := next + 1 ⟩
4      await serving = my
5      # critical section
6      serving := serving + 1
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness?

Ticket lock

Sharing variables $next = 0$ and $serving = 0$

```
1  loop
2      # non-critical section
3      ⟨ my := next; next := next + 1 ⟩
4      await serving = my
5      # critical section
6      serving := serving + 1
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**

Ticket lock

Sharing variables $next = 0$ and $serving = 0$

```
1  loop
2      # non-critical section
3      ⟨ my := next; next := next + 1 ⟩
4      await serving = my
5      # critical section
6      serving := serving + 1
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness?

Ticket lock

Sharing variables $next = 0$ and $serving = 0$

```
1  loop
2      # non-critical section
3      ⟨ my := next; next := next + 1 ⟩
4      await serving = my
5      # critical section
6      serving := serving + 1
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **Yes!**

Ticket lock

Sharing variables $next = 0$ and $serving = 0$

```
1  loop
2      # non-critical section
3      ⟨ my := next; next := next + 1 ⟩
4      await serving = my
5      # critical section
6      serving := serving + 1
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **Yes!**
  - Under strong fairness?

Ticket lock

Sharing variables $next = 0$ and $serving = 0$

```
1  loop
2      # non-critical section
3      ⟨ my := next; next := next + 1 ⟩
4      await serving = my
5      # critical section
6      serving := serving + 1
```

- Mutual exclusion? **Yes!**
- Deadlock freedom? **Yes!**
- Starvation freedom...
  - Without fairness? **No!**
  - Under weak fairness? **Yes!**
  - Under strong fairness? **Yes!**