# Concurrent Algorithms and Data Structures

# Concurrent Algorithms and Data Structures

- Slides uploaded to “Studium”
- Please, download the slides first, and only then run the animations
- Report problems with downloading/ watching the slides
- Low-Resolution, No-Animation pdfs are available (around 50 MBs)

# Concurrent Algorithms and Data Structures

- Slides uploaded to “Studium”
- Please, download the slides first, and only then run the animations
- Report problems with downloading/ watching the slides
- Low-Resolution, No-Animation pdfs are available (around 50 MBs)

- The first theory assignment is now available.
- I will explain the problems in the assignment in today’s lecture.
- Deadline: Sunday, next week.

# Outline

- Overview
- Abstract Data Types
- Sequential Programs
- Concurrent Programs
- Coarse-Grained Locking
- Fine-Grained Locking
- Lazy Lists
- Optimistic  Lists
- Treiber Stack  Algorithm
- Michael&Scott Queue Algorithm

# Outline

- Overview
- Abstract Data Types
- Sequential Programs
- Concurrent Programs
- Coarse-Grained Locking
- Fine-Grained Locking
- Lazy Lists
- Optimistic  Lists
- Treiber Stack  Algorithm
- Michael&Scott Queue Algorithm

# Sets | Coarse-Grained Synchronization

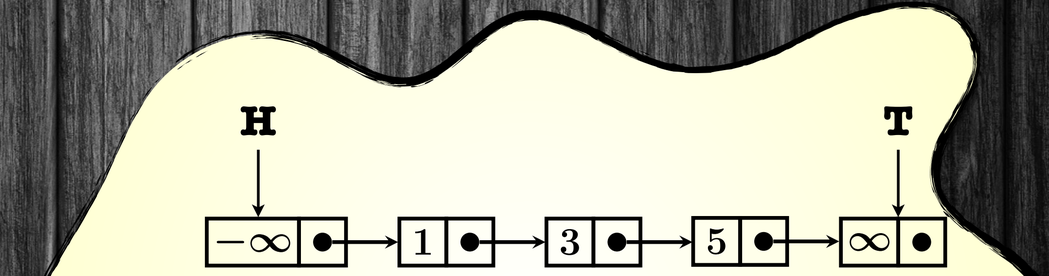
```
add(k):
Node p, c
Lock l
1  lock l
2  p = H;
3  c = p.next
4  while (c.key < k)
5    p = c;
6    c = c.next;
7  if (c.key = k)
8    return false
9  else
10   n = new Node(k,-)
11   n.next = c
12   p.next = n
13   return true
14 unlock l
```

LP

linearization point

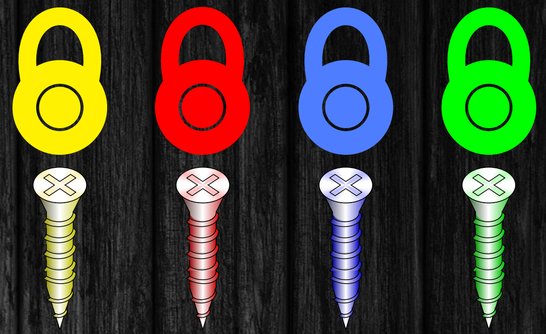
code + linearization point

method implementation

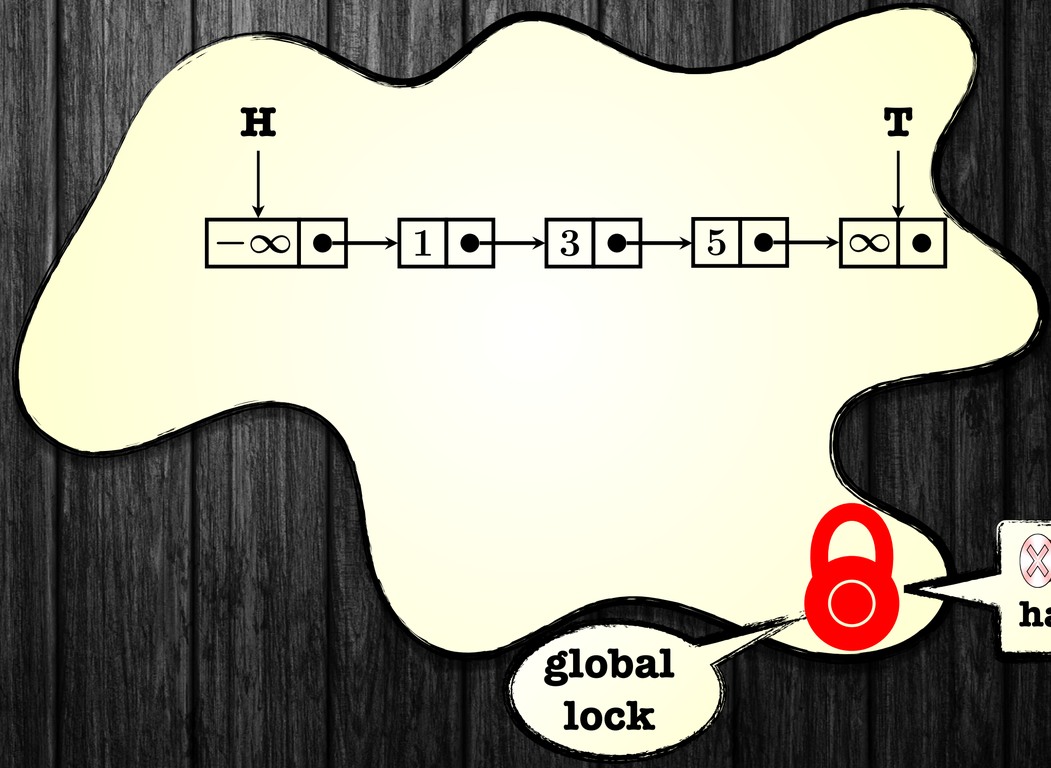



global lock

# Sets Coarse-Grained Synchronization



At most one thread has lock

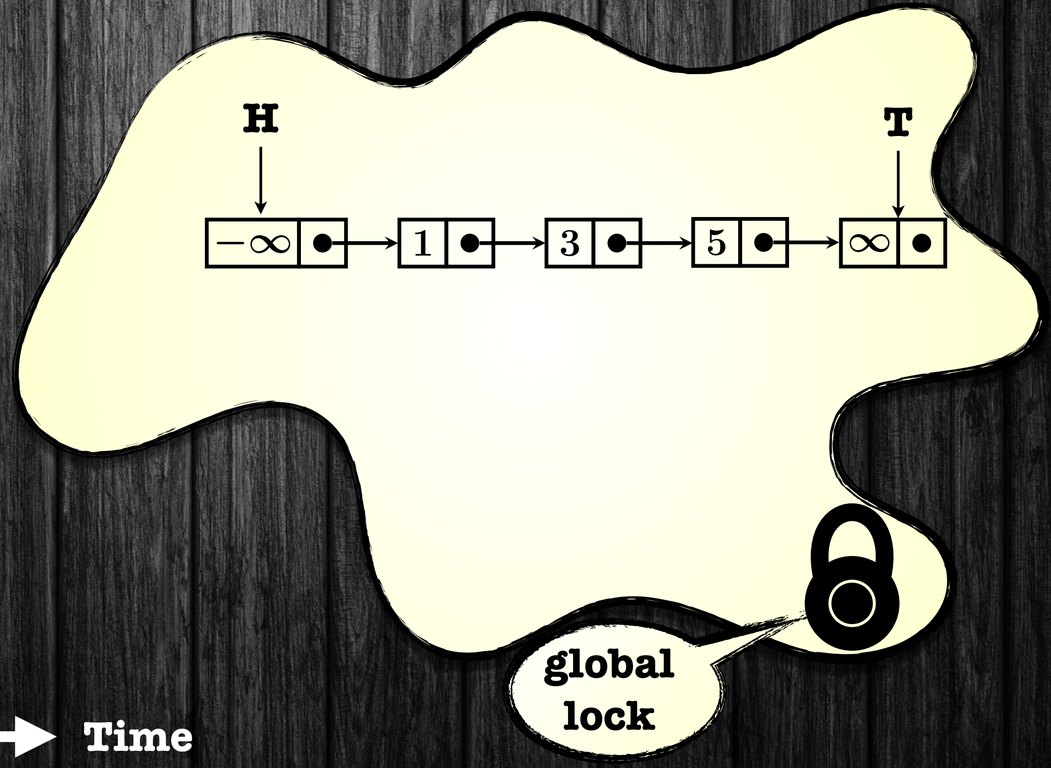


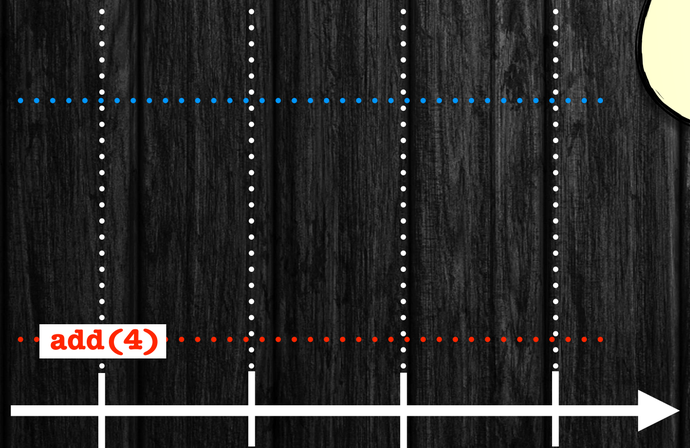
# Sets Coarse-Grained Synchronization

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```




```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



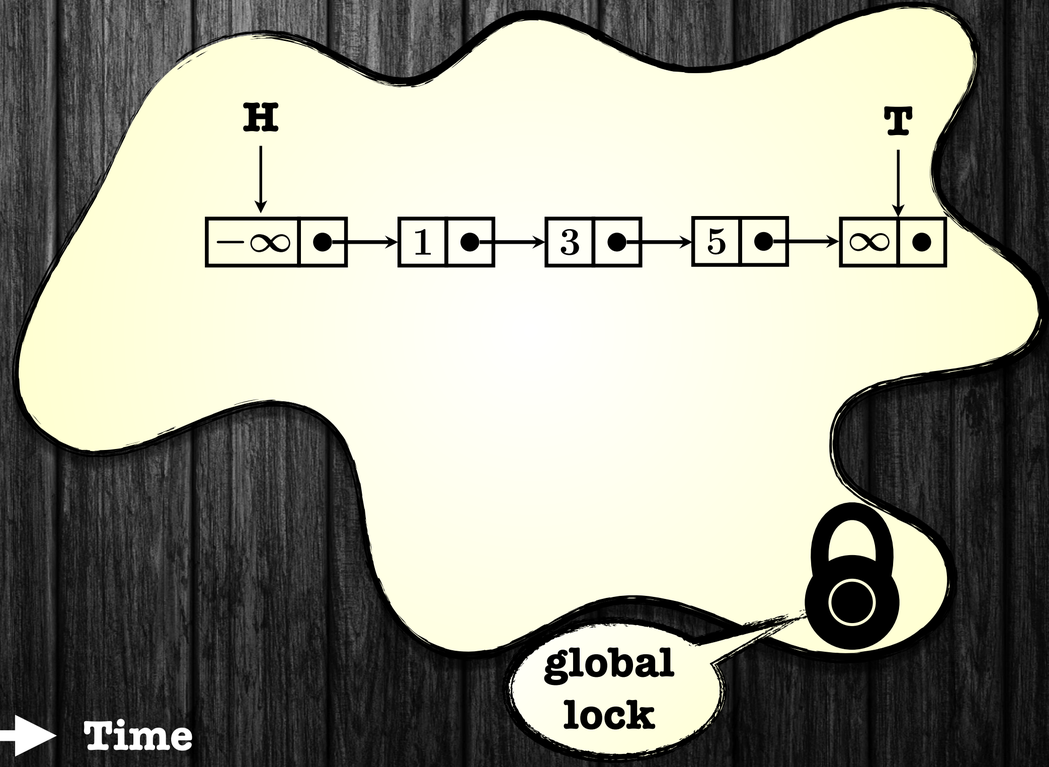
# Sets Coarse-Grained Synchronization

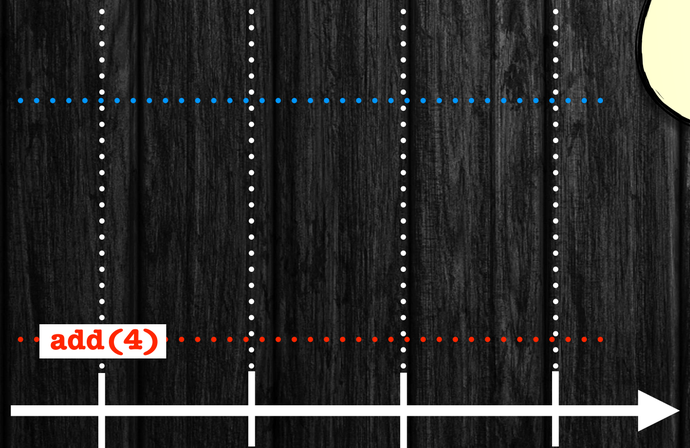
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



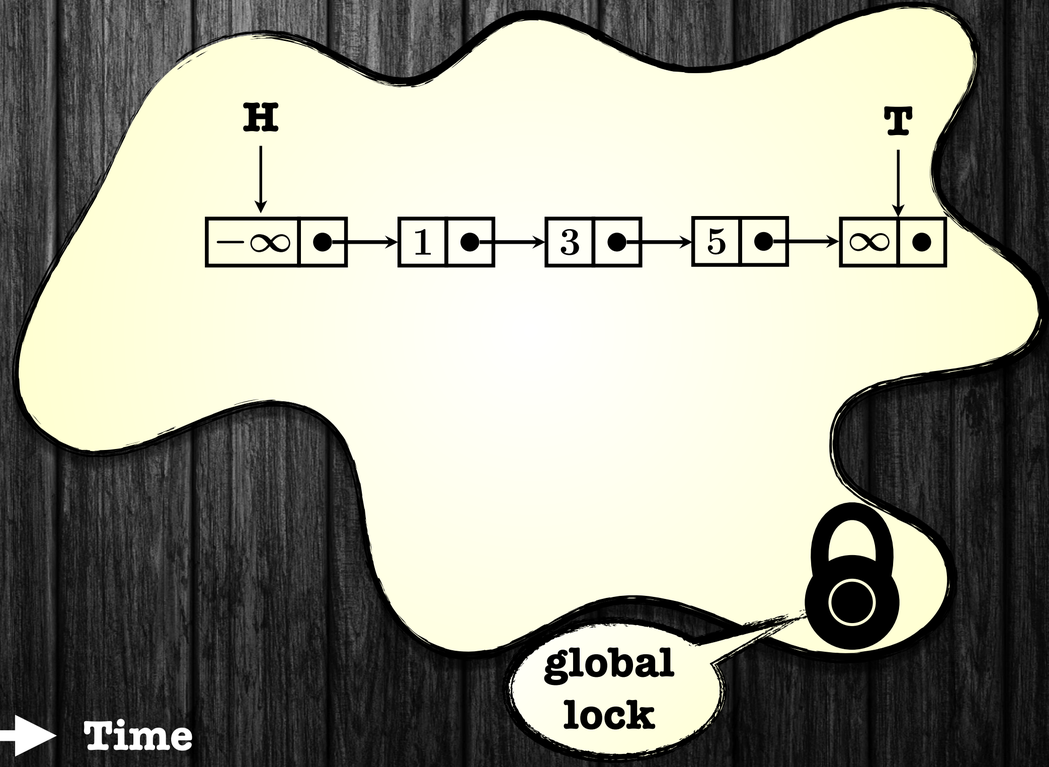
# Sets Coarse-Grained Synchronization

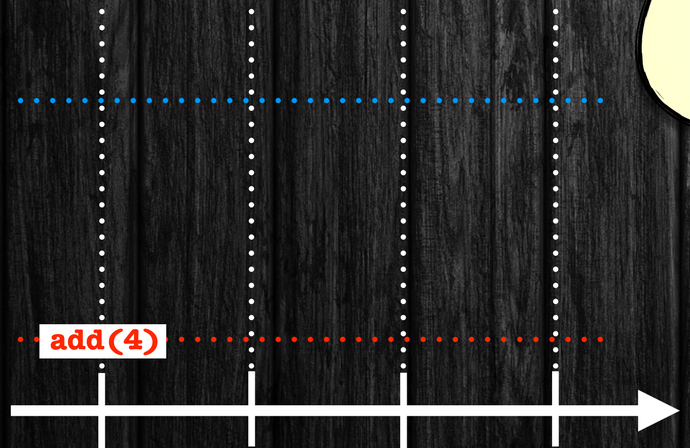
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



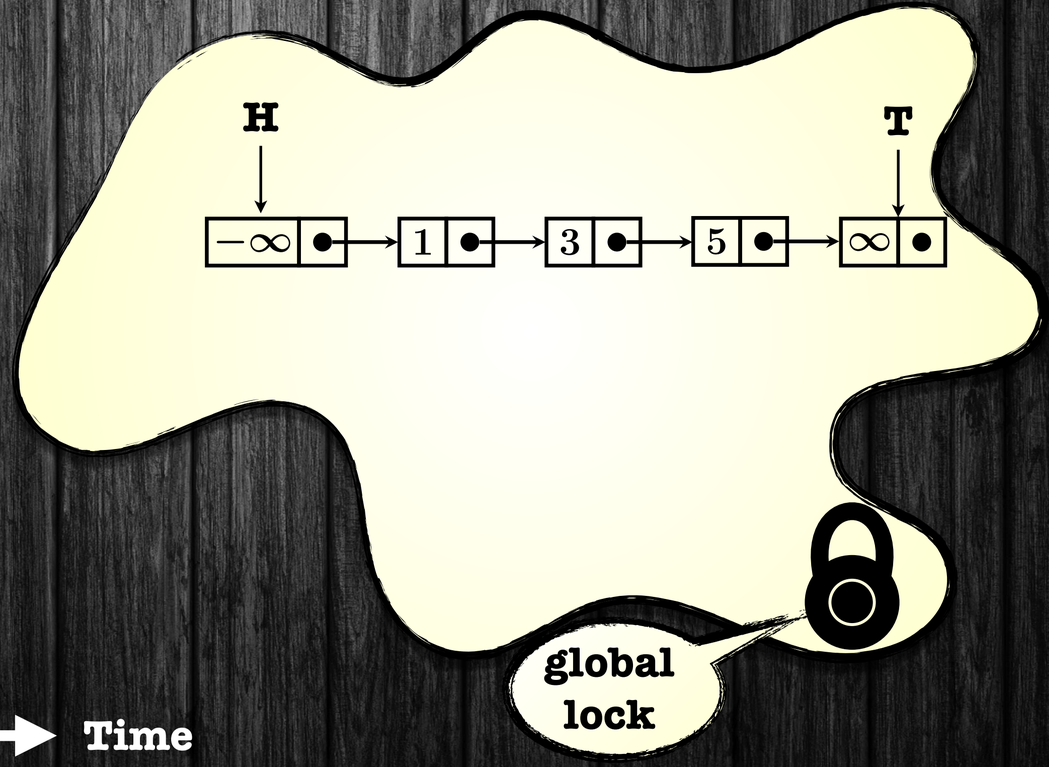
# Sets Coarse-Grained Synchronization

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```





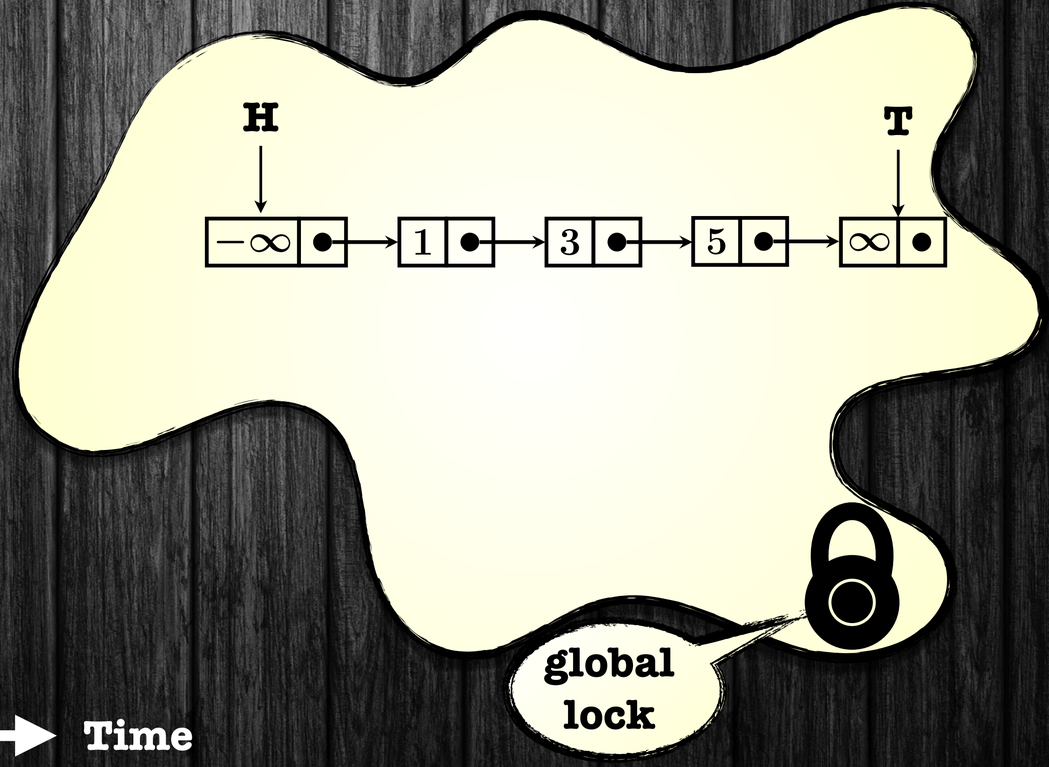

# Sets Coarse-Grained Synchronization

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```






# Sets Coarse-Grained Synchronization

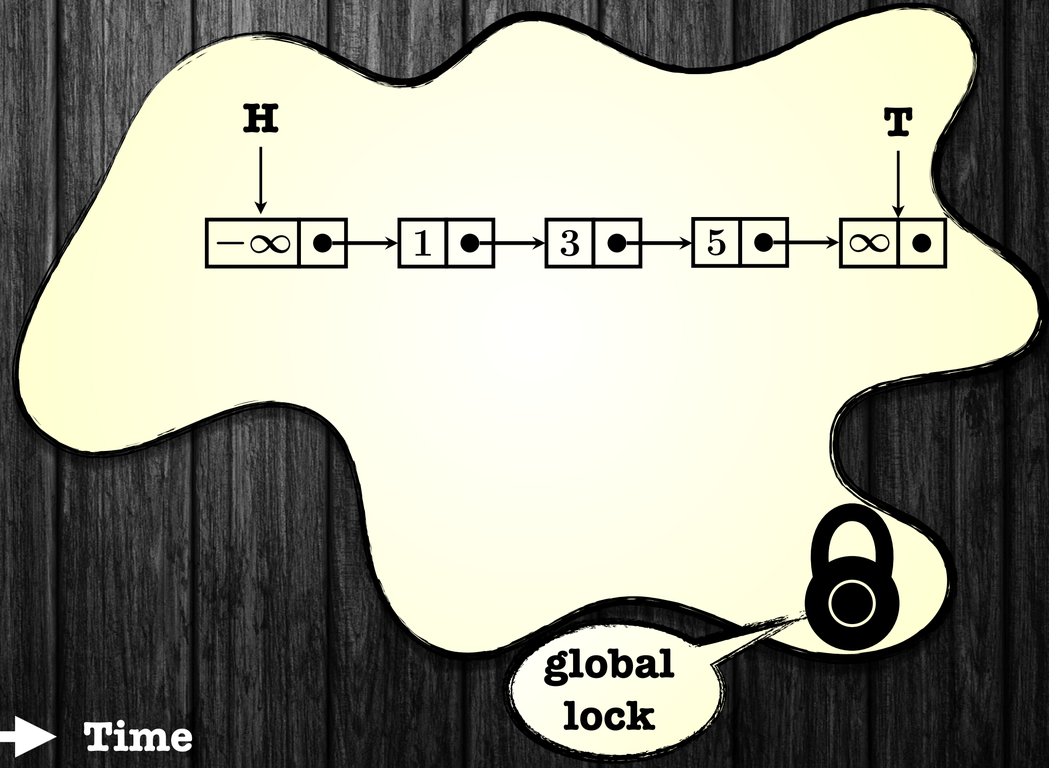
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



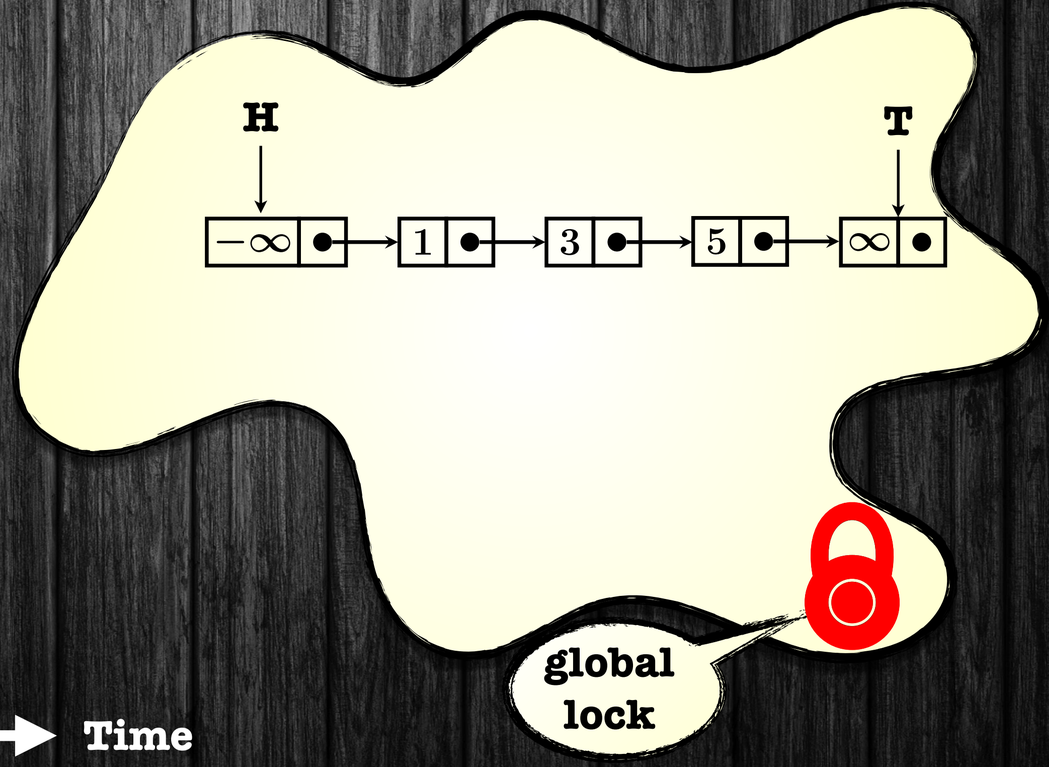
# Sets Coarse-Grained Synchronization

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```





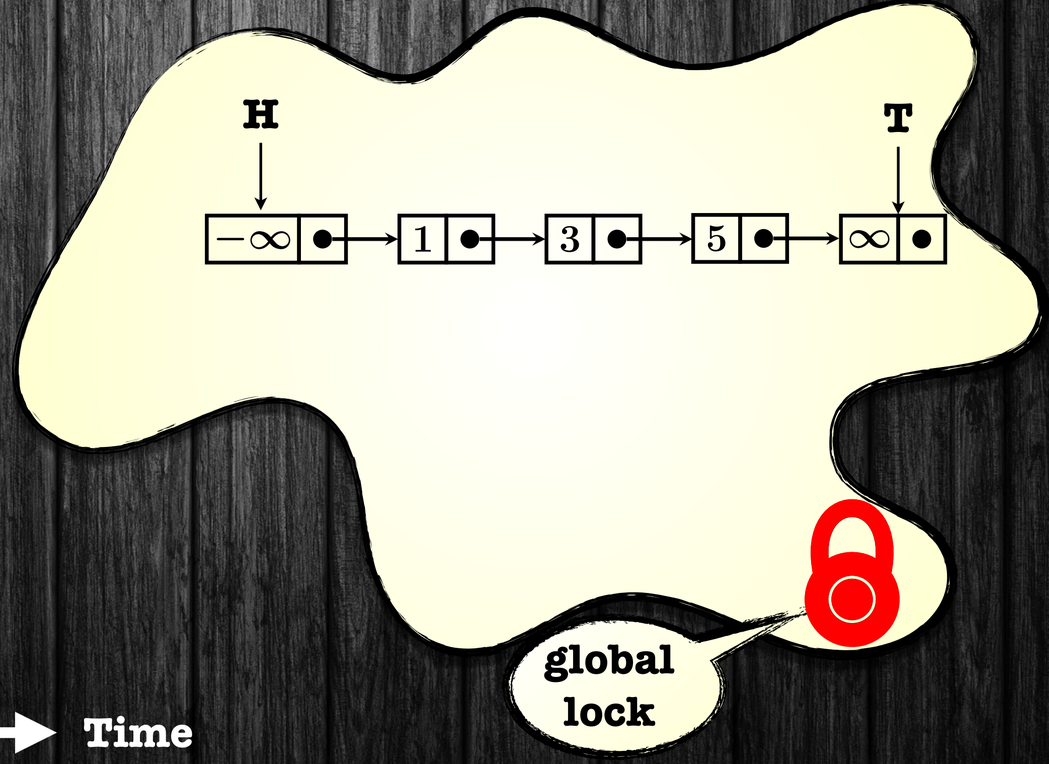

# Sets Coarse-Grained Synchronization

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```





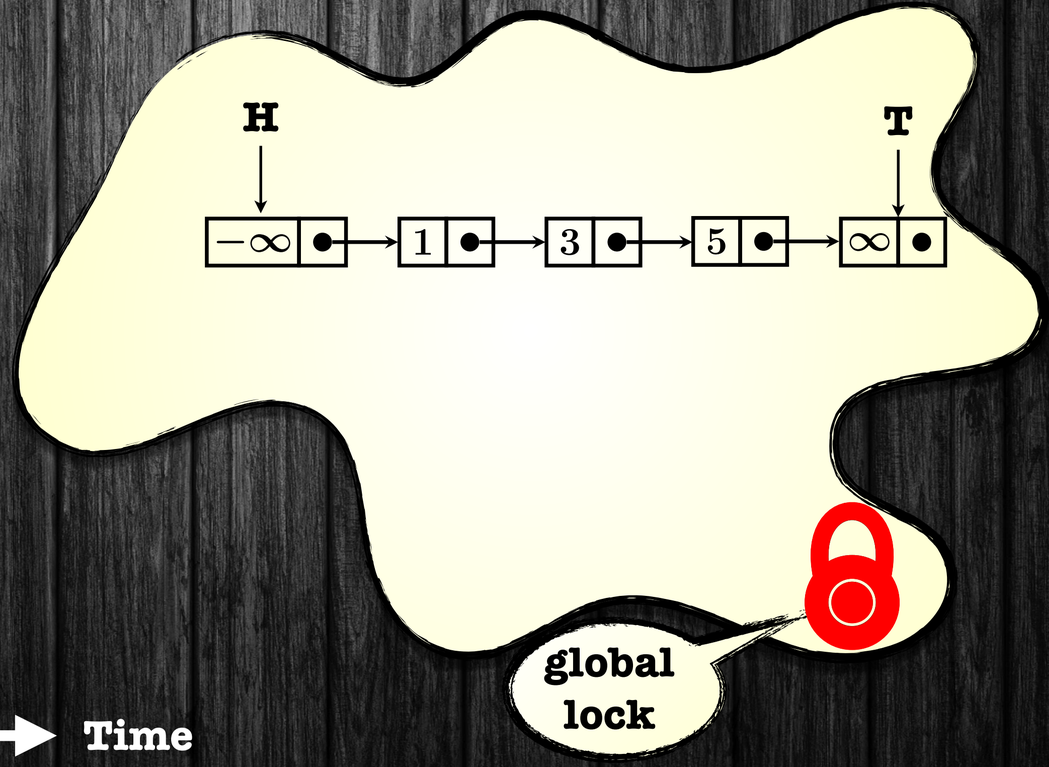

# Sets Coarse-Grained Synchronization

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

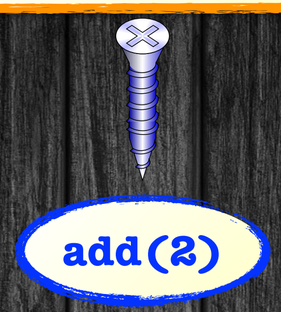




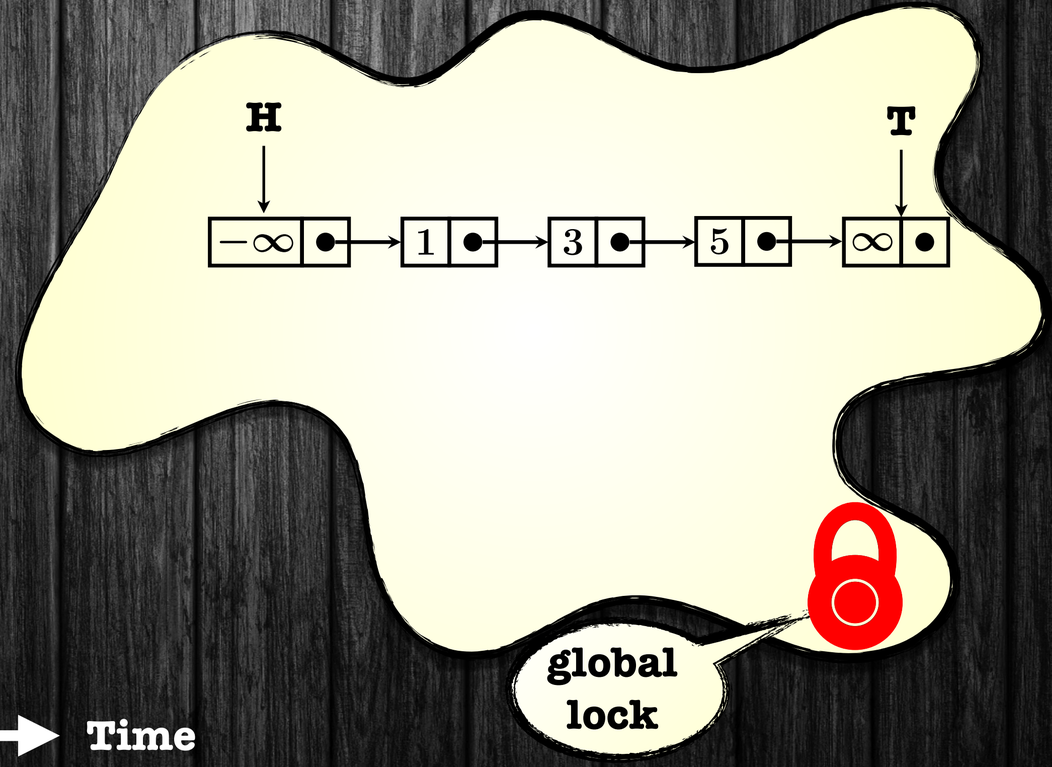

# Sets Coarse-Grained Synchronization

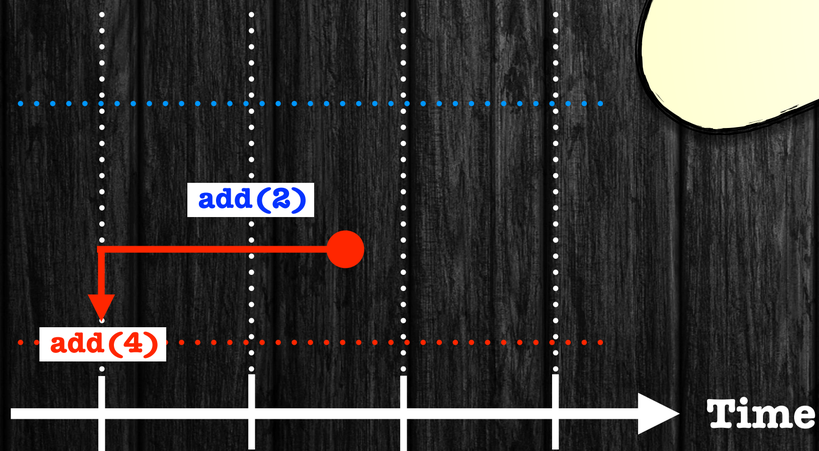
```
add(k):
•••
• lock l;   LP
• •••
• unlock l
```




```
add(k):
•••
• lock l;   LP
• •••
• unlock l
```



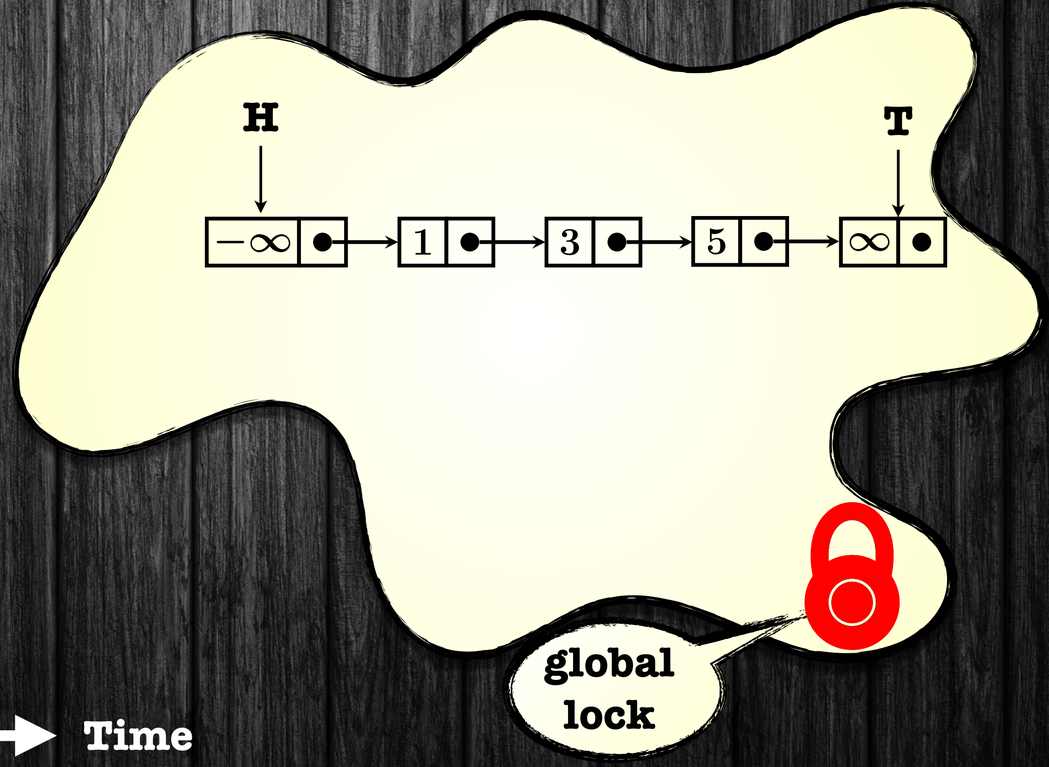
# Sets Coarse-Grained Synchronization

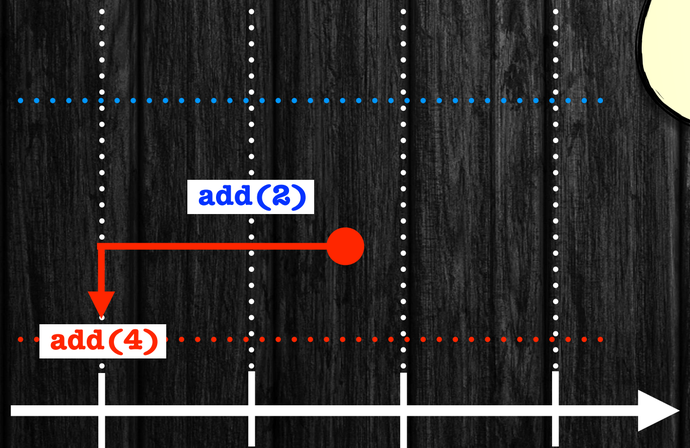
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



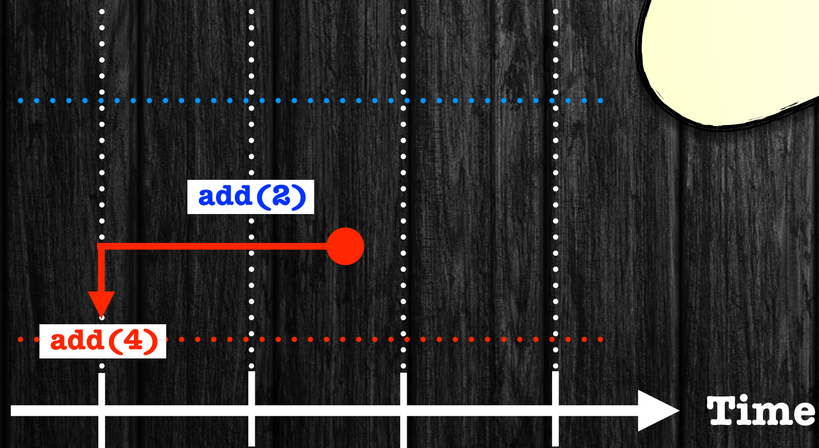
# Sets Coarse-Grained Synchronization

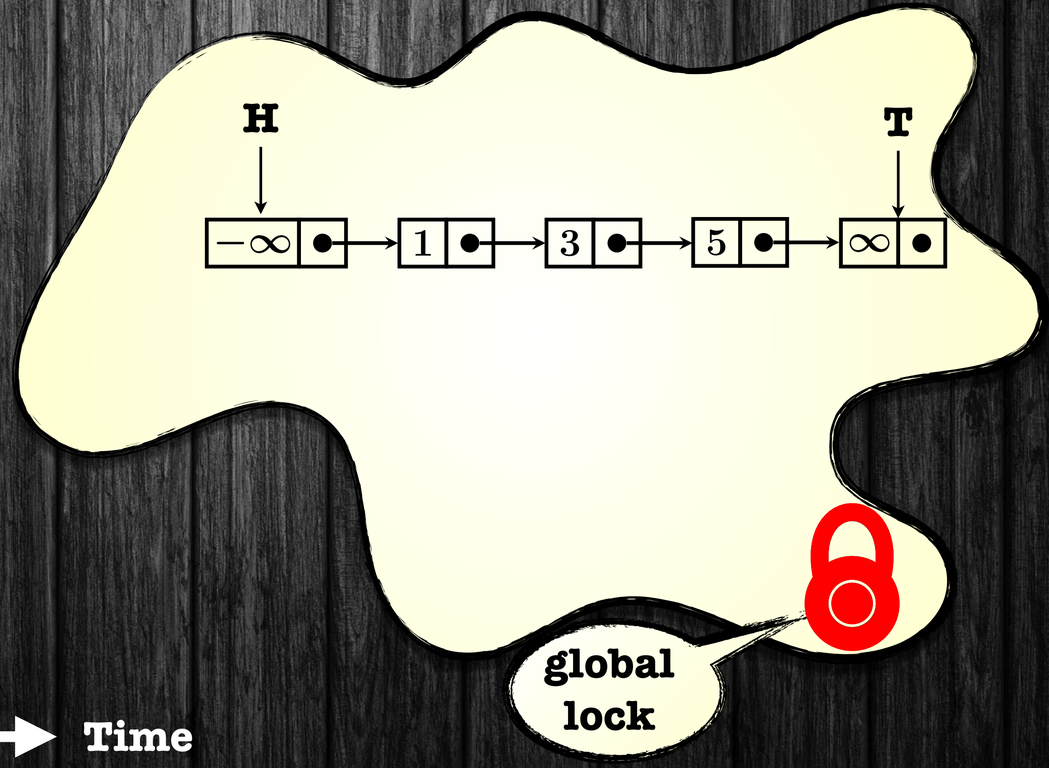
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



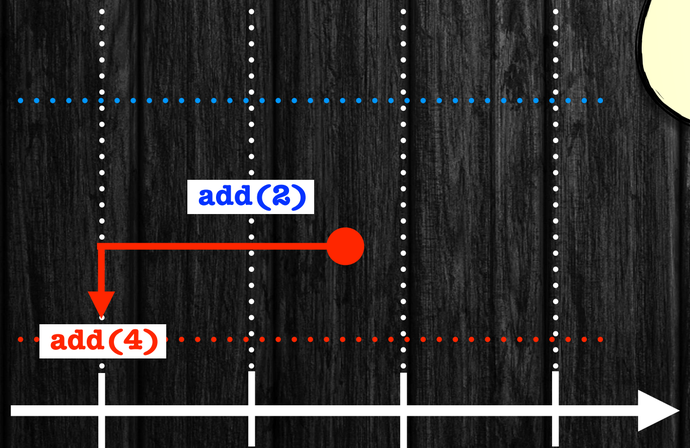
# Sets Coarse-Grained Synchronization

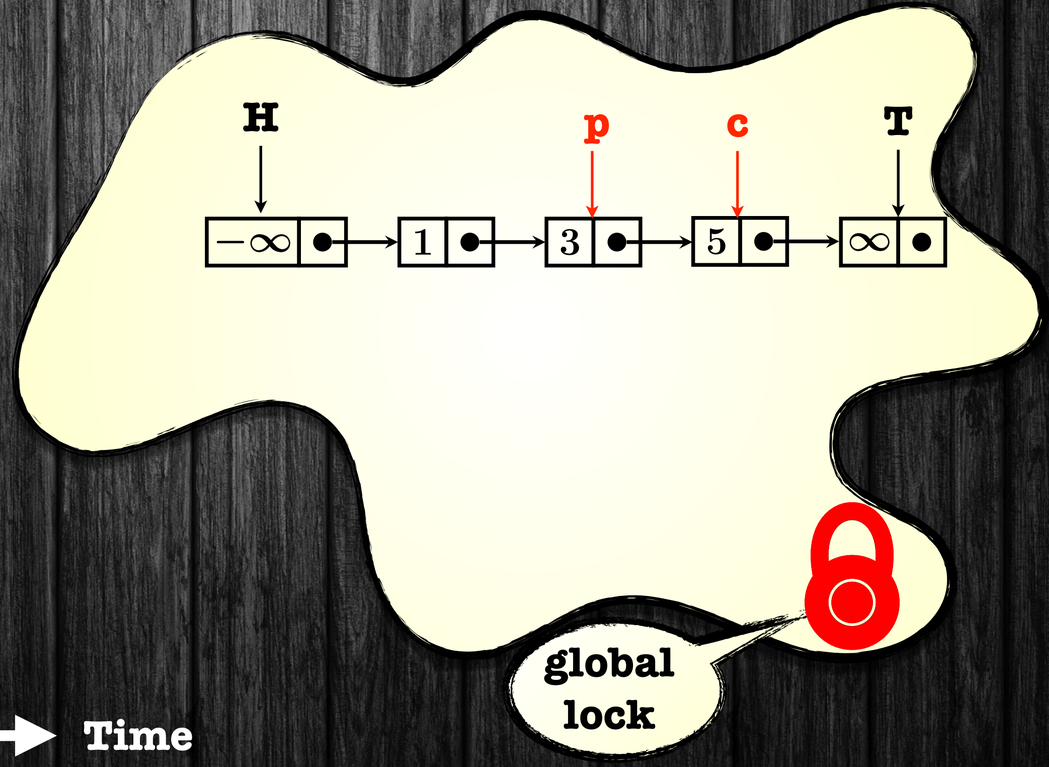
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



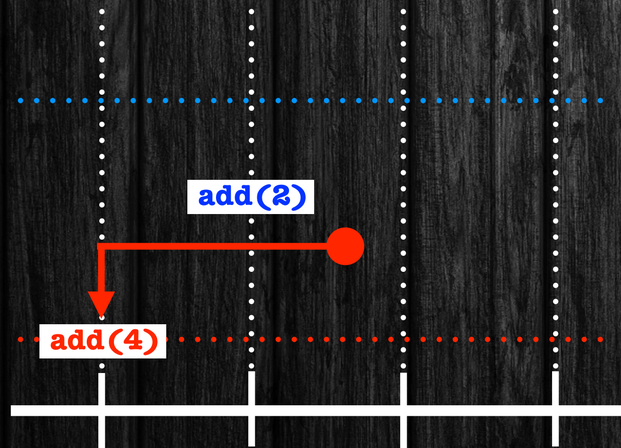
# Sets Coarse-Grained Synchronization

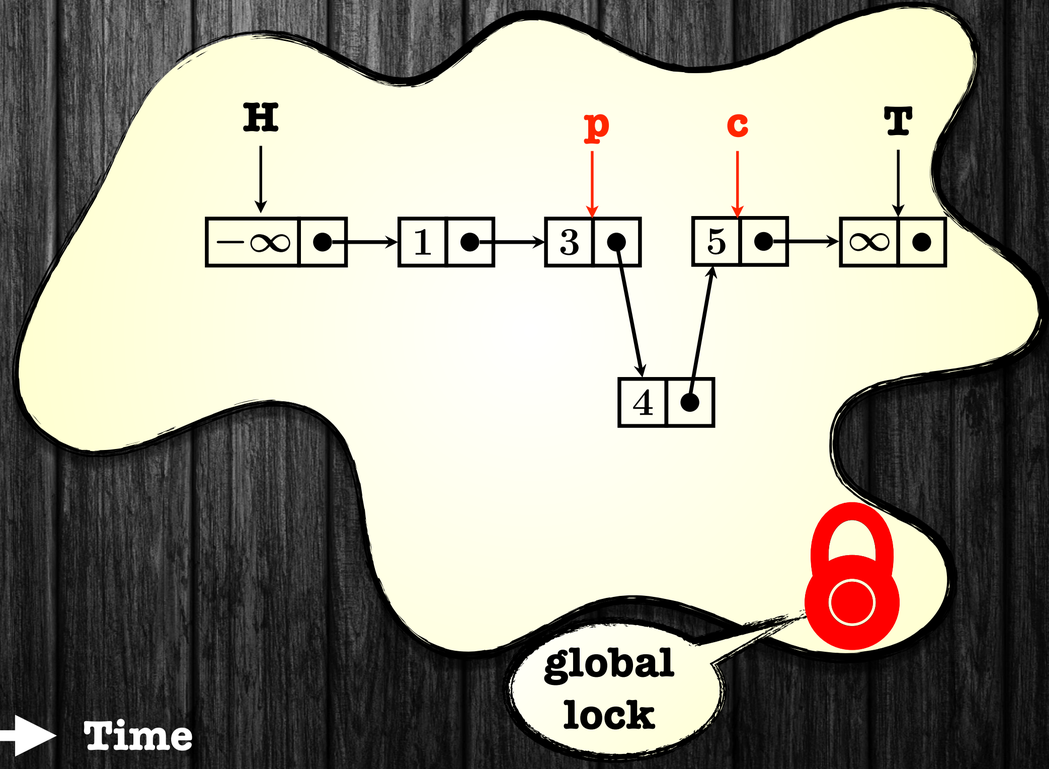
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



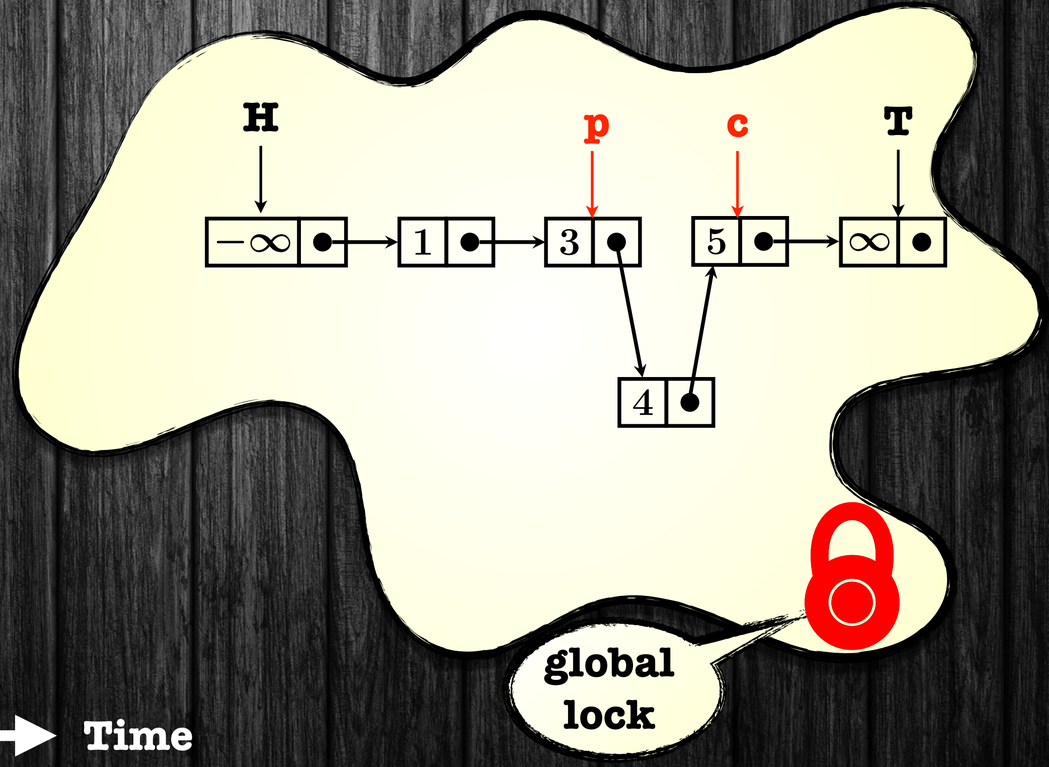
# Sets Coarse-Grained Synchronization

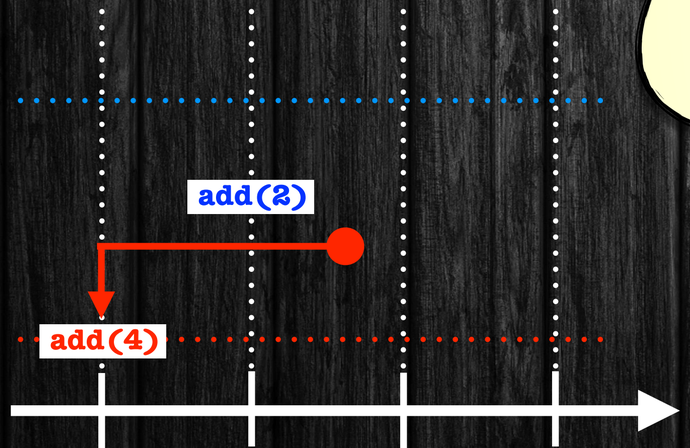
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



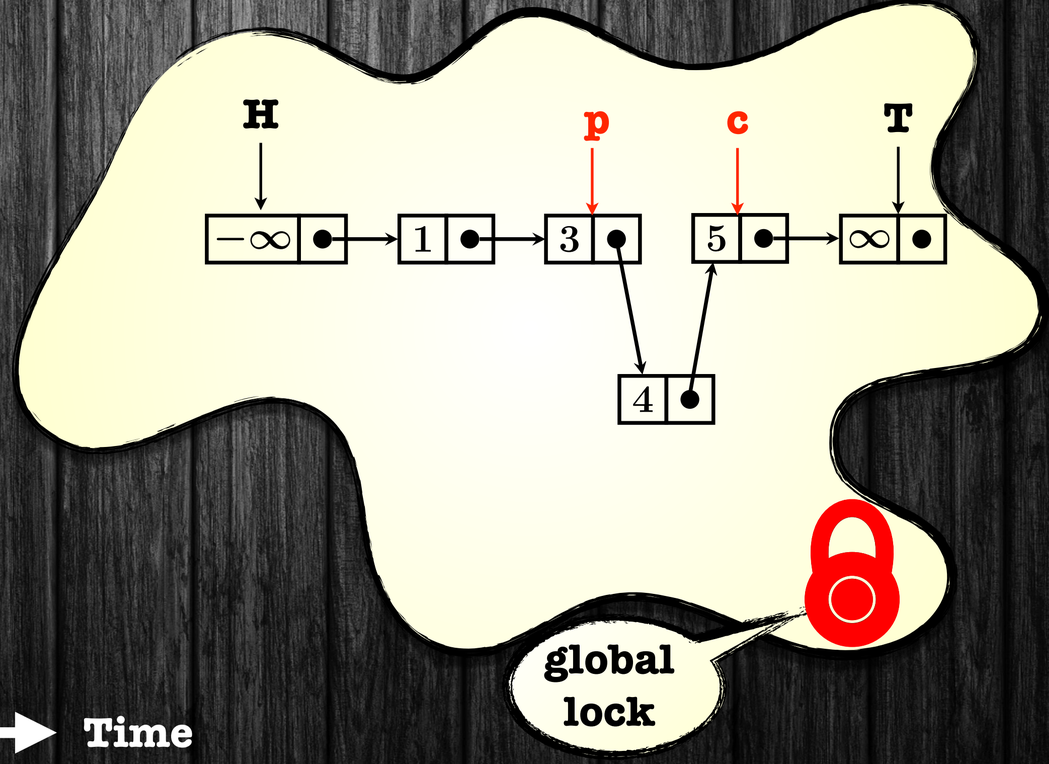
# Sets Coarse-Grained Synchronization

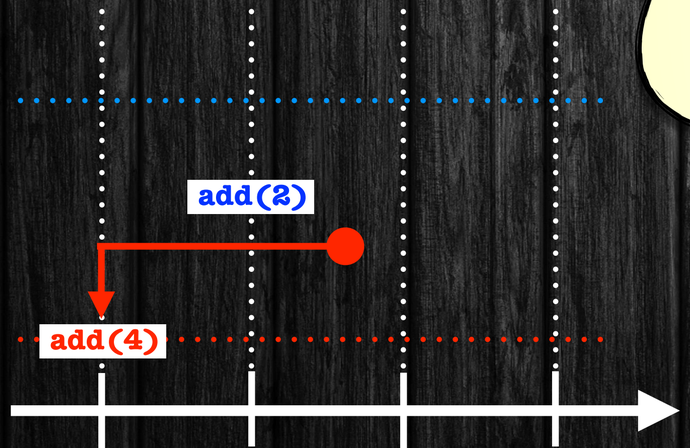
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



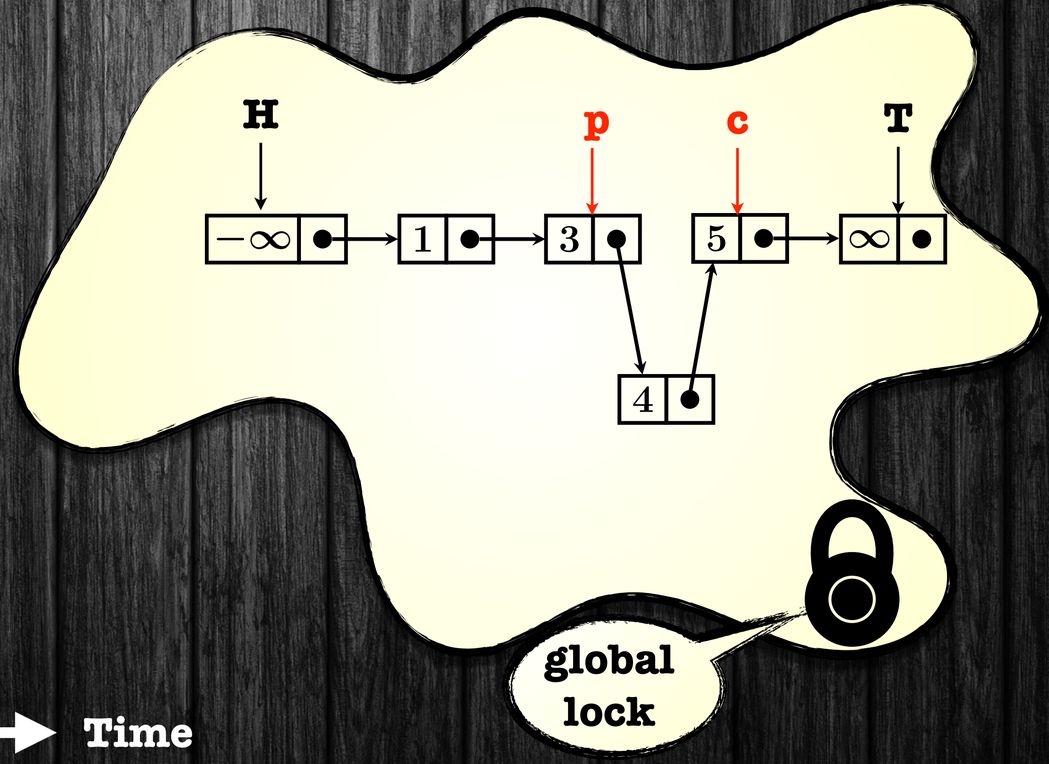
# Sets Coarse-Grained Synchronization

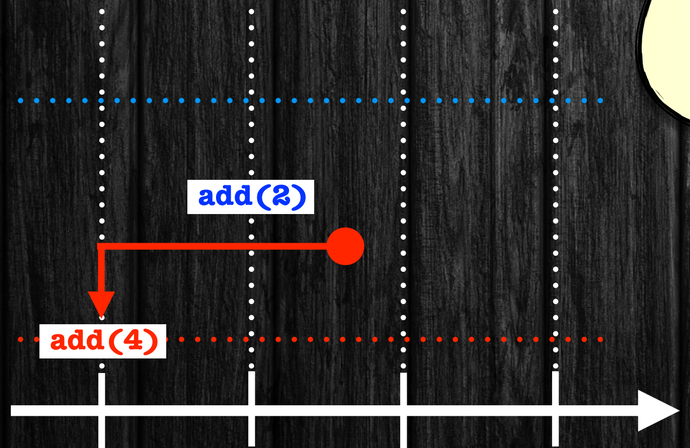
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



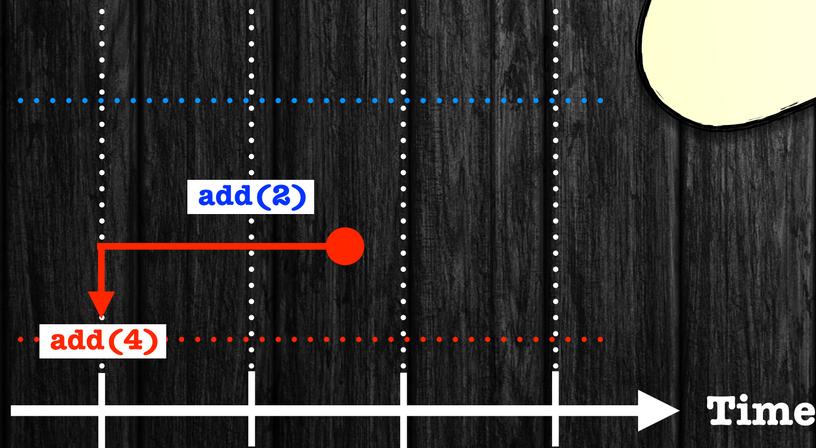
# Sets Coarse-Grained Synchronization

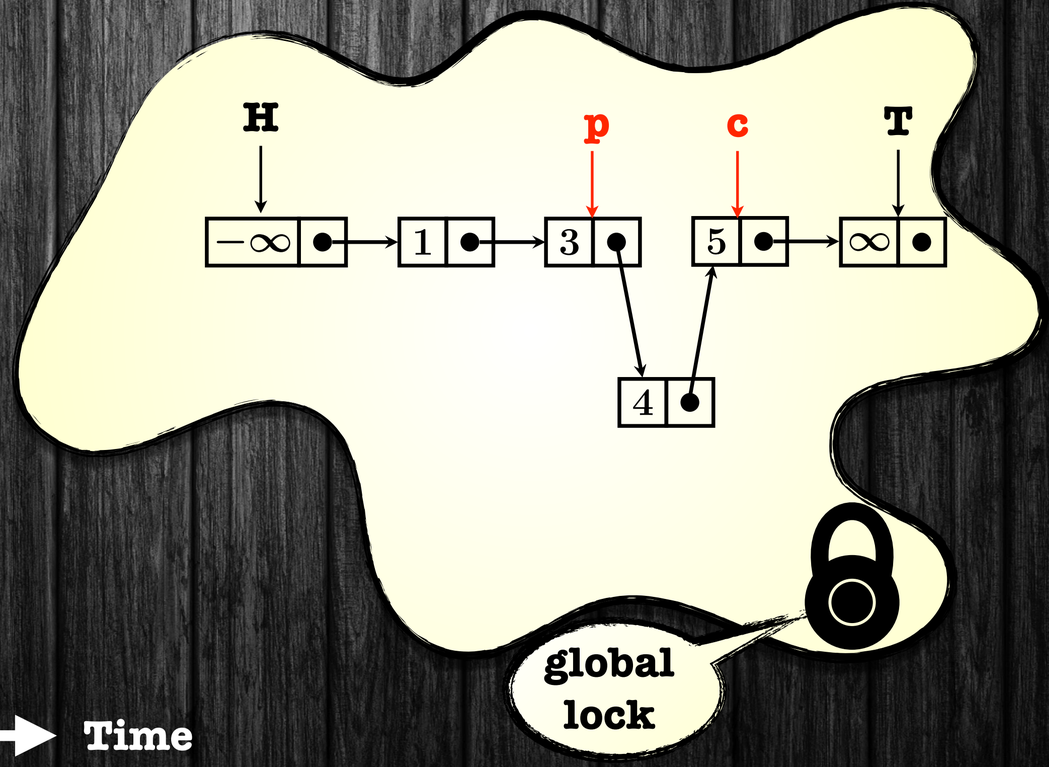
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



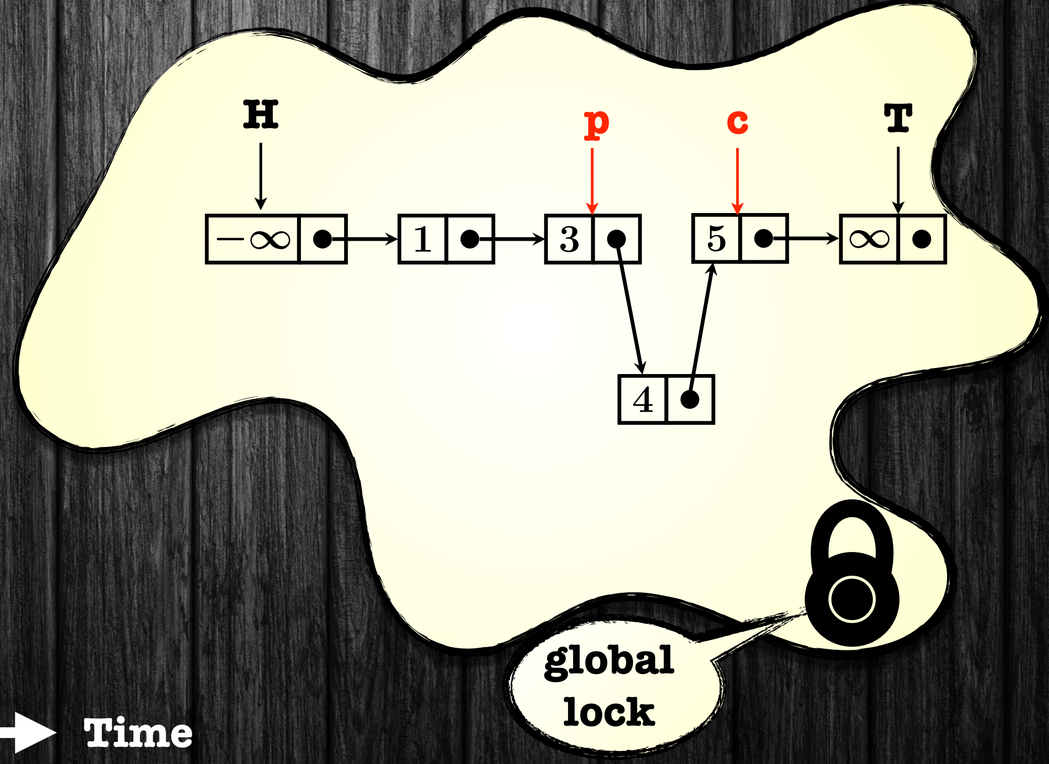
# Sets Coarse-Grained Synchronization

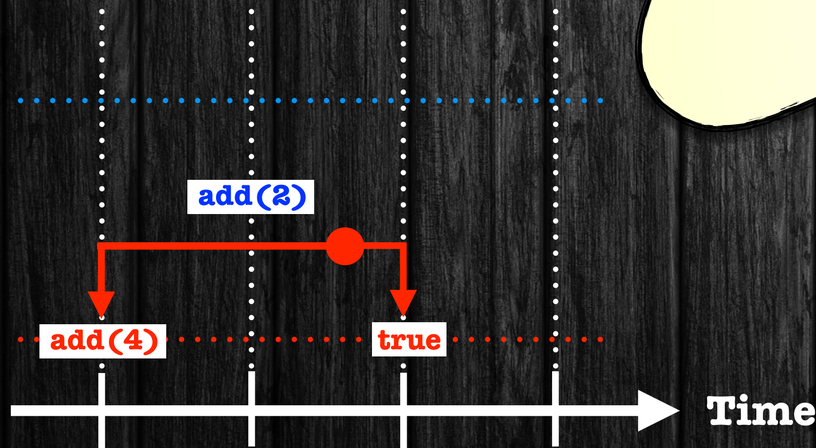
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



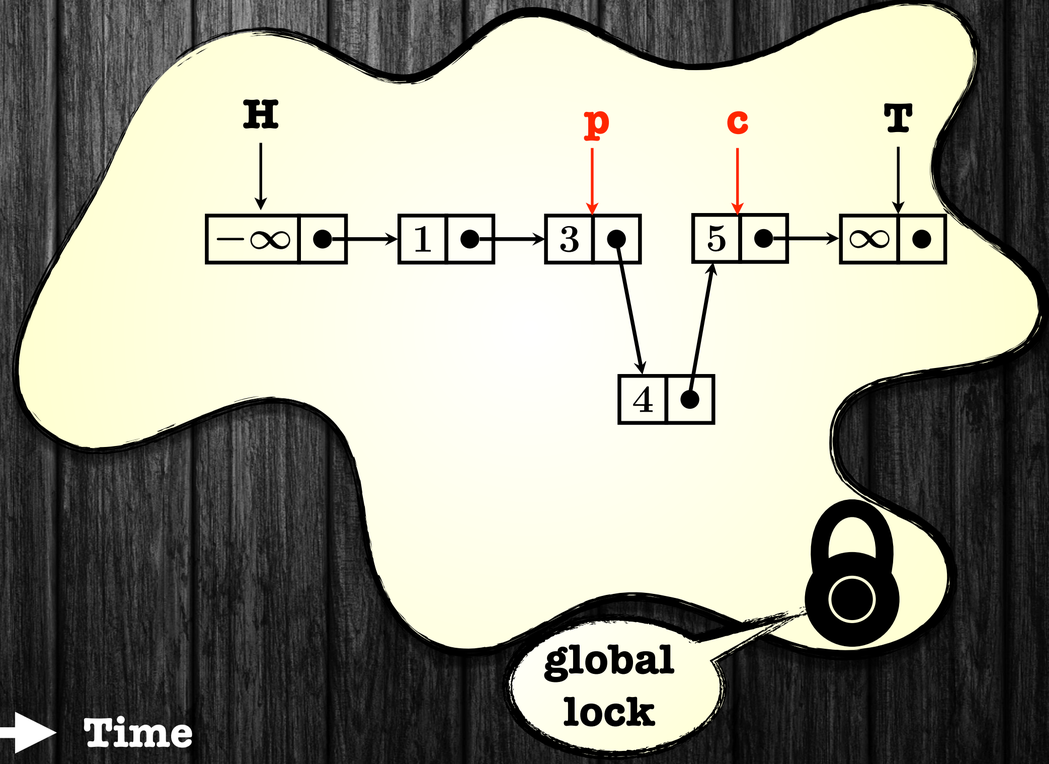
# Sets Coarse-Grained Synchronization

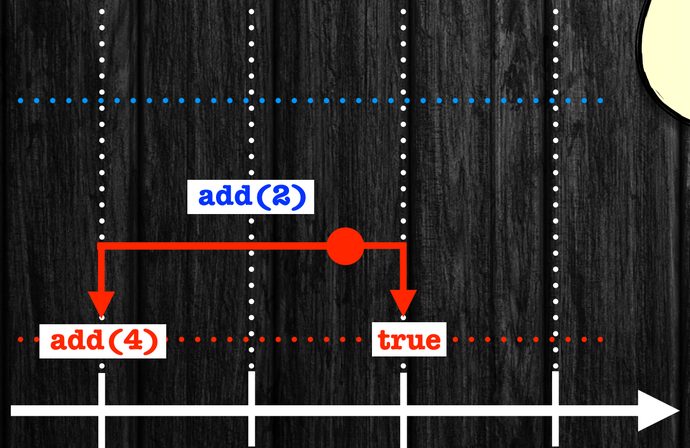
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



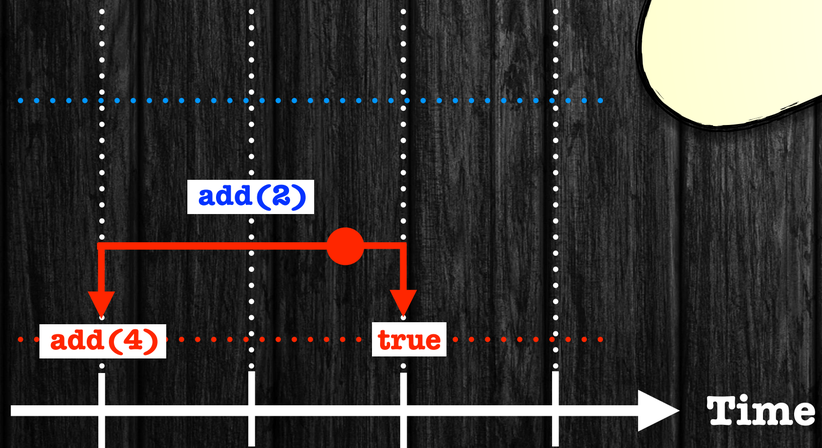
# Sets Coarse-Grained Synchronization

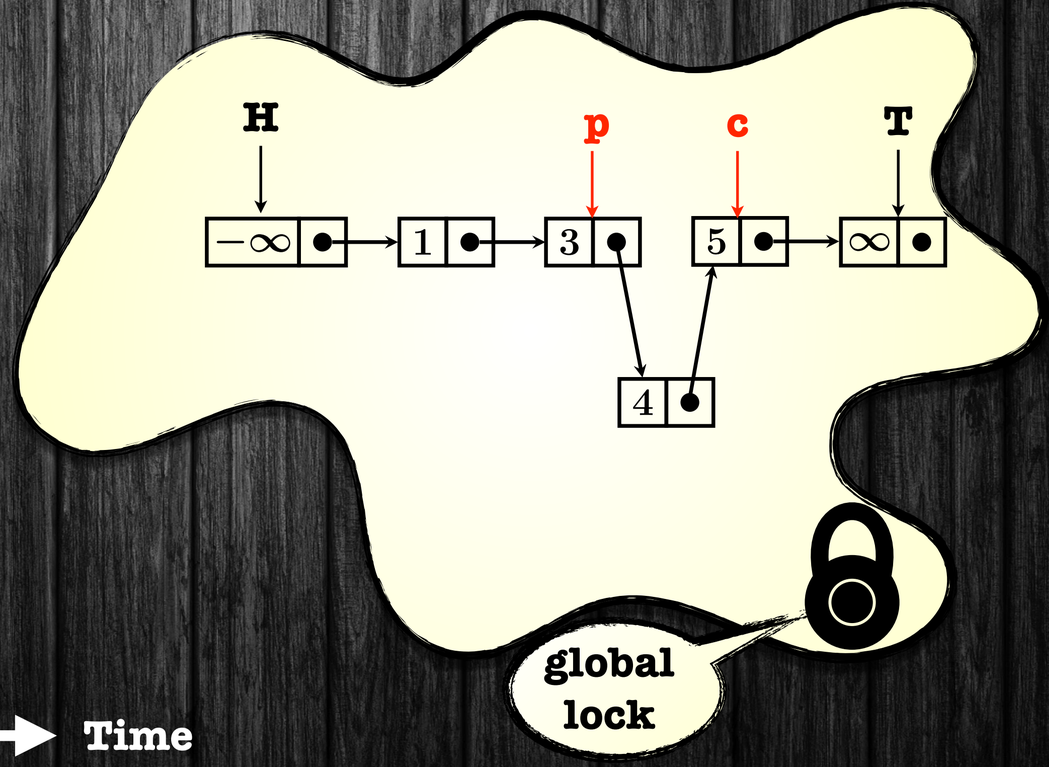
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l  LP
• •••
• unlock l
```



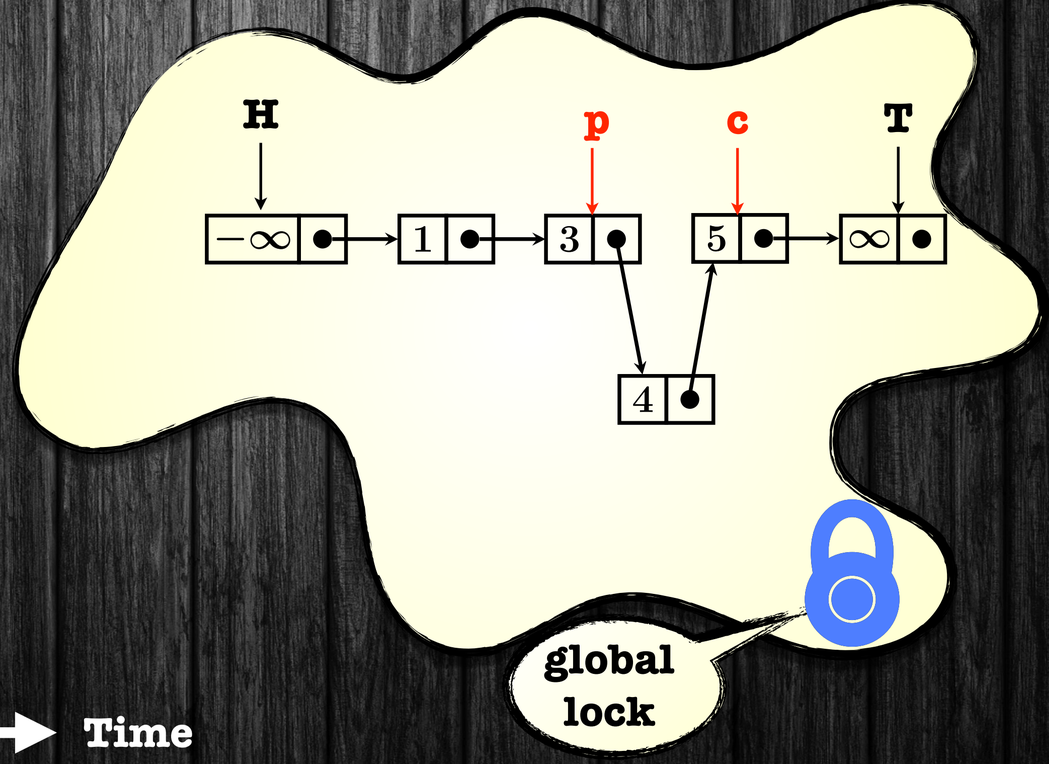
# Sets Coarse-Grained Synchronization

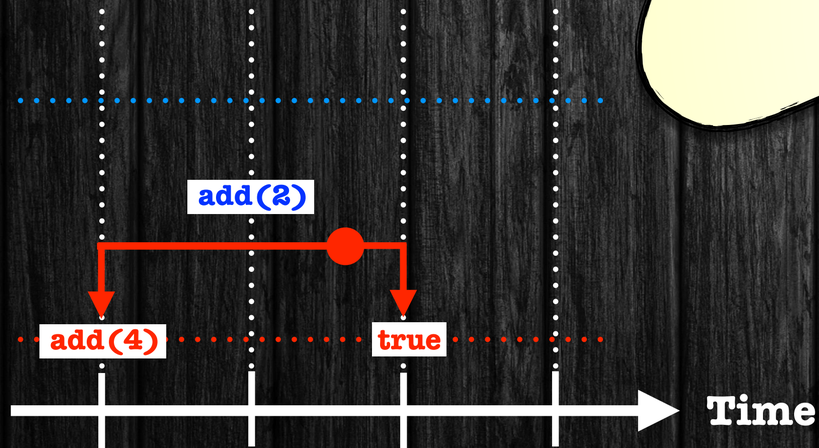
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l  LP
• •••
• unlock l
```



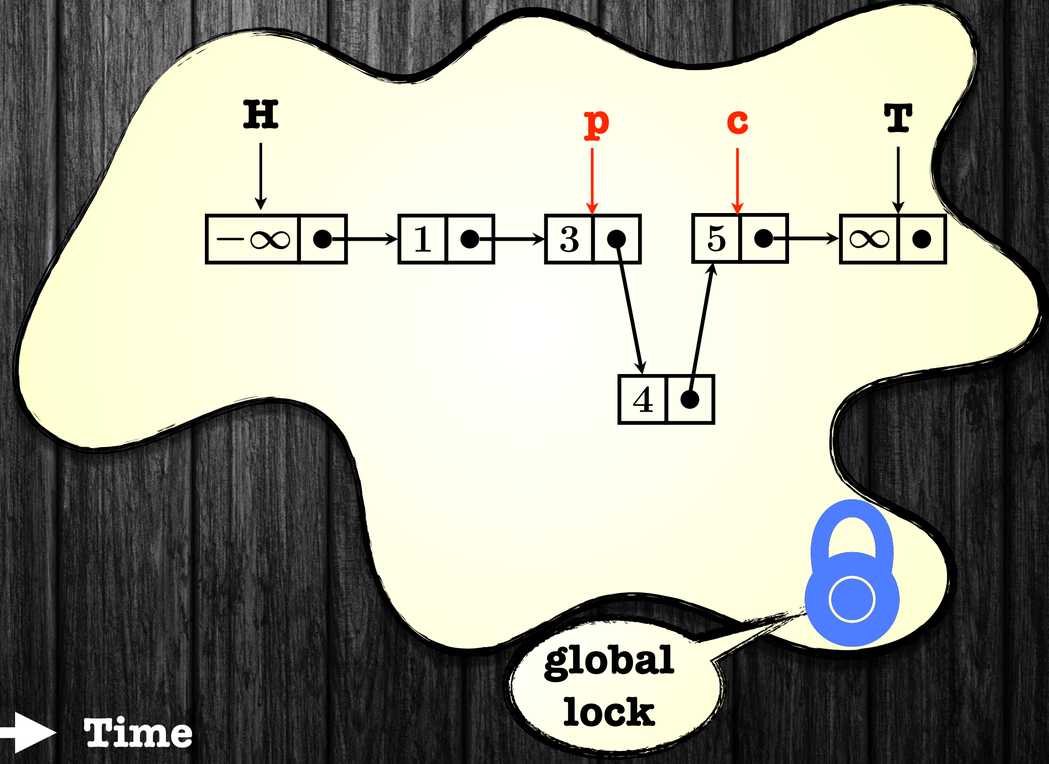
# Sets Coarse-Grained Synchronization

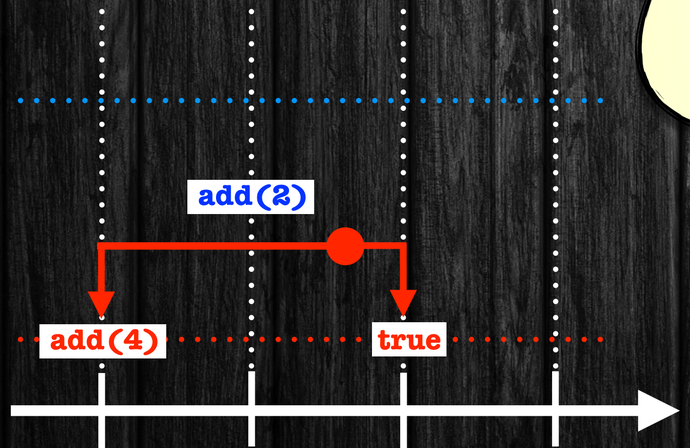
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

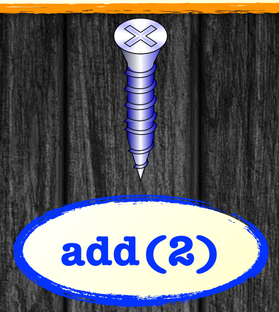


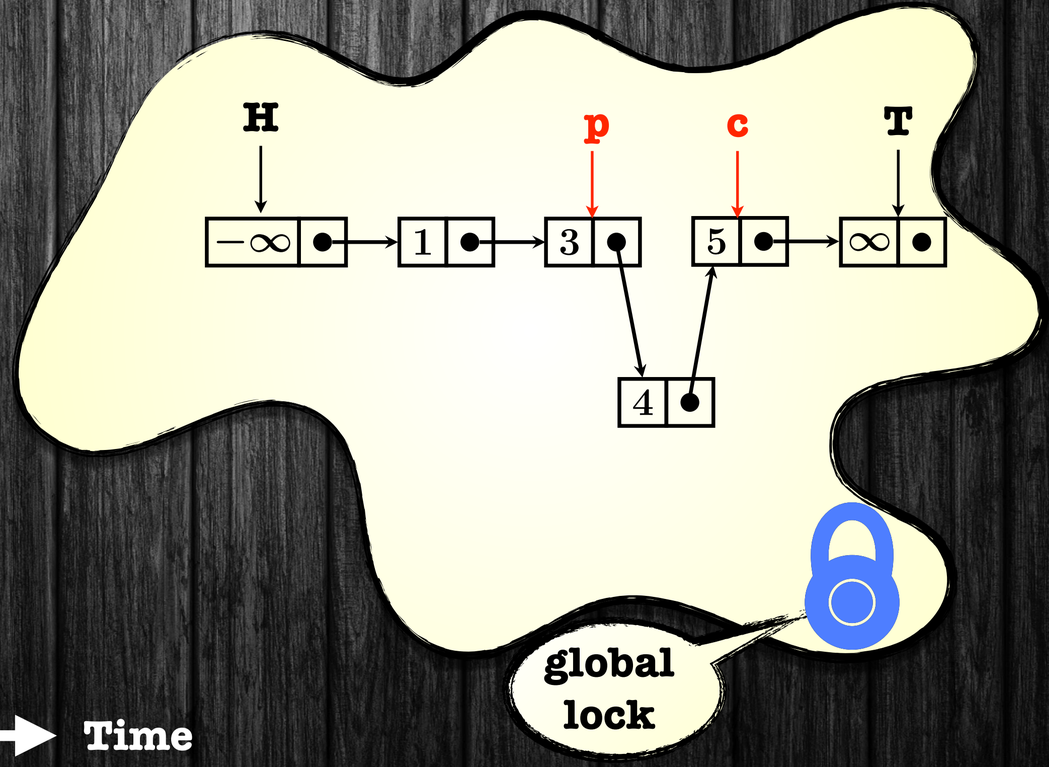
# Sets Coarse-Grained Synchronization

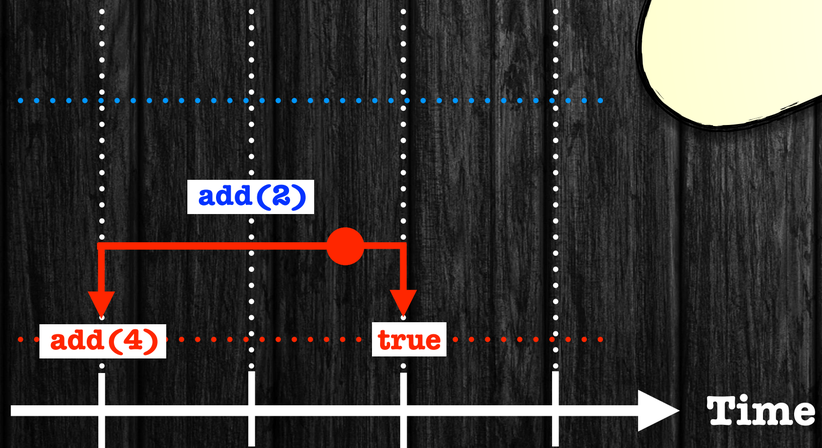
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```




```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



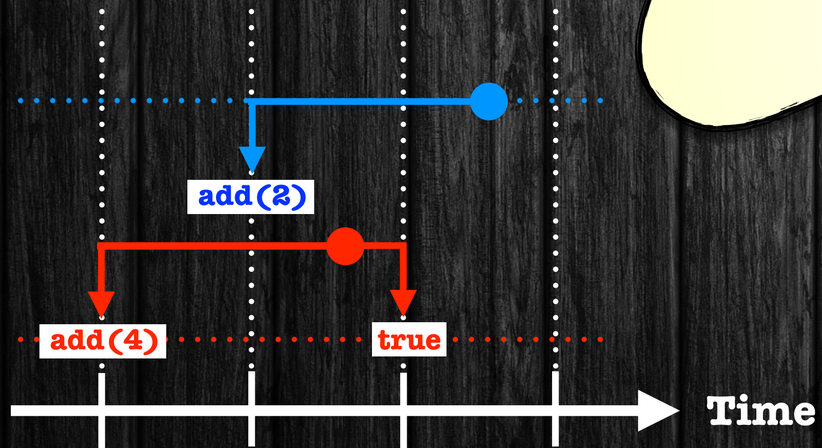
# Sets Coarse-Grained Synchronization

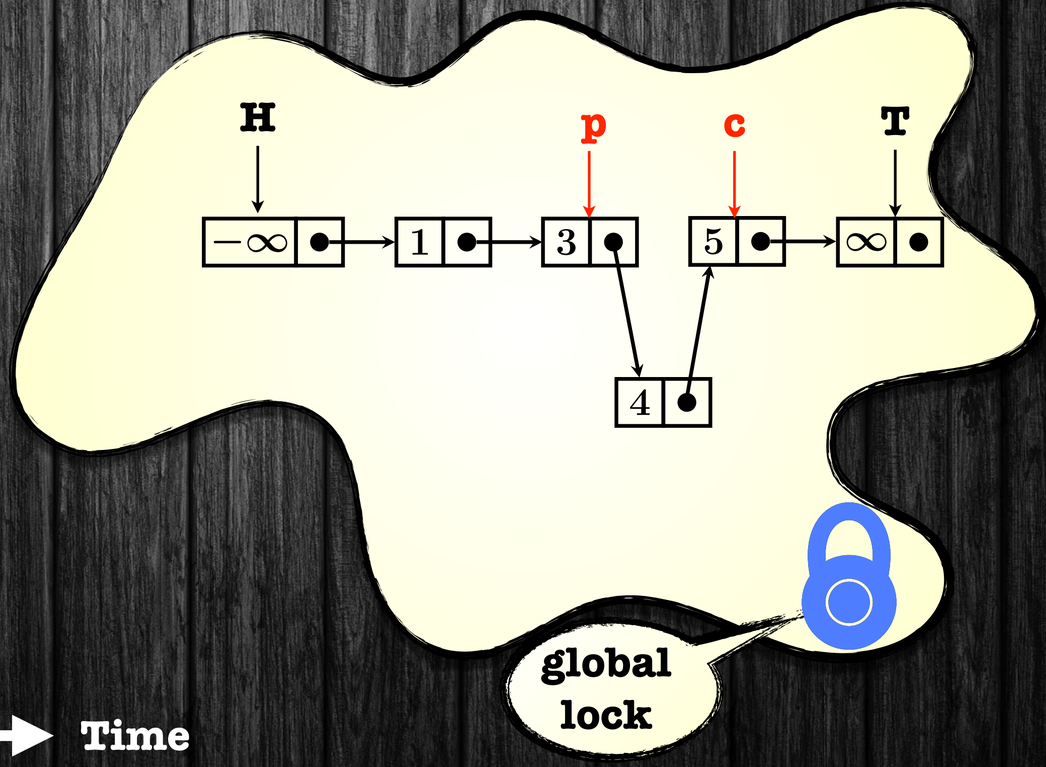
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



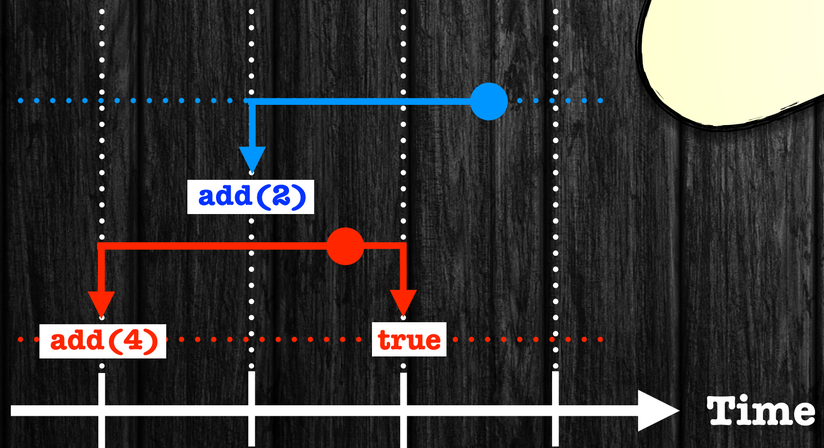
# Sets Coarse-Grained Synchronization

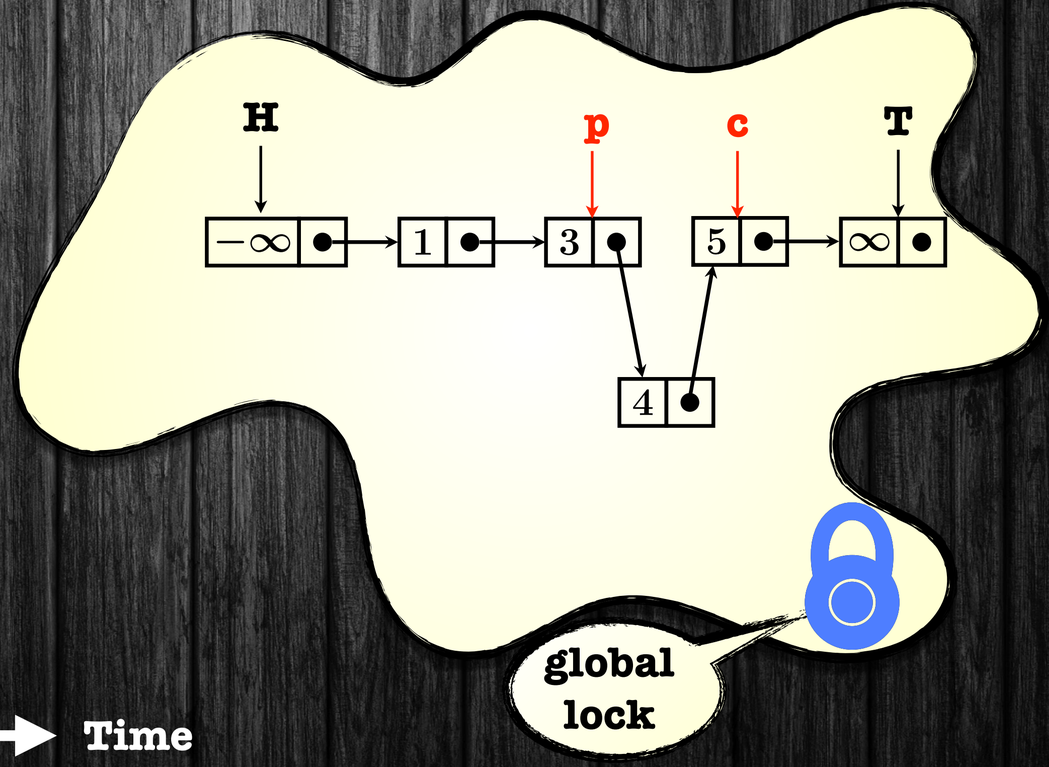
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



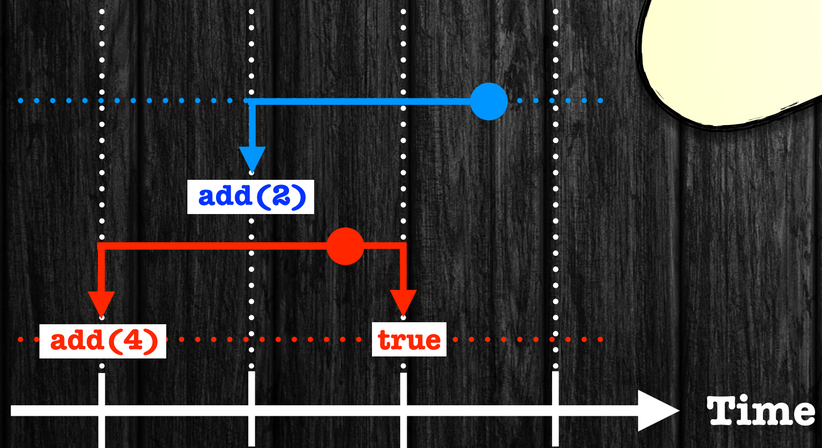
# Sets Coarse-Grained Synchronization

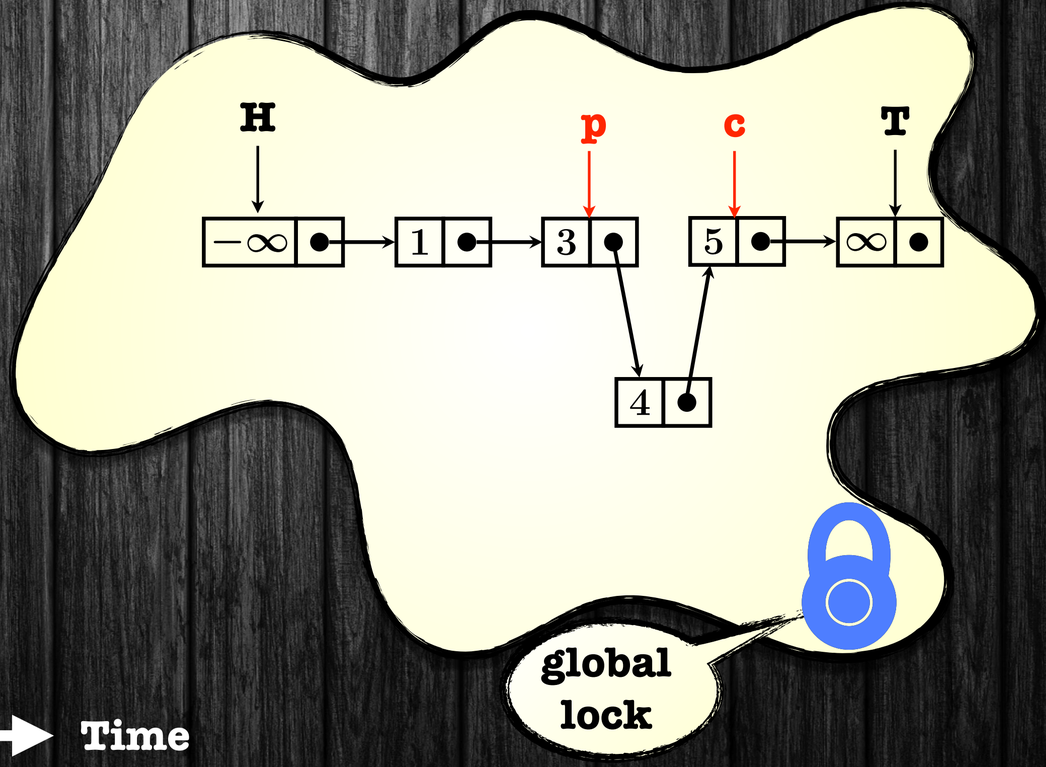
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



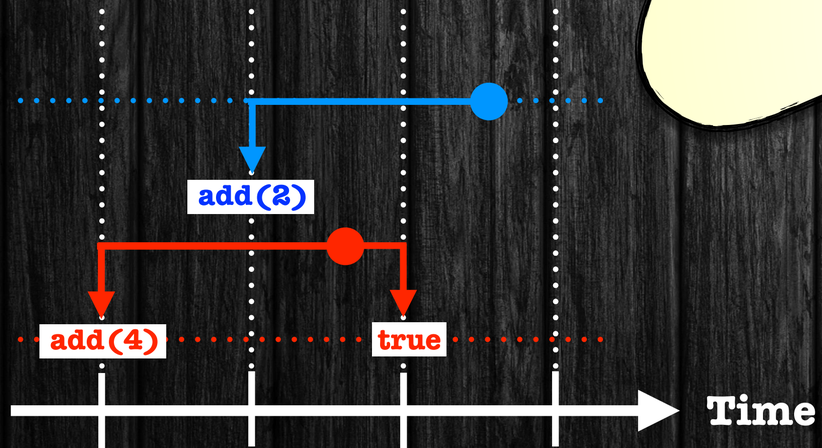
# Sets Coarse-Grained Synchronization

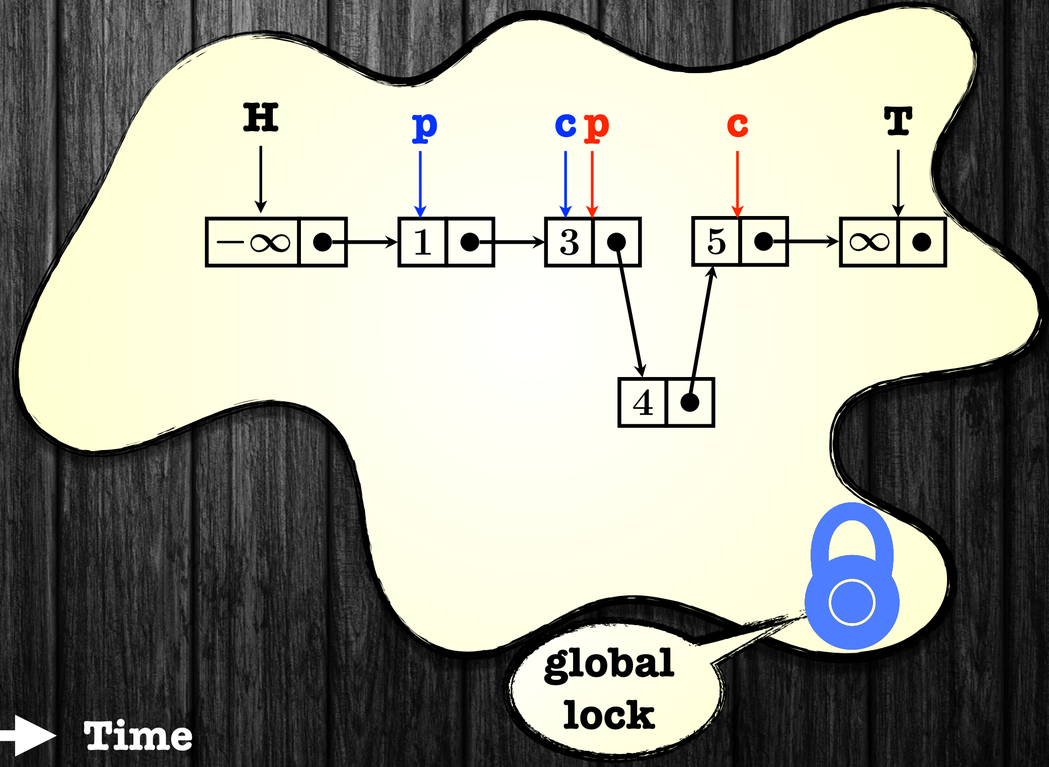
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



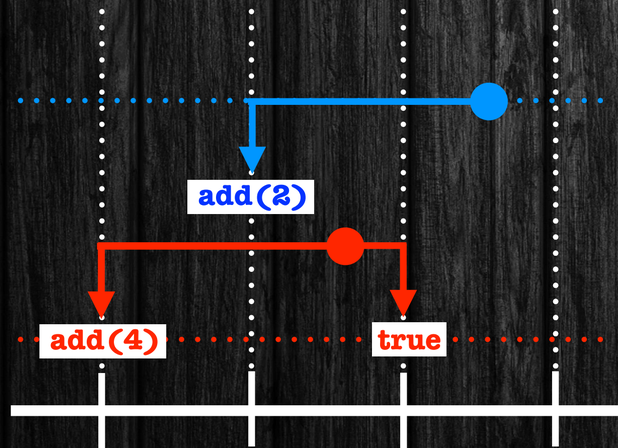
# Sets Coarse-Grained Synchronization

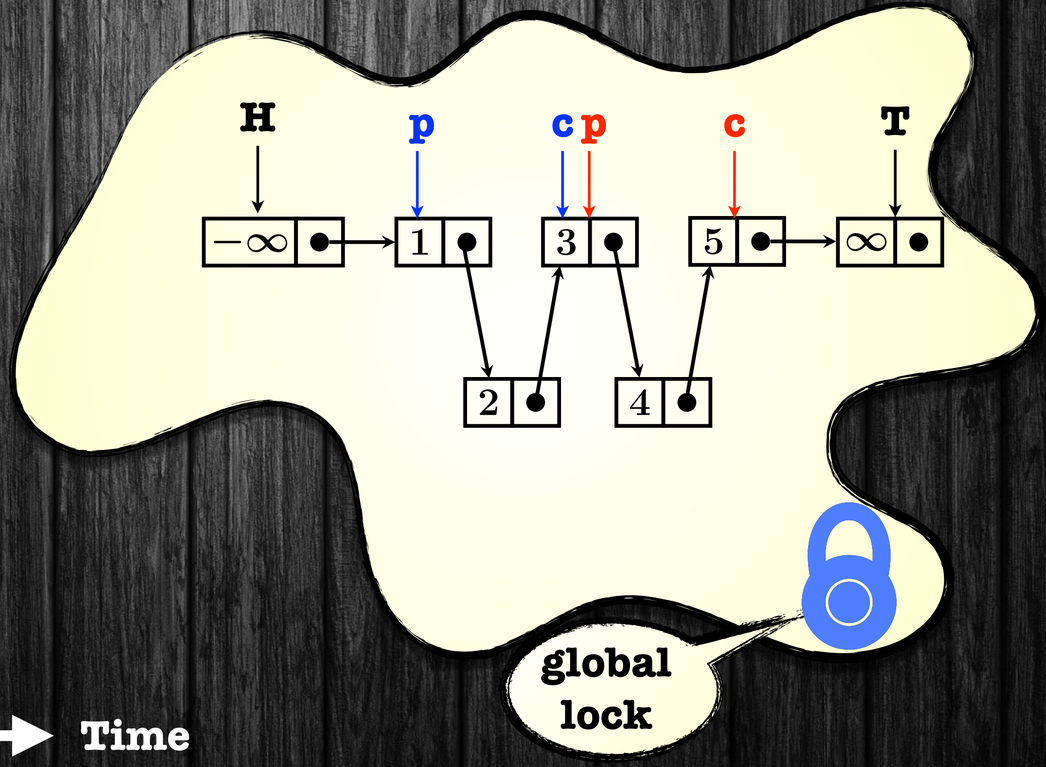
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



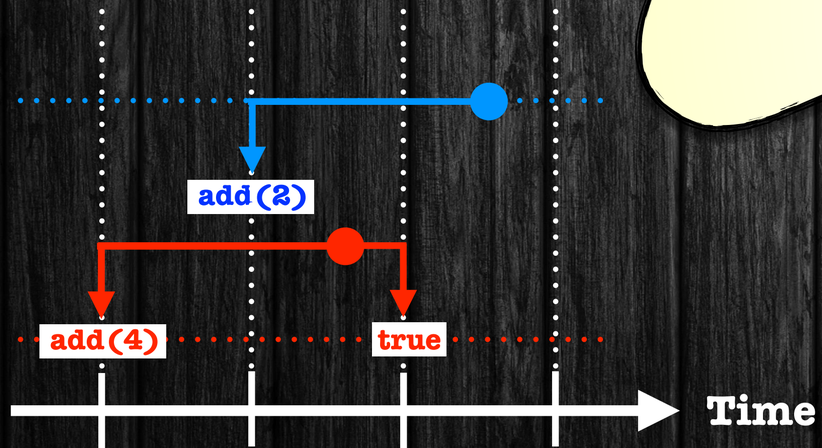
# Sets Coarse-Grained Synchronization

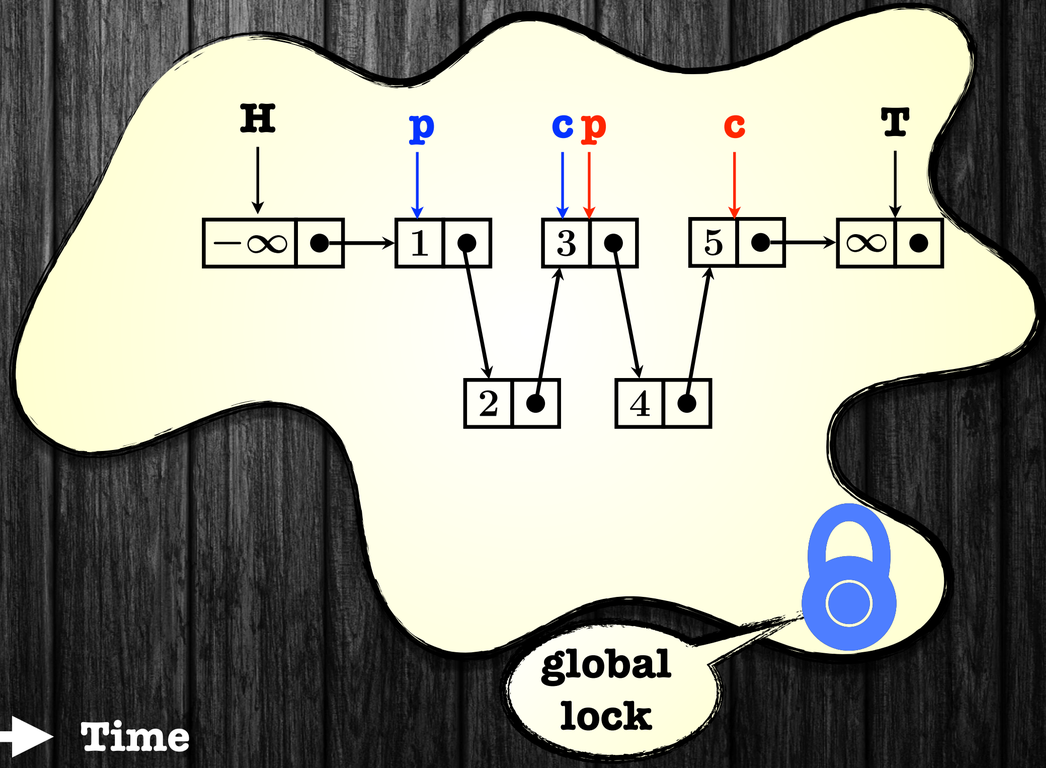
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



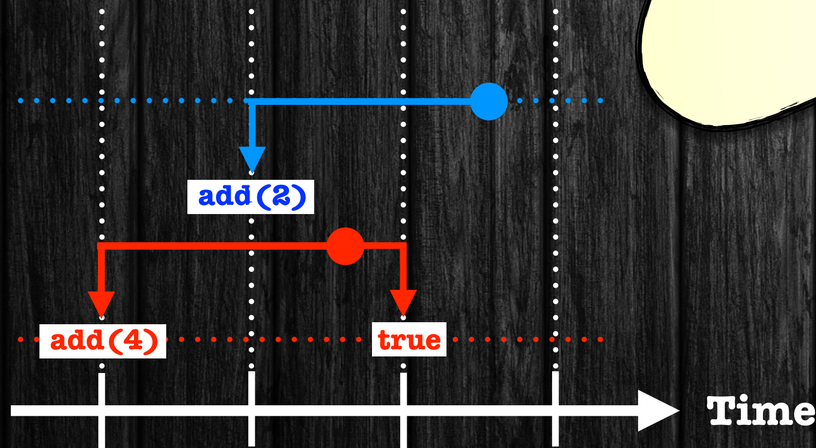
# Sets Coarse-Grained Synchronization

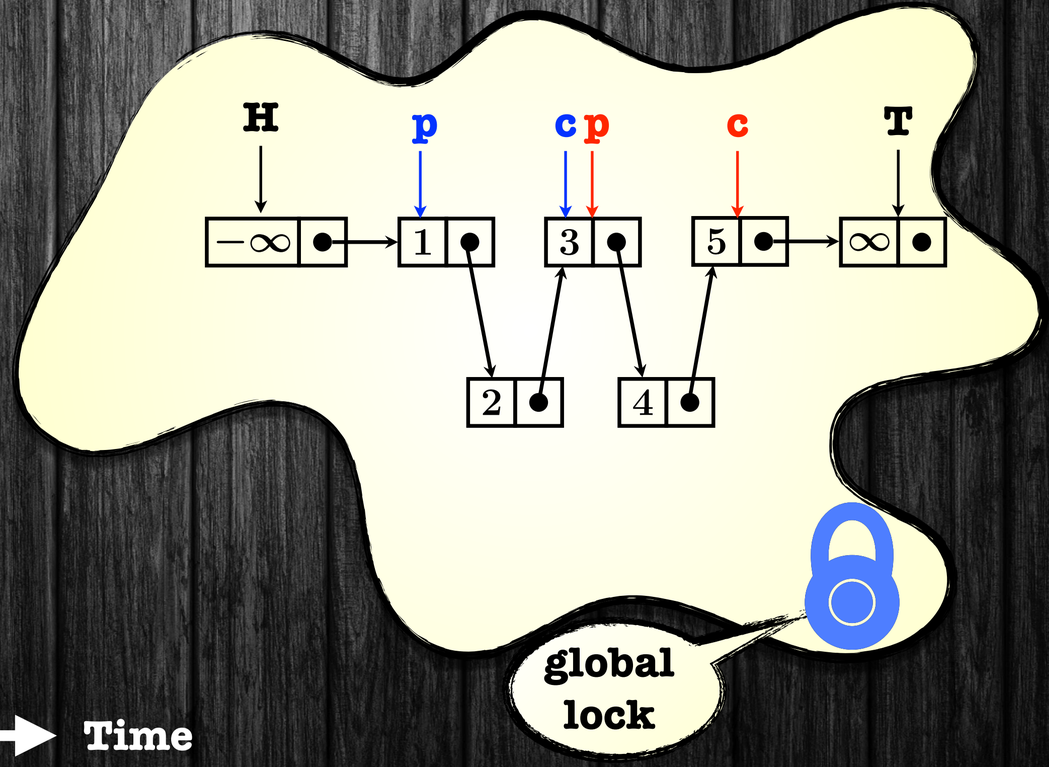
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



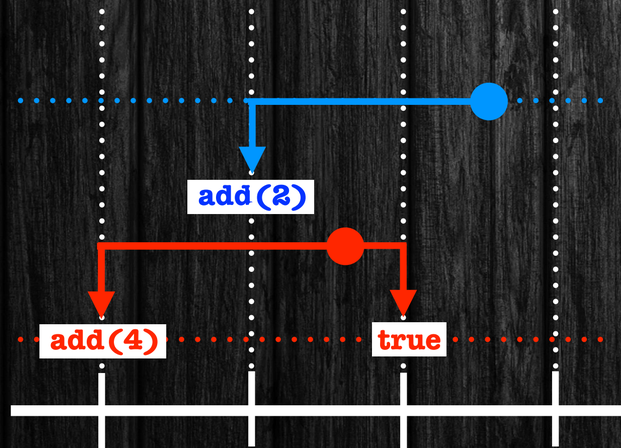
# Sets Coarse-Grained Synchronization

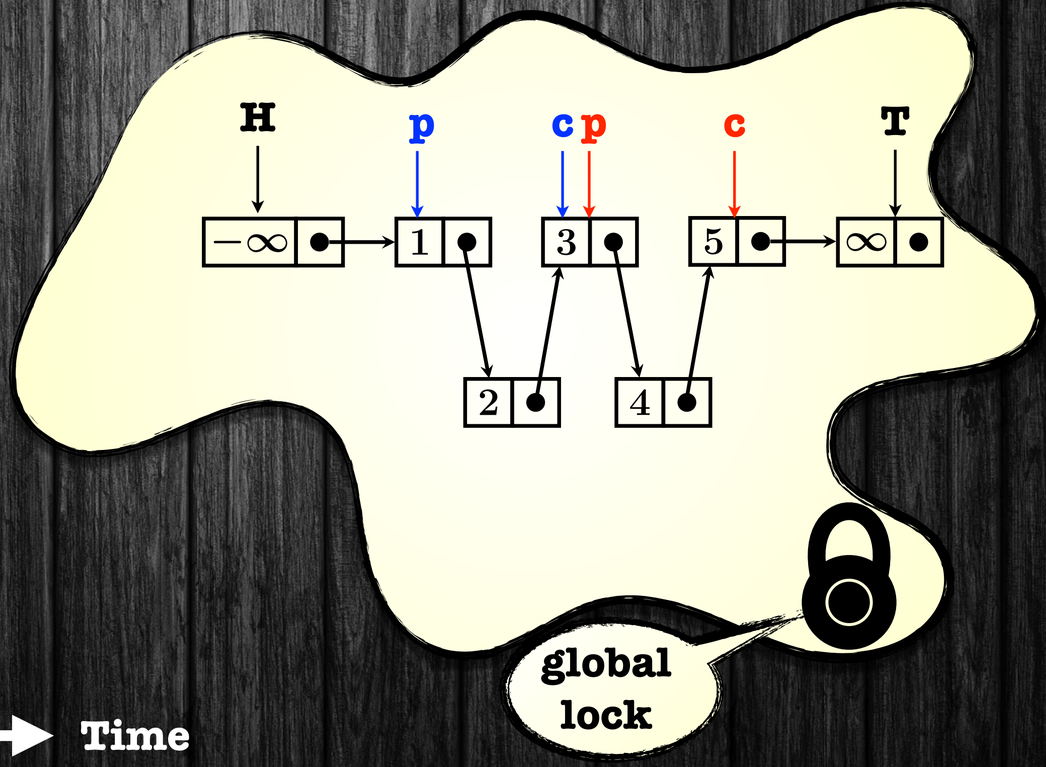
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



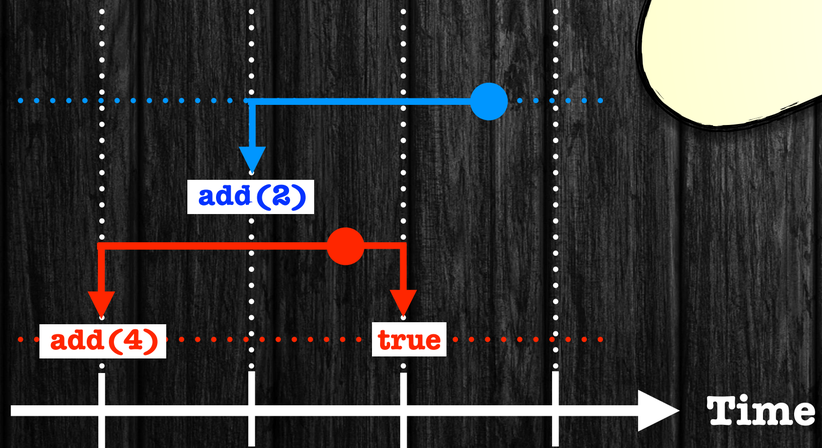
# Sets Coarse-Grained Synchronization

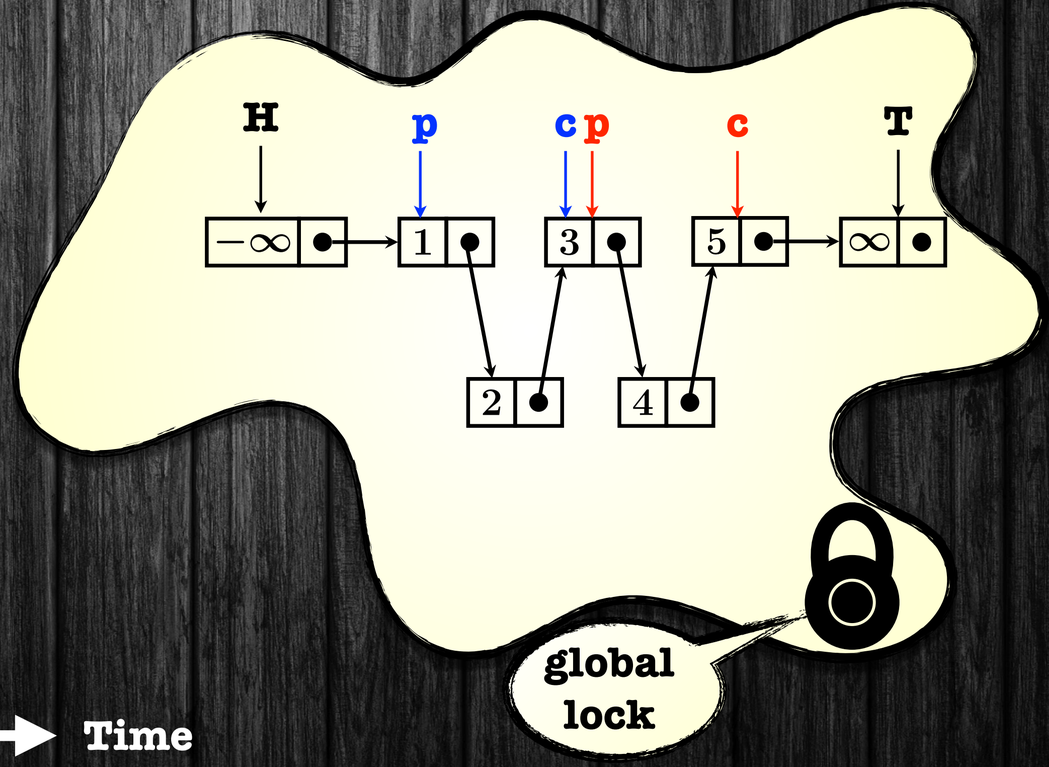
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

add(2)

```
add(k):
•••
• lock l; LP
• •••
• unlock l
```



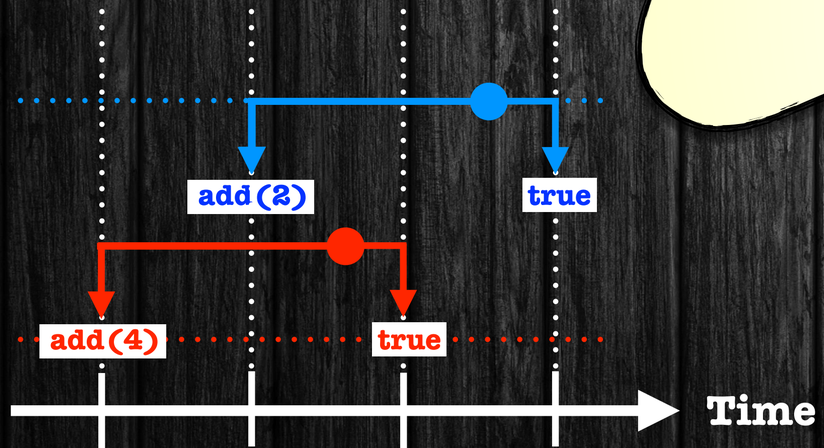
# Sets Coarse-Grained Synchronization

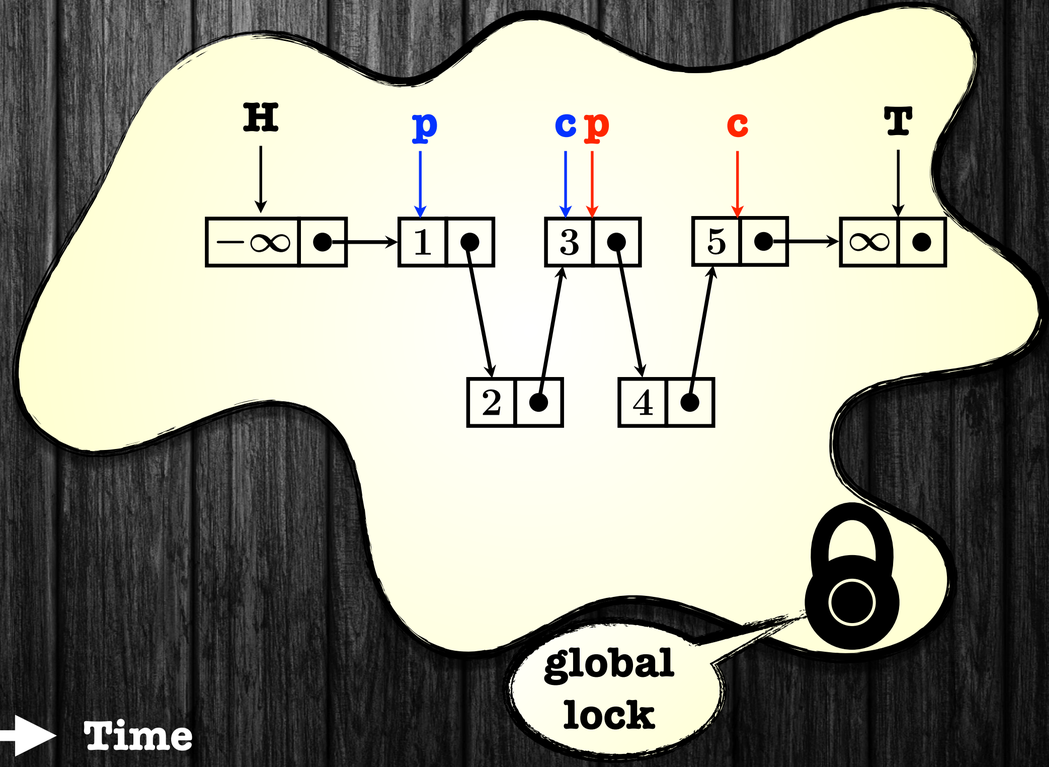
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

add(4)

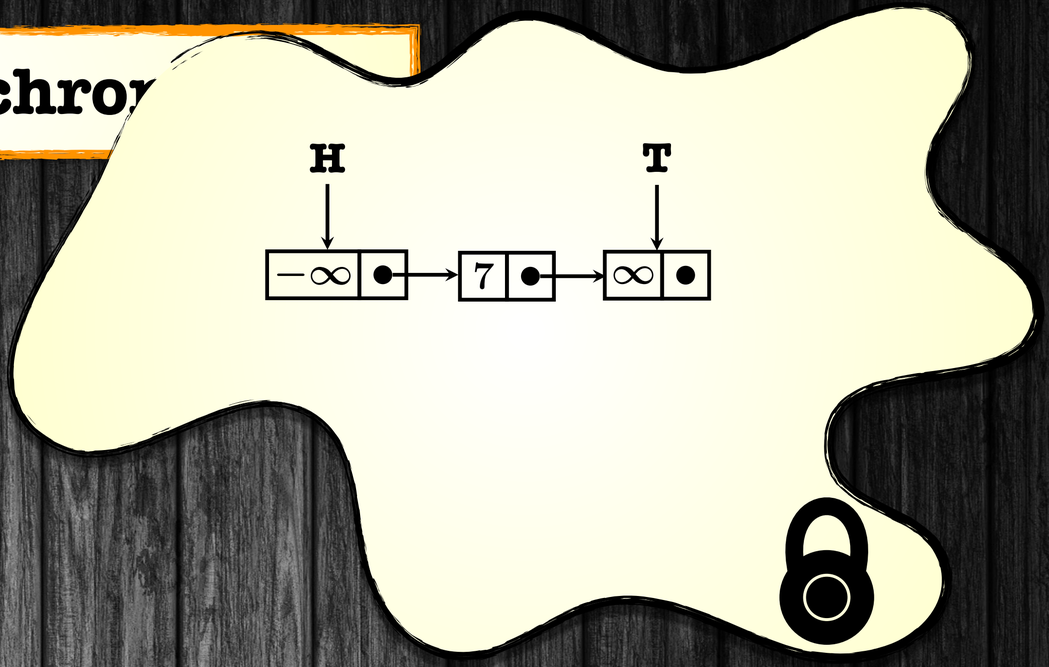
add(2)

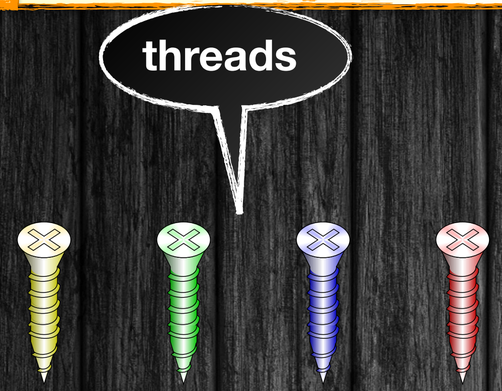
```
add(k):
•••
• lock l; LP
• •••
• unlock l
```

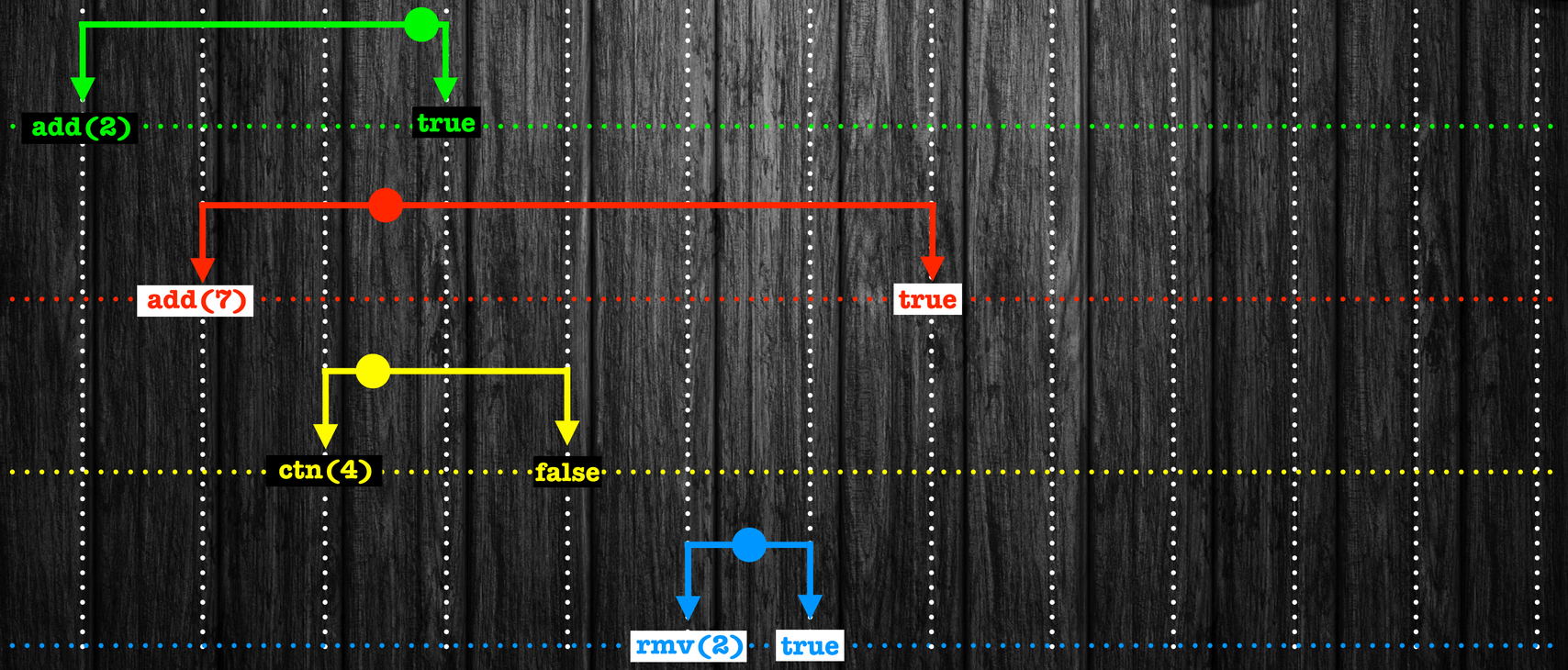


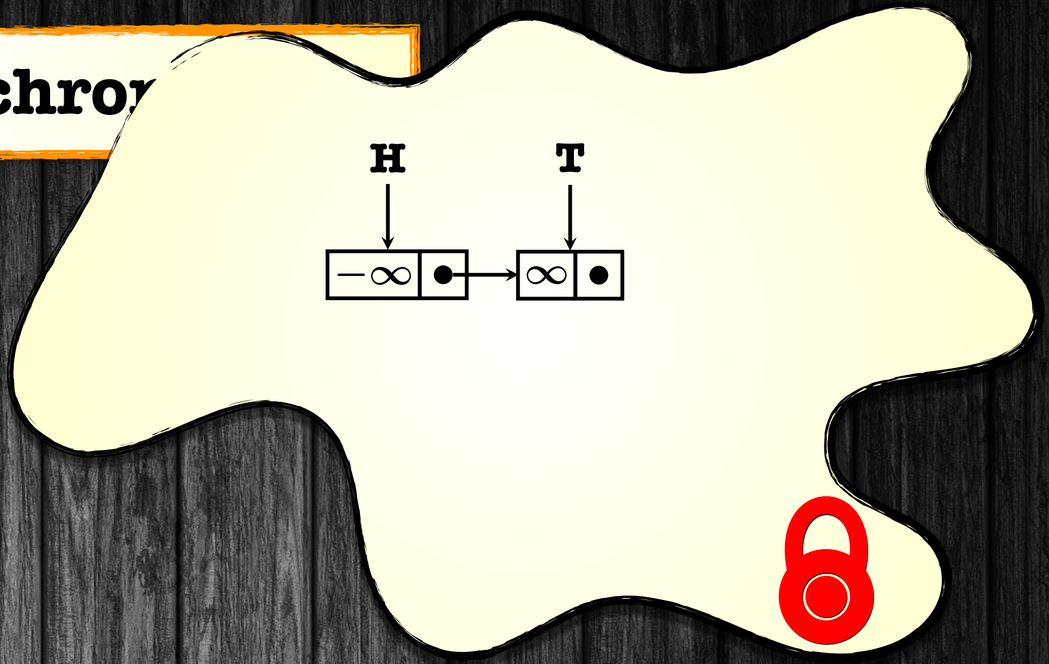
# Sets Coarse-Grained Synchron

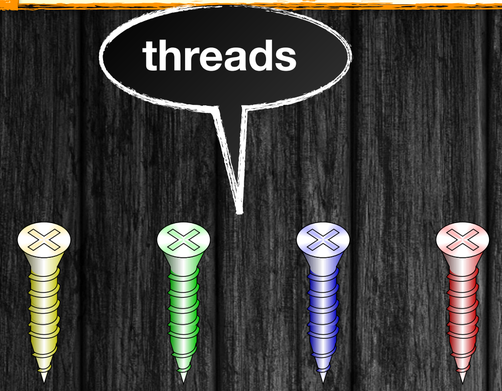


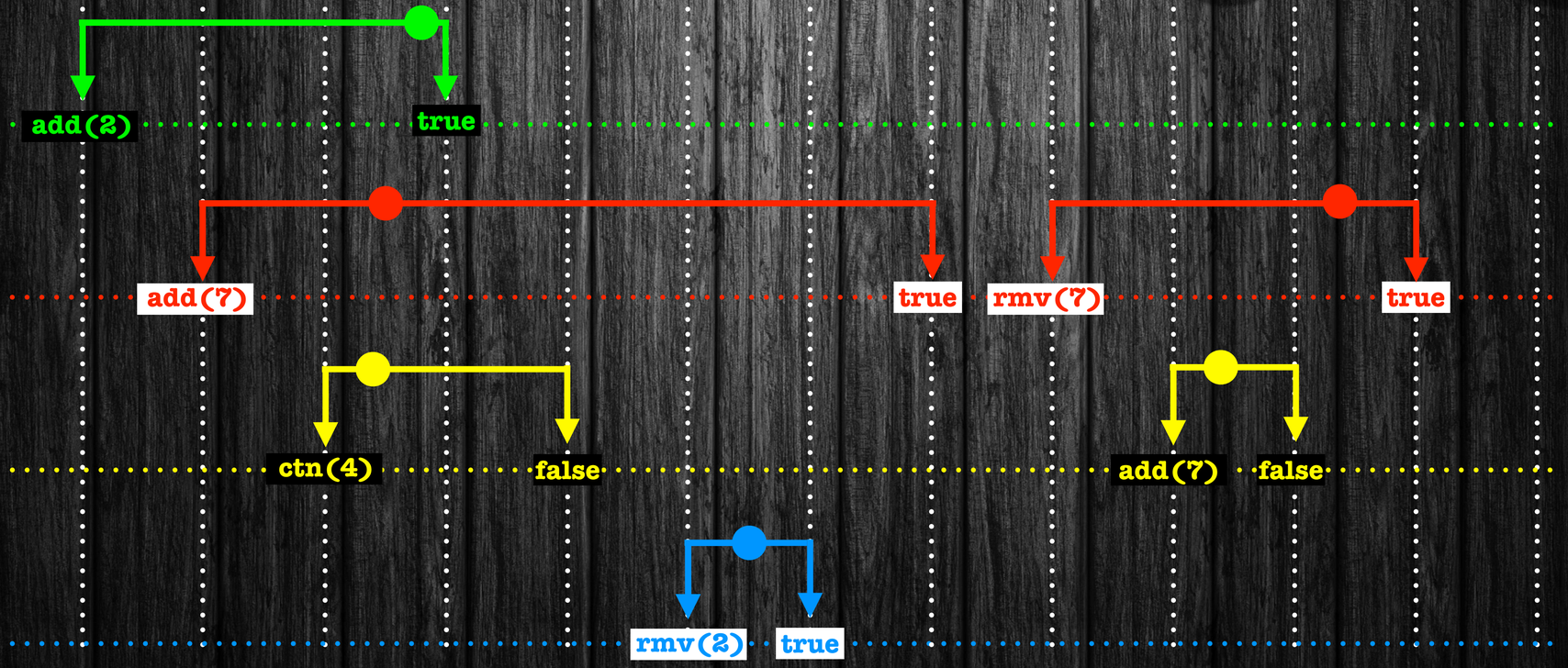



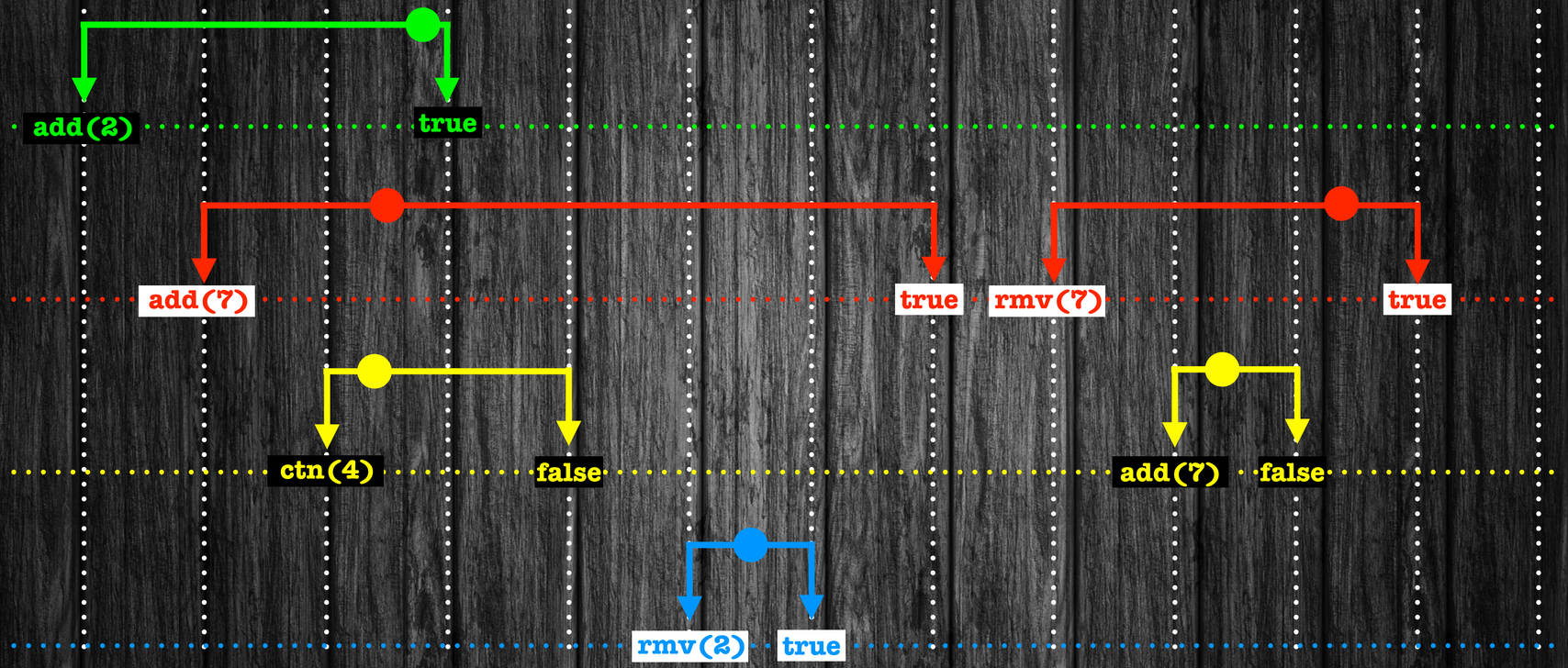
# Sets Coarse-Grained Synchron



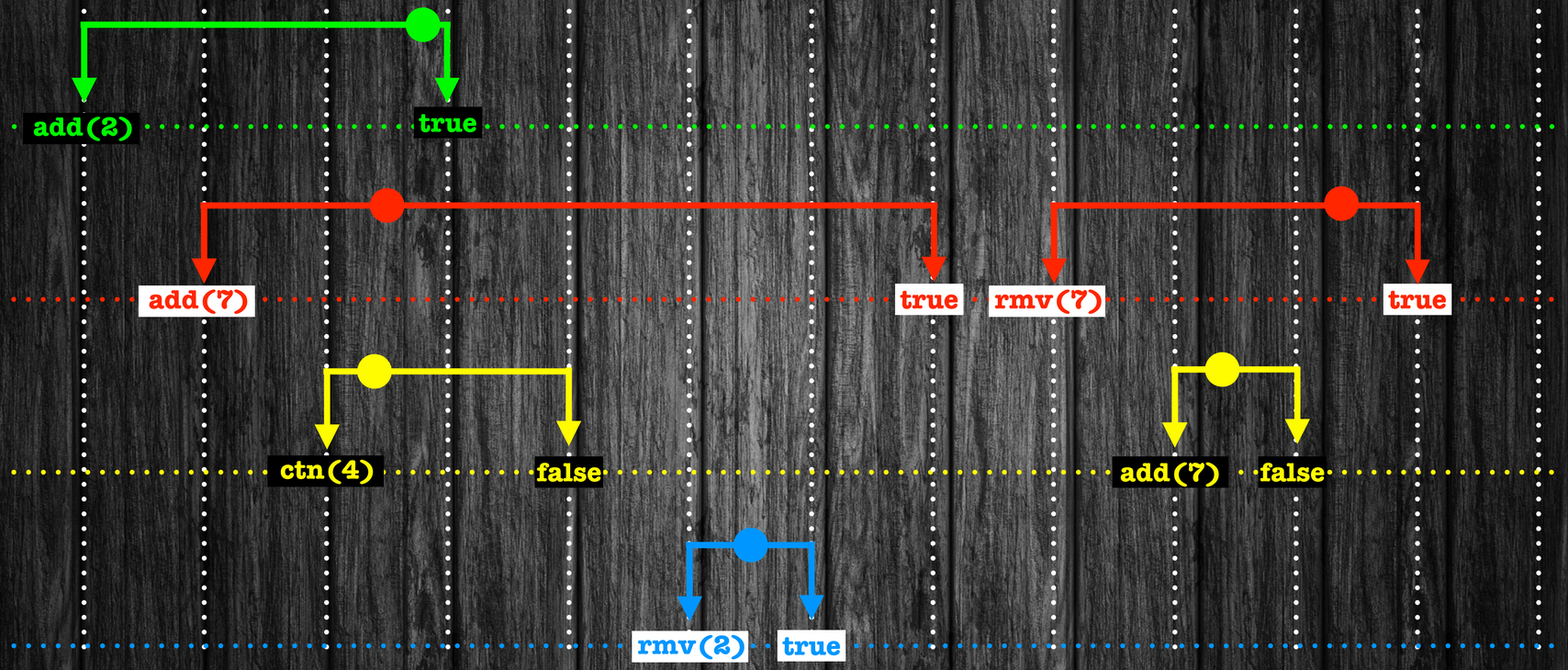



# Sets Coarse-Grained Synchronization

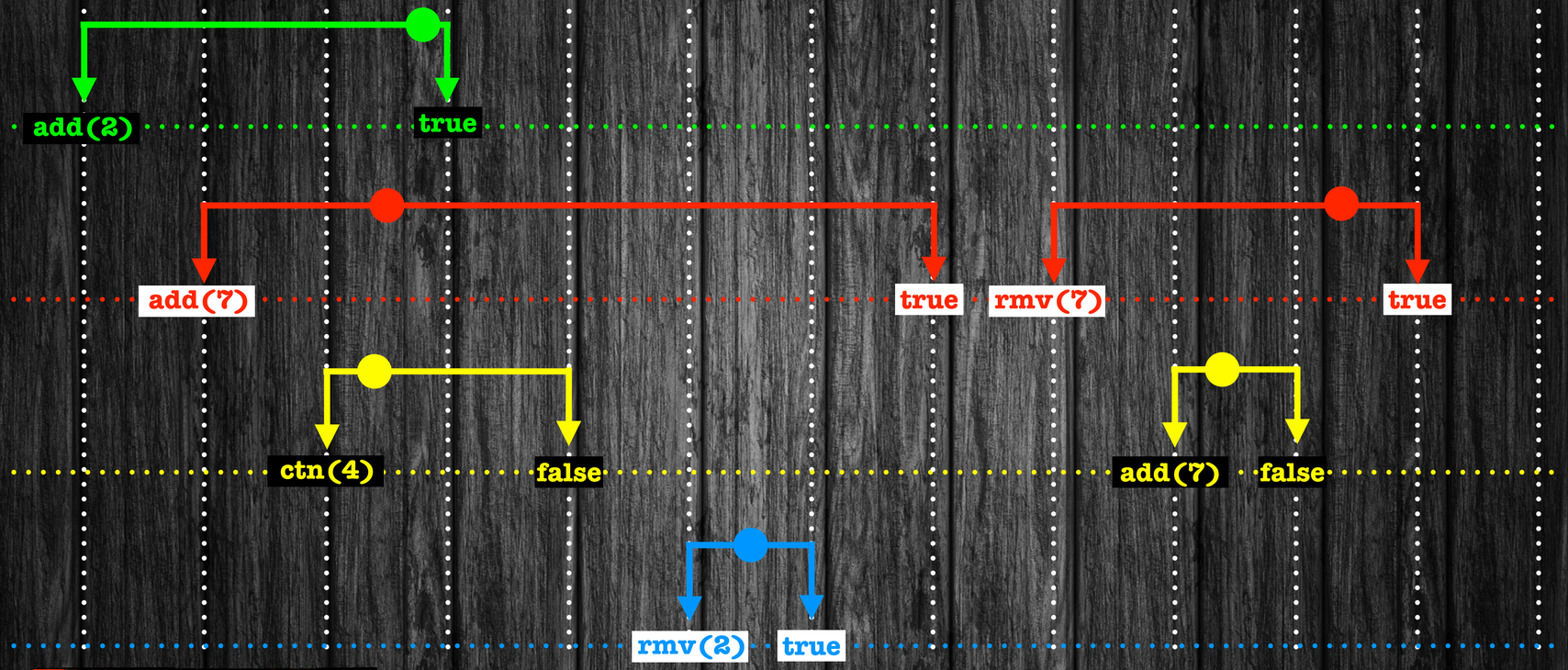


# Sets Coarse-Grained Synchronization




✔

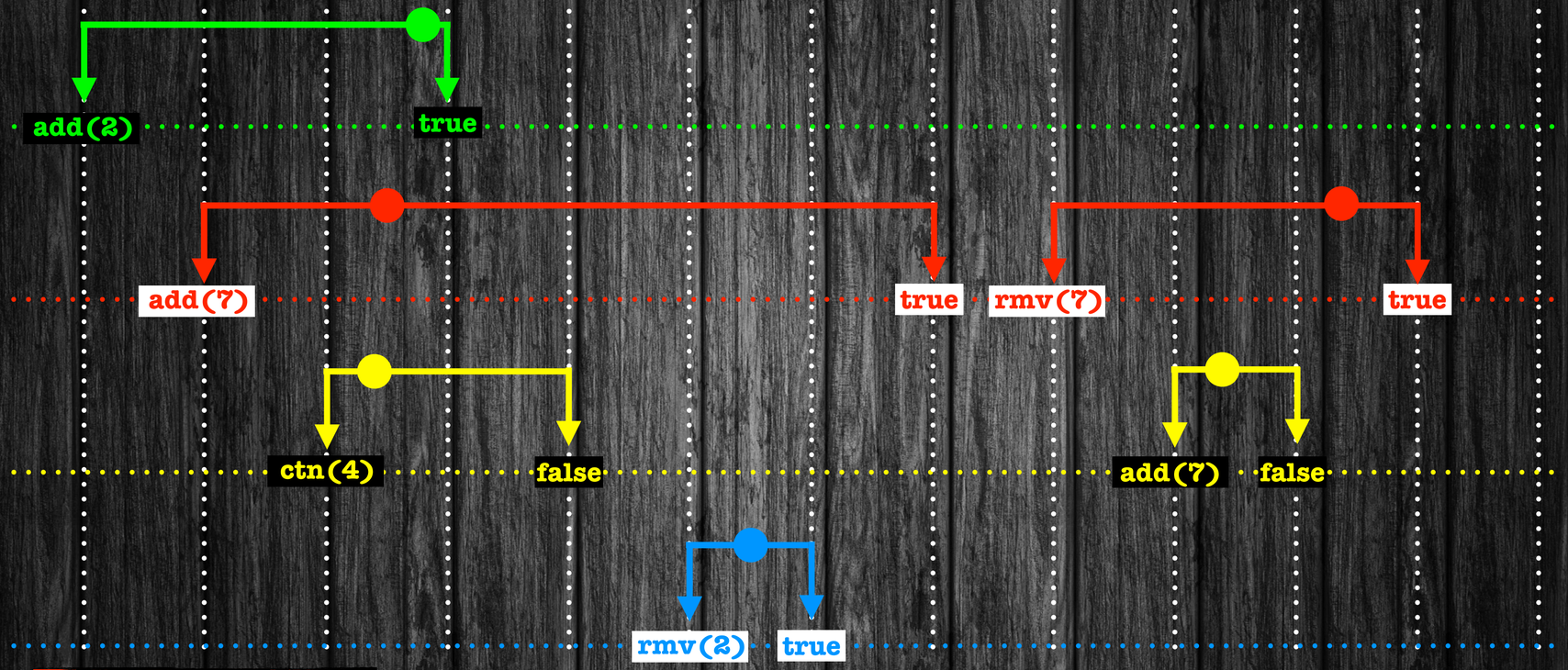
# Sets Coarse-Grained Synchronization




**Linearizable**

✔

# Sets | Coarse-Grained Synchronization




**Linearizable**

**Program linearizable wrt. Set**

✔

# Sets

## Coarse-Grained Synchronization

```
 add(k):
 Node p, c
 Lock l
 1  lock l;
 2  p = H;
 3  c = p.next
 4  while (c.key < k)
 5    p = c;
 6    c = c.next;
 7  if (c.key = k)
 8    return false
 9  else
10    n = new Node(k,-)
11    n.next = c
12    p.next = n
13    return true
14  unlock l
```

LP

# Sets

## Coarse-Grained Synchronization

```
 add(k):
 Node p, c
 Lock l
 1  lock l;
 2  p = H;
 3  c = p.next
 4  while (c.key < k)
 5    p = c;
 6    c = c.next;
 7  if (c.key = k)
 8    return false
 9  else
10    n = new Node(k,-)
11    n.next = c
12    p.next = n
13    return true
14  unlock l
```

LP ?

# Sets Coarse-Grained Synchronization

```
 add(k):
 Node p, c
 Lock l
 1  lock l;
 2  p = H;
 3  c = p.next
 4  while (c.key < k)
 5    p = c;
 6    c = c.next;
 7  if (c.key = k)
 8    return false
 9  else
10    n = new Node(k,-)
11    n.next = c
12    p.next = n
13    return true
14  unlock l
```

LP ?

✔

# Sets Coarse-Grained Synchronization

```
 add(k):
 Node p, c
 Lock l
 1  lock l;
 2  p = H;
 3  c = p.next
 4  while (c.key < k)
 5    p = c;
 6    c = c.next;
 7  if (c.key = k)
 8    return false
 9  else
10    n = new Node(k,-)
11    n.next = c
12    p.next = n
13    return true
14  unlock l
```

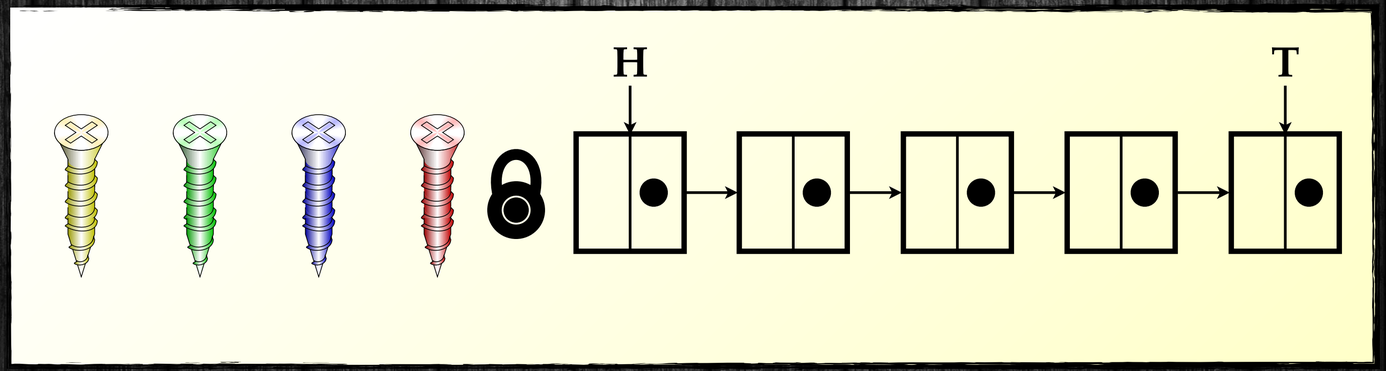
LP ? ✔

**“linearization point anywhere between lock and unlock**

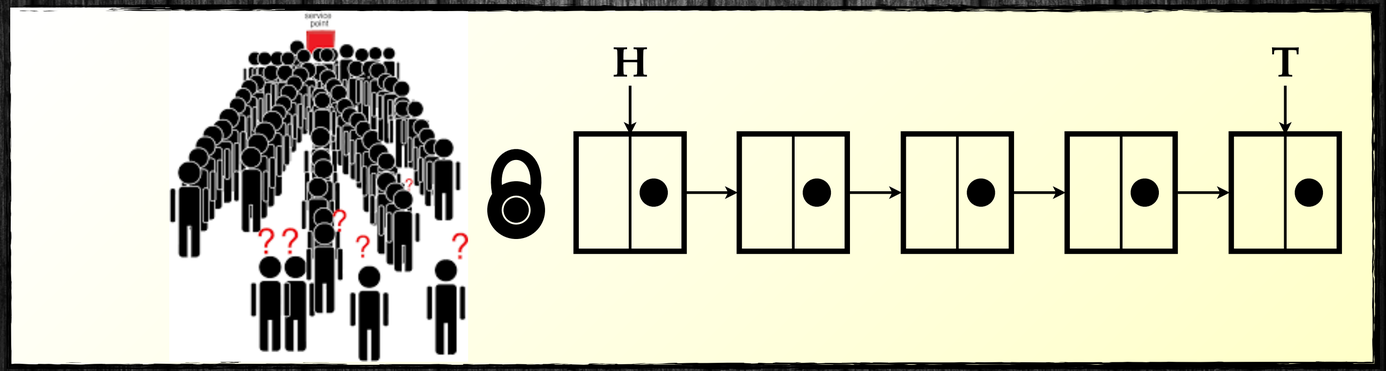
# Sets | Coarse-Grained Synchronization

# Sets | Coarse-Grained Synchronization

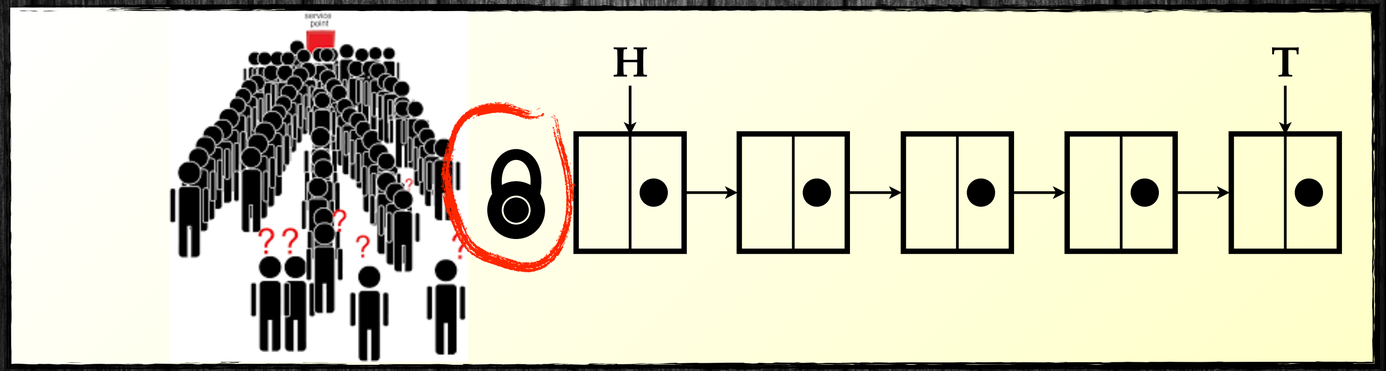
# Sets Coarse-Grained Synchronization



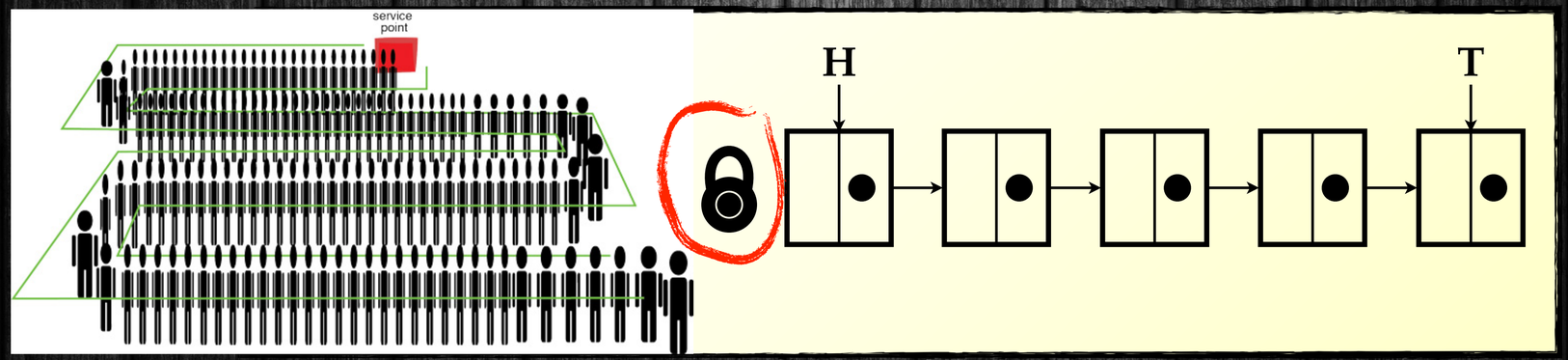
# Sets Coarse-Grained Synchronization



# Sets Coarse-Grained Synchronization



# Sets Coarse-Grained Synchronization



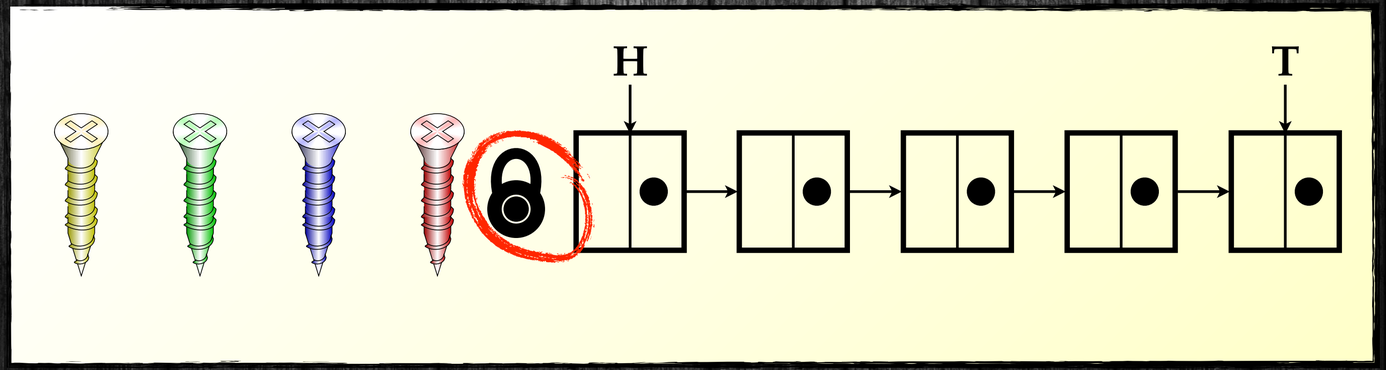
# Sets Coarse-Grained Synchronization

advantages

- simple — "essentially sequential"
- efficient — "if low contention"
- correct

disadvantages

- inefficient — "if high contention"



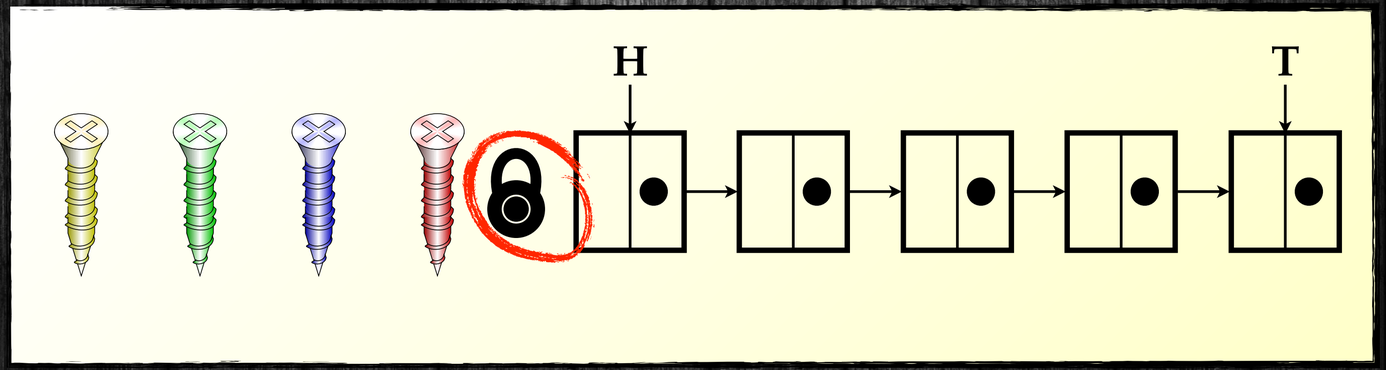
# Sets Coarse-Grained Synchronization

advantages

- simple — "essentially sequential"
- efficient — "if low contention"
- correct

disadvantages

- inefficient — "if high contention"



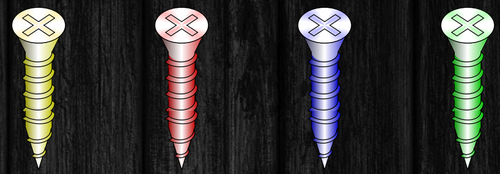
# Outline

- Overview
- Abstract Data
- Sequential
- Concurrent
- Coarse-Grained Locking
- Fine-Grained Locking
- Lazy Lists
- Optimistic  Lists
- Treiber Stack  Algorithm
- Michael&Scott Queue

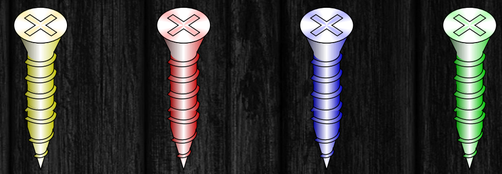
# Outline

- Overview
- Abstract Data
- Sequential
- Concurrent
- ~~Coarse-Grained Locking~~
- Fine-Grained Locking
- Lazy Lists
- Optimistic  Lists
- Treiber Stack  Algorithm
- Michael&Scott Queue

# Sets Fine-Grained Synchronization



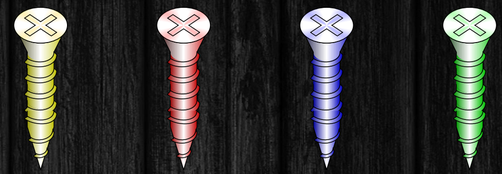
# Sets Fine-Grained Synchronization



Goal

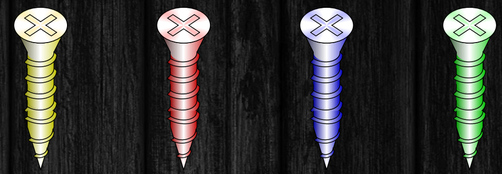
more concurrency

# Sets | Fine-Grained Synchronization



- **Goal** — more concurrency
- **How?** — do not lock whole data structure

# Sets Fine-Grained Synchronization

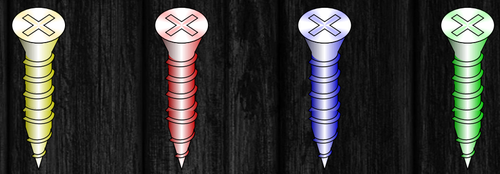


Goal: more concurrency

How?

- do not lock whole data structure
- lock individual cells

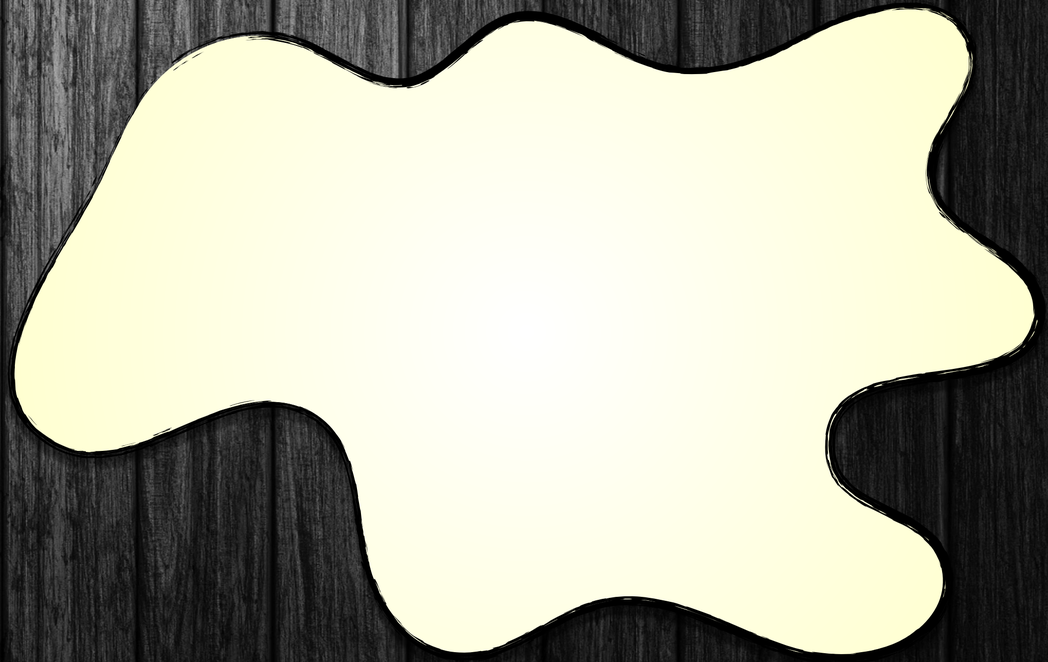
# Sets Fine-Grained Synchronization



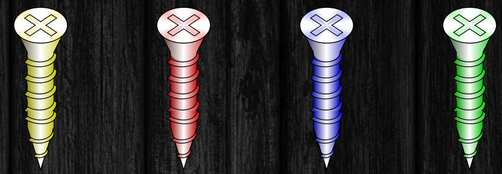
Goal: more concurrency

How?: do not lock whole data structure

lock individual cells



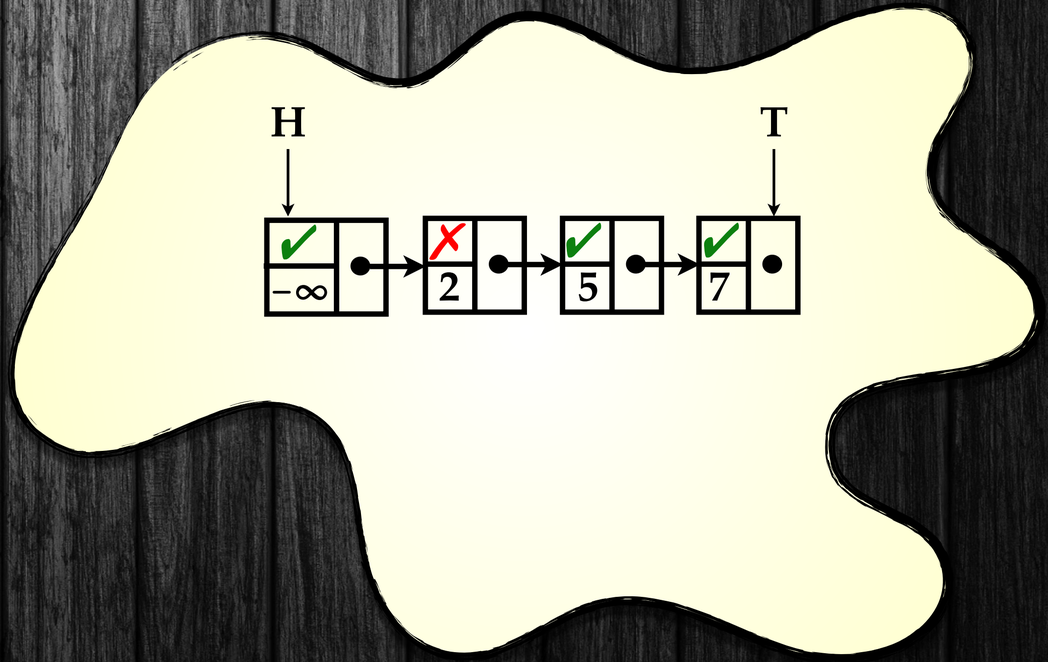
# Sets Fine-Grained Synchronization



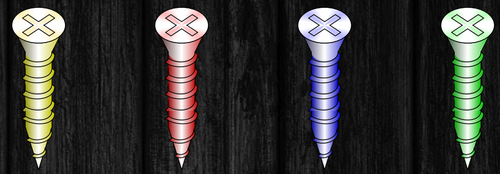
Goal — more concurrency

How? — do not lock whole data structure

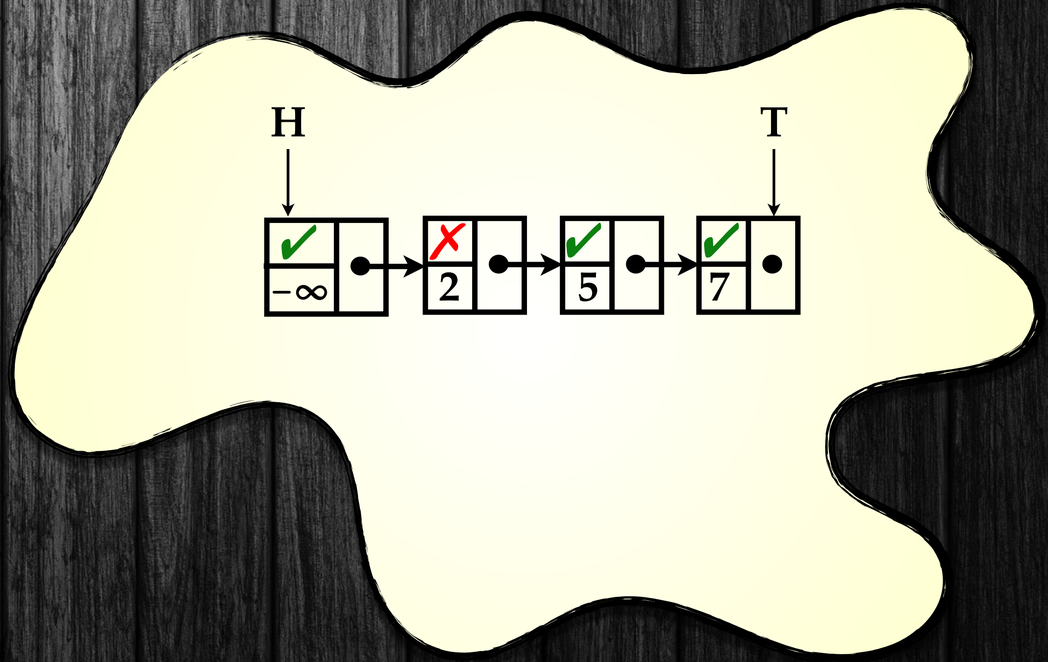
lock individual cells



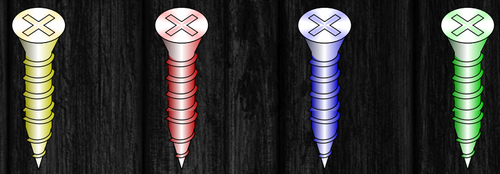
# Sets Fine-Grained Synchronization



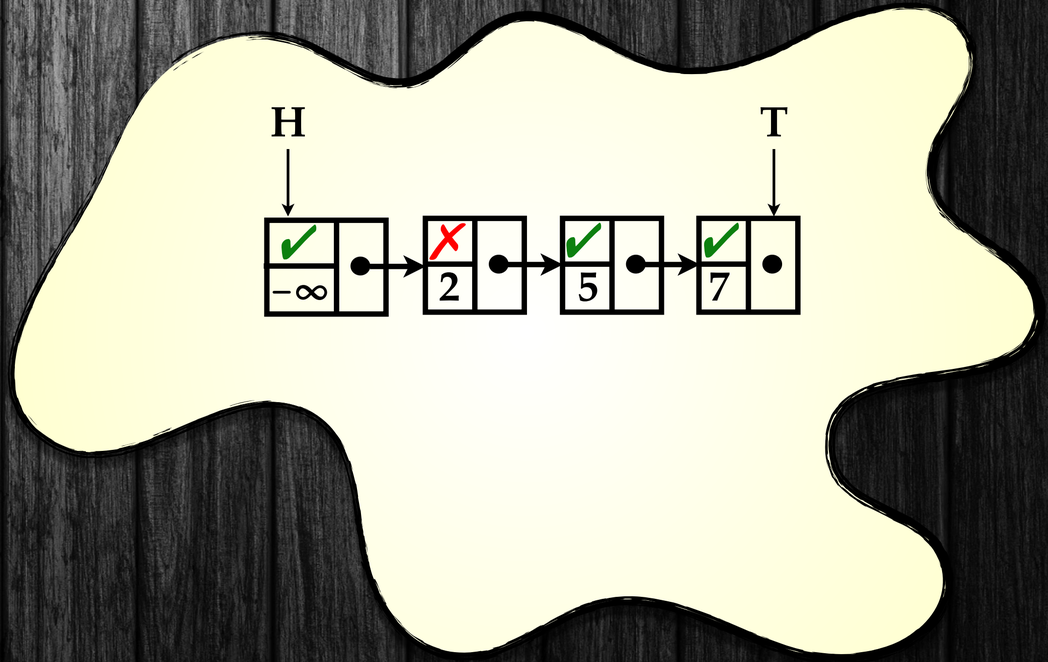
- **Goal** → more concurrency
- **How?** → do not lock whole data structure
  - lock individual cells
- **Strategy** → “Lock Coupling”



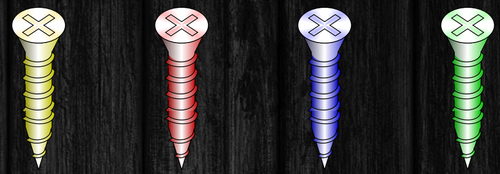
# Sets Fine-Grained Synchronization



- **Goal** — more concurrency
- **How?** — do not lock whole data structure; lock individual cells
- **Strategy** — “Lock Coupling”; “hand-over-hand lock acquirement”



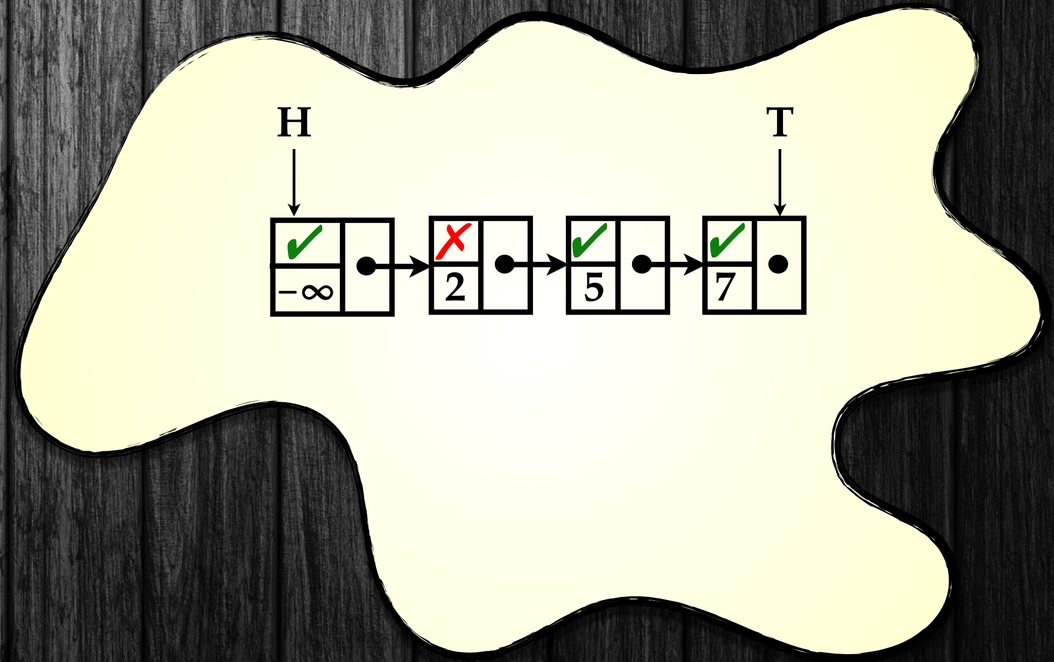
# Sets Fine-Grained Synchronization



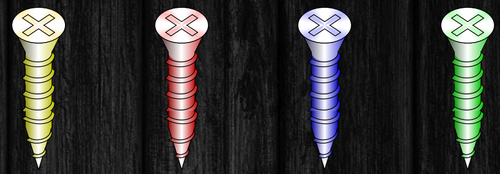
**Goal** — more concurrency

**How?** — do not lock whole data structure | lock individual cells

**Strategy** — “Lock Coupling” | “hand-over-hand lock acquirement” | “acquire lock for next cell before releasing the lock for the current cell”



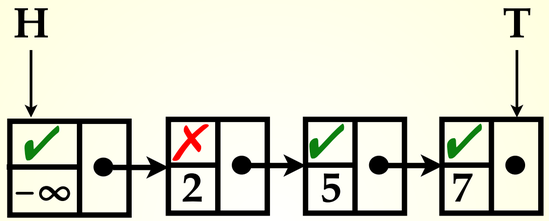
# Sets Fine-Grained Synchronization



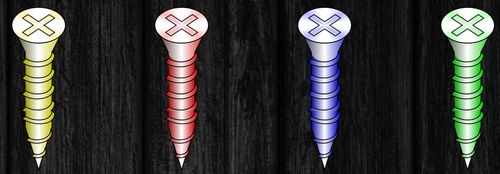
**Goal** — more concurrency

**How?** — do not lock whole data structure | lock individual cells

**Strategy** — “Lock Coupling” | “hand-over-hand lock acquirement” | “acquire lock for next cell before releasing the lock for the current cell”



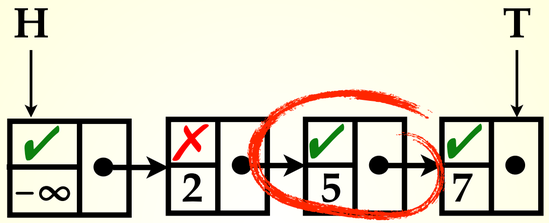
# Sets Fine-Grained Synchronization



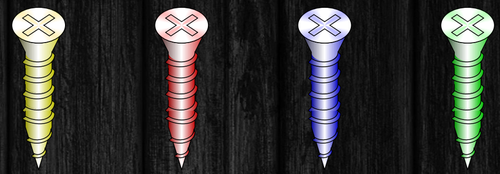
**Goal** — more concurrency

**How?** — do not lock whole data structure | lock individual cells

**Strategy** — “Lock Coupling” | “hand-over-hand lock acquirement” | “acquire lock for next cell before releasing the lock for the current cell”



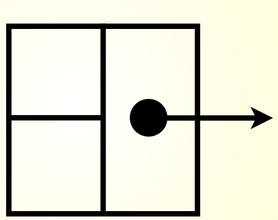
# Sets Fine-Grained Synchronization

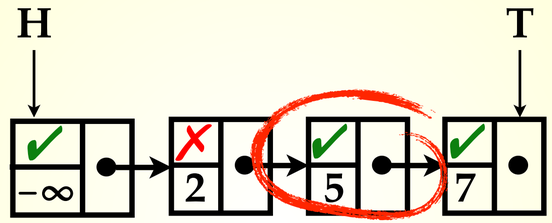


**Goal** — more concurrency

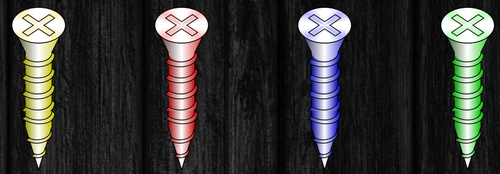
**How?** — do not lock whole data structure | lock individual cells

**Strategy** — “Lock Coupling” | “hand-over-hand lock acquirement” | “acquire lock for next cell before releasing the lock for the current cell”





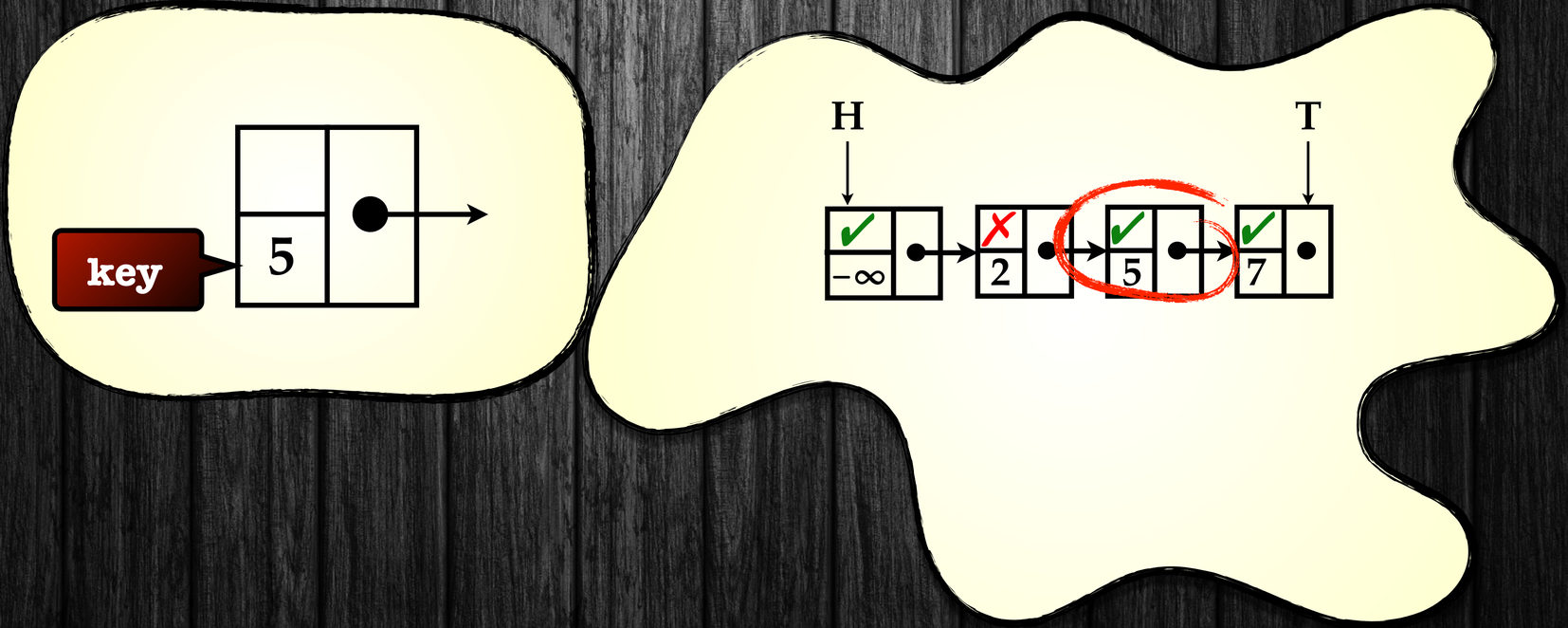
# Sets Fine-Grained Synchronization



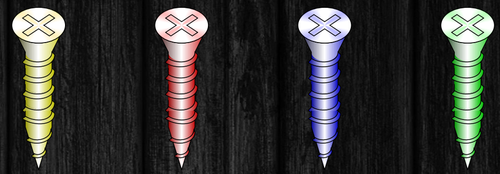
**Goal** — more concurrency

**How?** — do not lock whole data structure; lock individual cells

**Strategy** — “Lock Coupling”; “hand-over-hand lock acquirement”; “acquire lock for next cell before releasing the lock for the current cell”



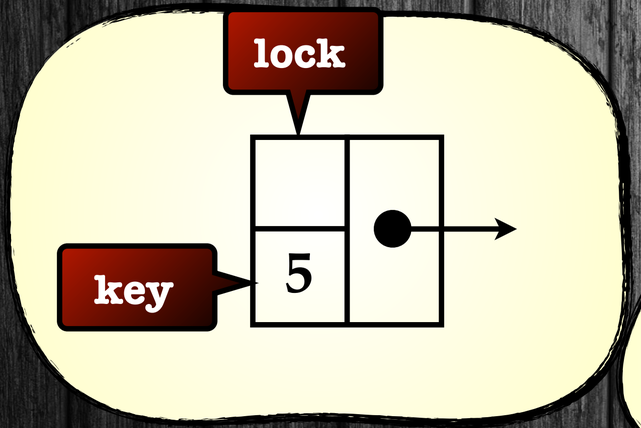
# Sets Fine-Grained Synchronization

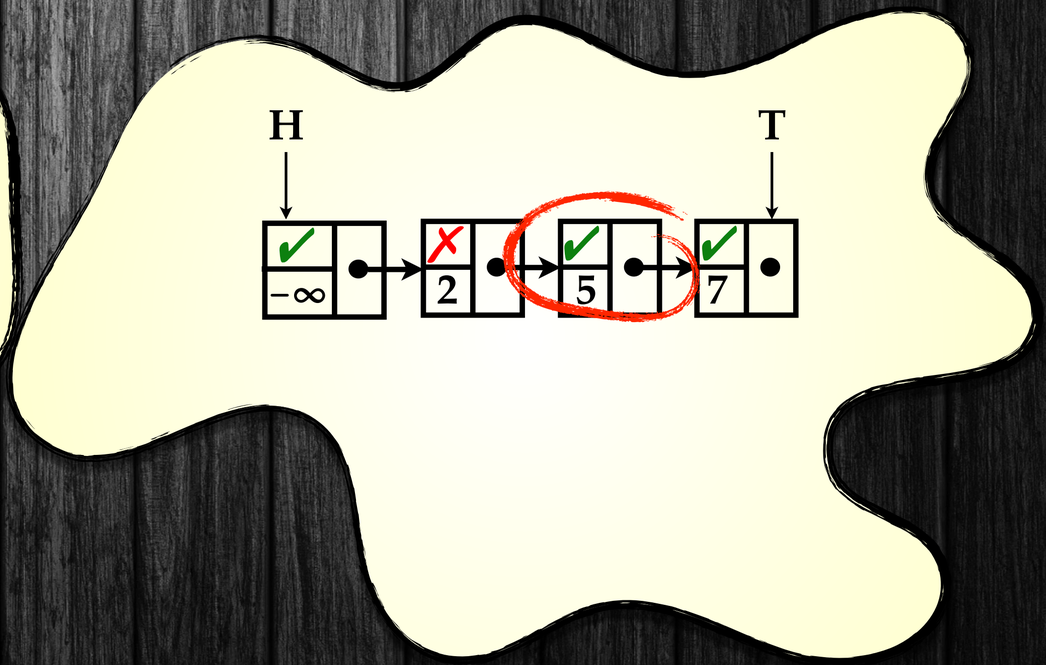


**Goal** — more concurrency

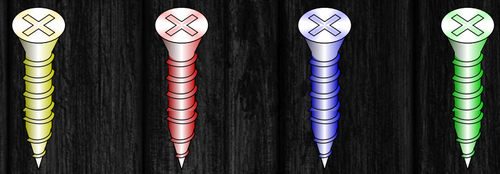
**How?** — do not lock whole data structure; lock individual cells

**Strategy** — "Lock Coupling"; "hand-over-hand lock acquirement"; "acquire lock for next cell before releasing the lock for the current cell"



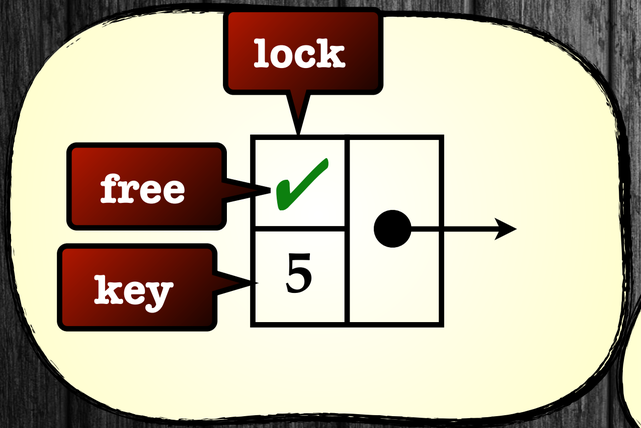
# Sets Fine-Grained Synchronization

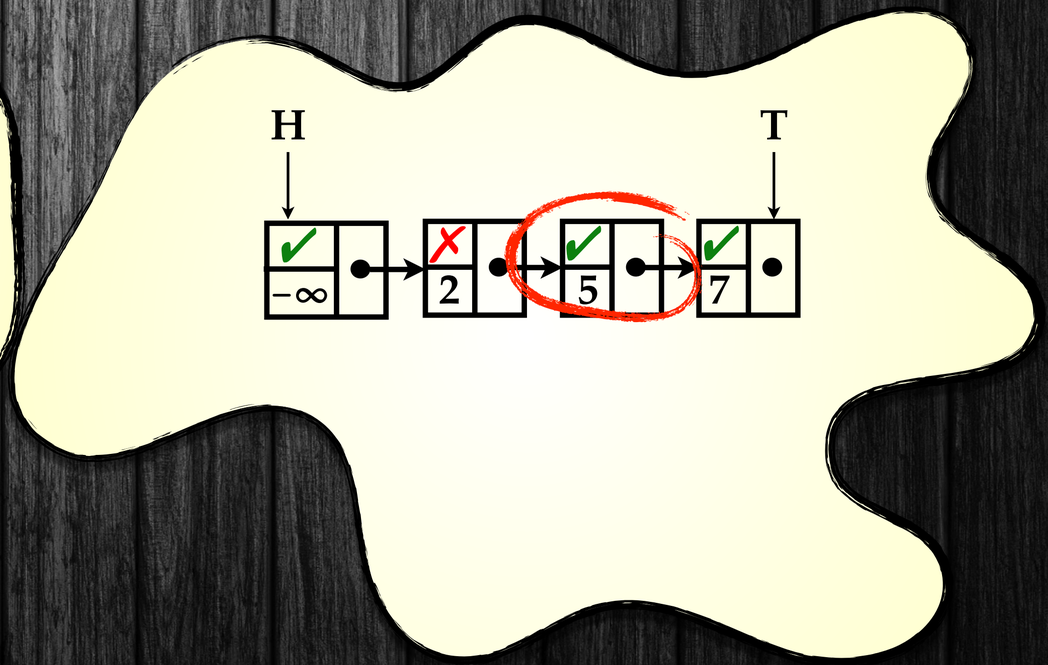


**Goal** — more concurrency

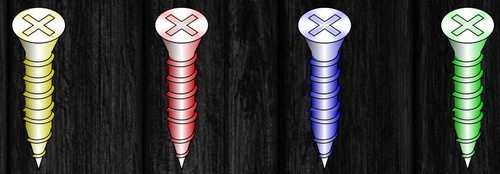
**How?** — do not lock whole data structure; lock individual cells

**Strategy** — “Lock Coupling”; “hand-over-hand lock acquirement”; “acquire lock for next cell before releasing the lock for the current cell”



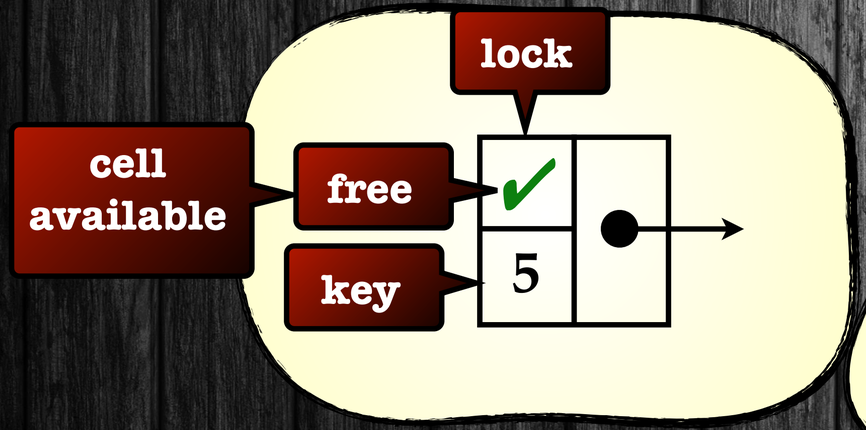
# Sets Fine-Grained Synchronization

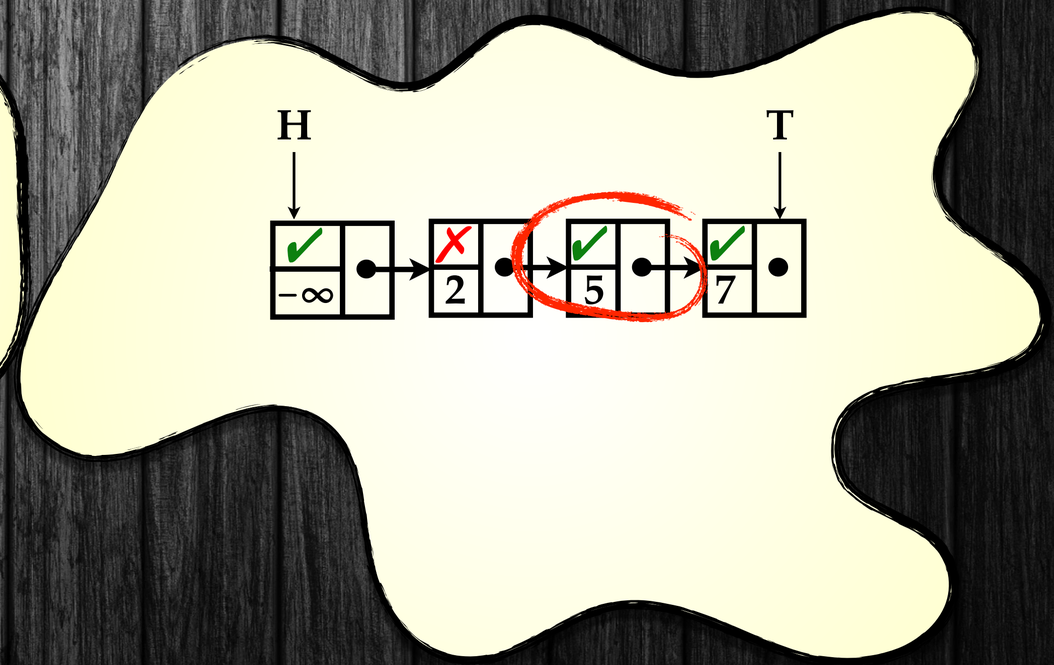


Goal: more concurrency

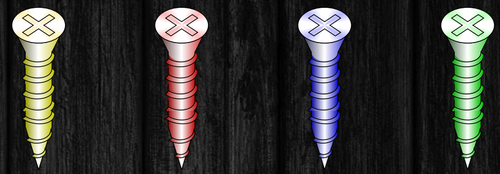
How?: do not lock whole data structure; lock individual cells

Strategy: "Lock Coupling"; "hand-over-hand lock acquirement"; "acquire lock for next cell before releasing the lock for the current cell"



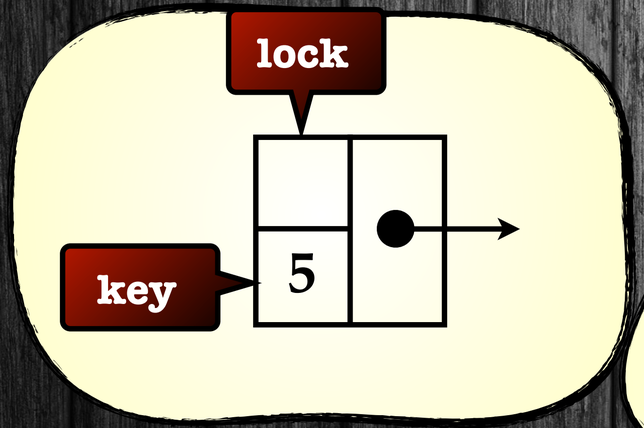
# Sets Fine-Grained Synchronization

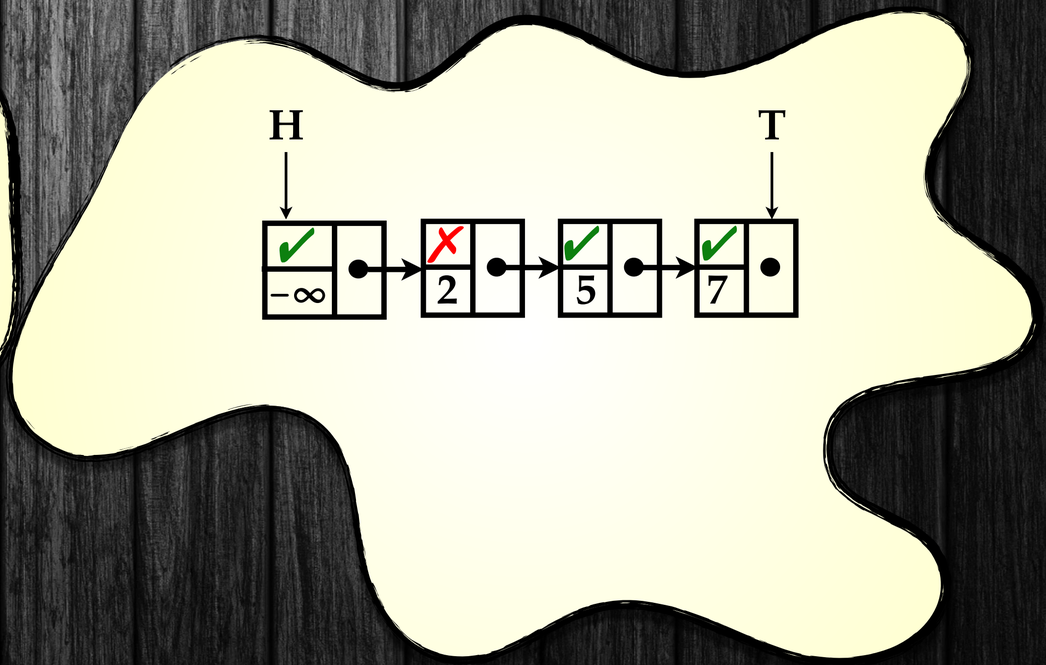


**Goal** — more concurrency

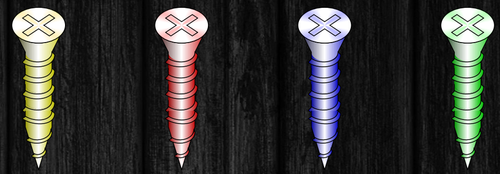
**How?** — do not lock whole data structure; lock individual cells

**Strategy** — "Lock Coupling"; "hand-over-hand lock acquirement"; "acquire lock for next cell before releasing the lock for the current cell"



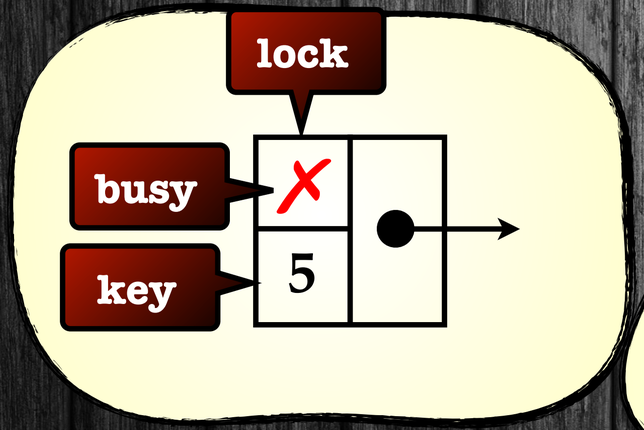
# Sets Fine-Grained Synchronization

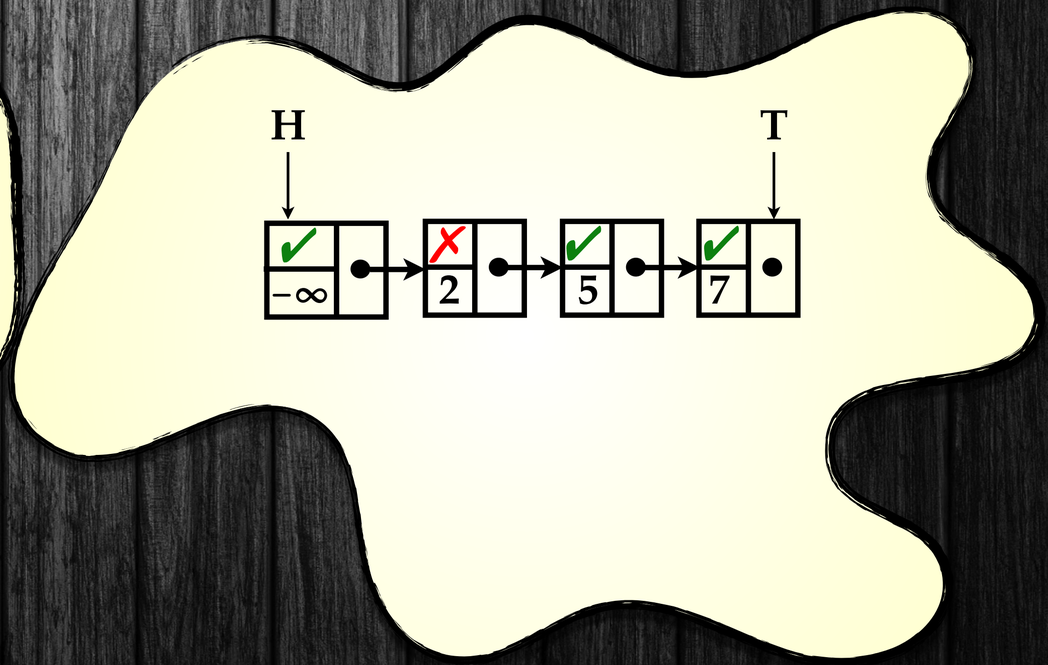


**Goal** — more concurrency

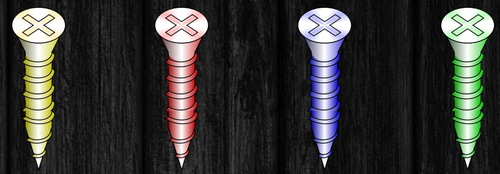
**How?** — do not lock whole data structure; lock individual cells

**Strategy** — “Lock Coupling”; “hand-over-hand lock acquirement”; “acquire lock for next cell before releasing the lock for the current cell”



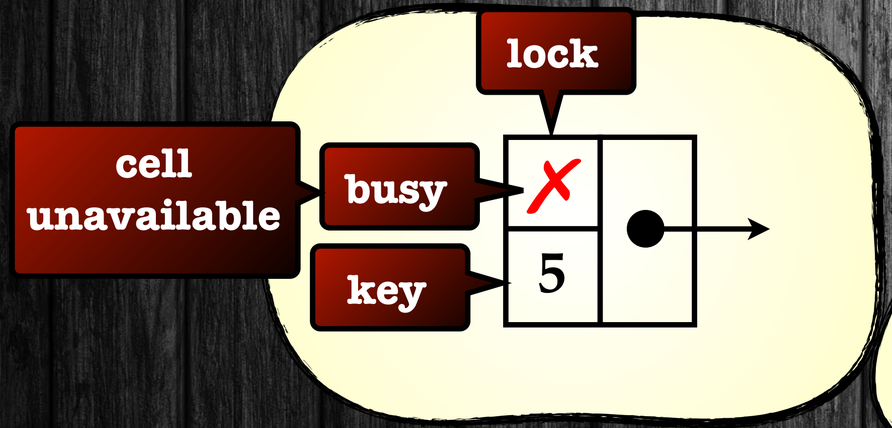
# Sets Fine-Grained Synchronization

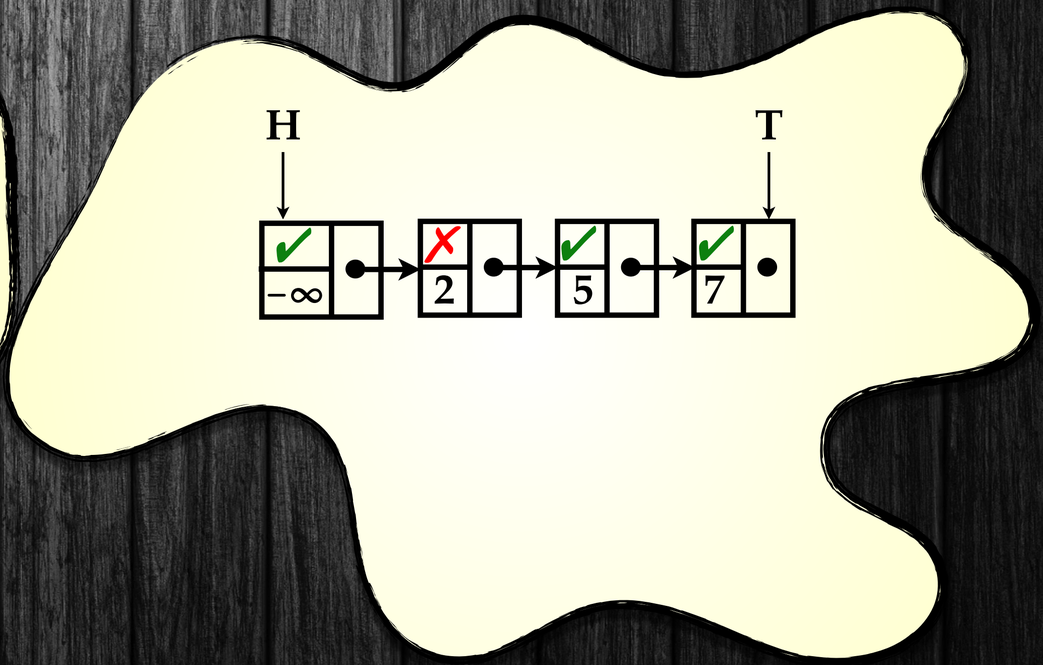


Goal: more concurrency

How?: do not lock whole data structure; lock individual cells

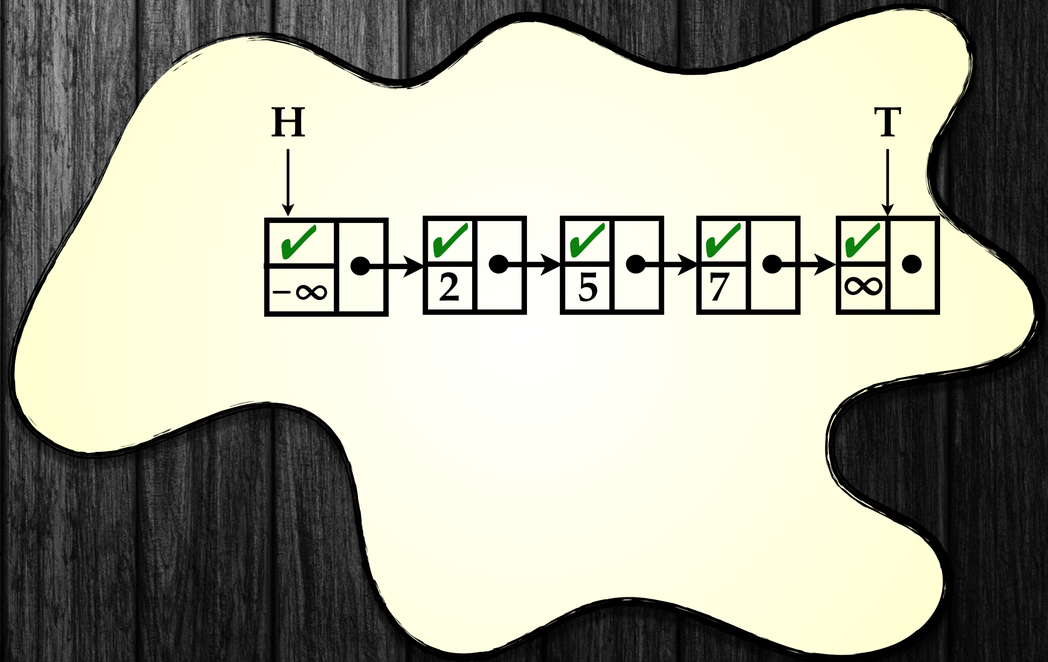
Strategy: "Lock Coupling"; "hand-over-hand lock acquirement"; "acquire lock for next cell before releasing the lock for the current cell"



# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

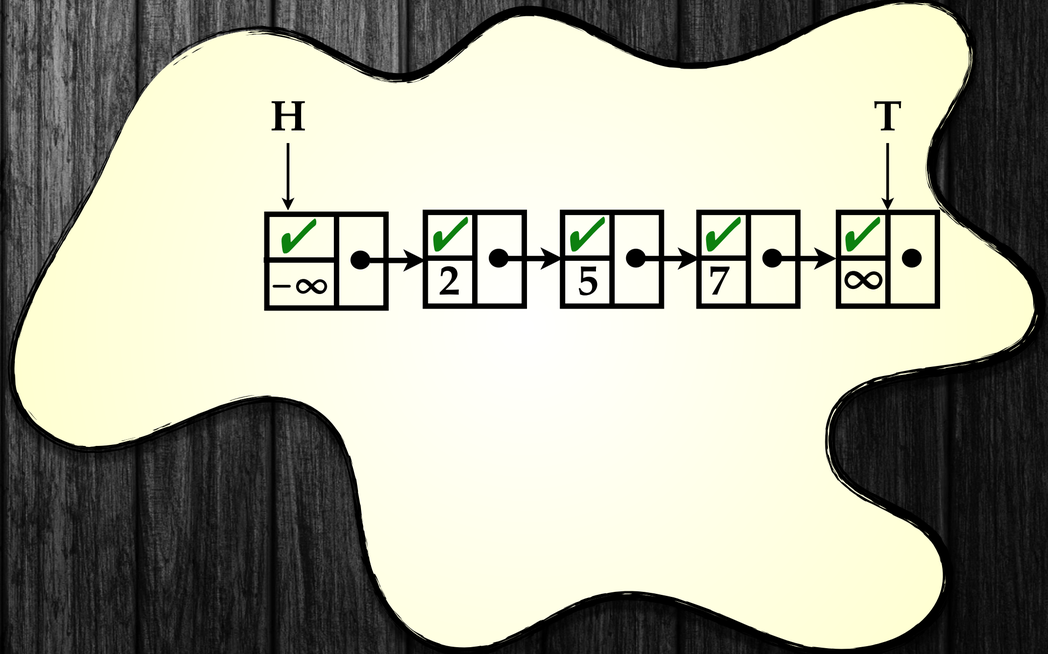


# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(4)



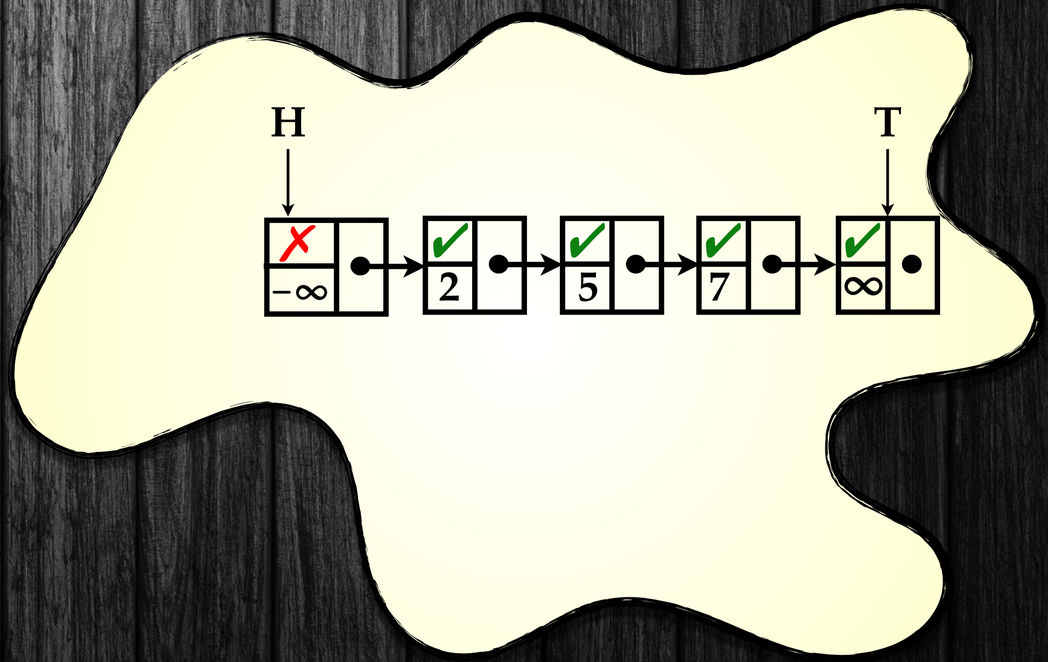
# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



add(4)



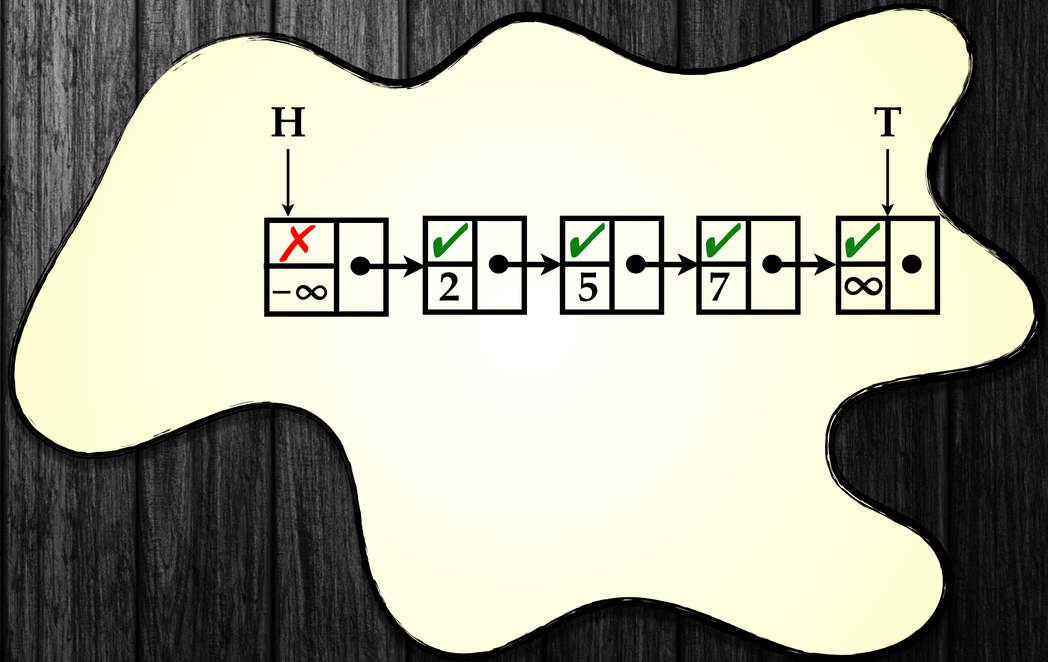
# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



add(4)



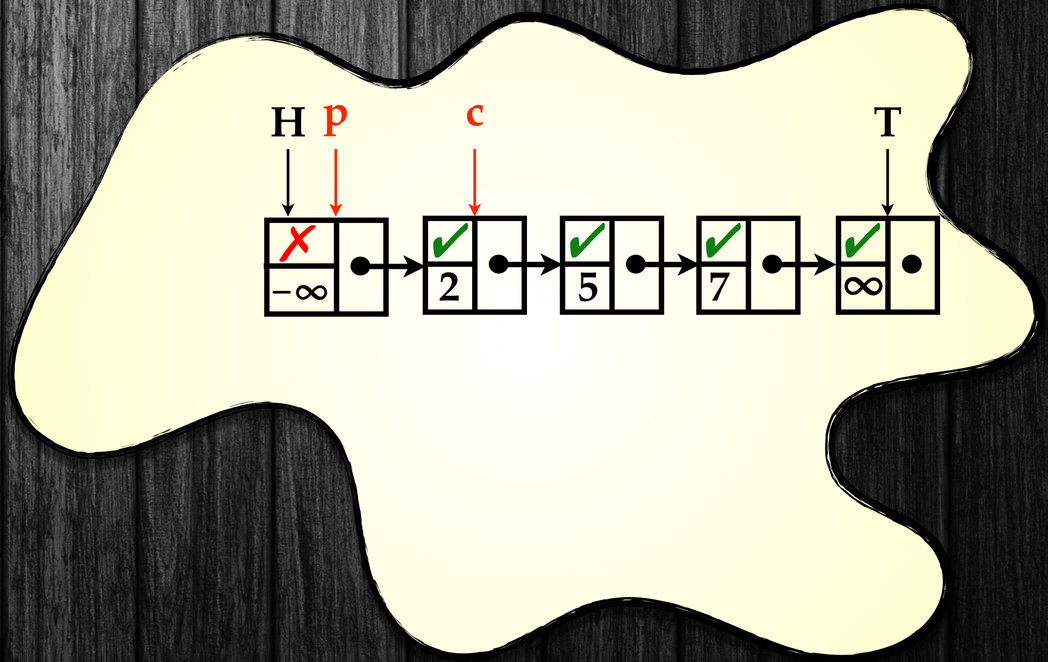
# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



add(4)

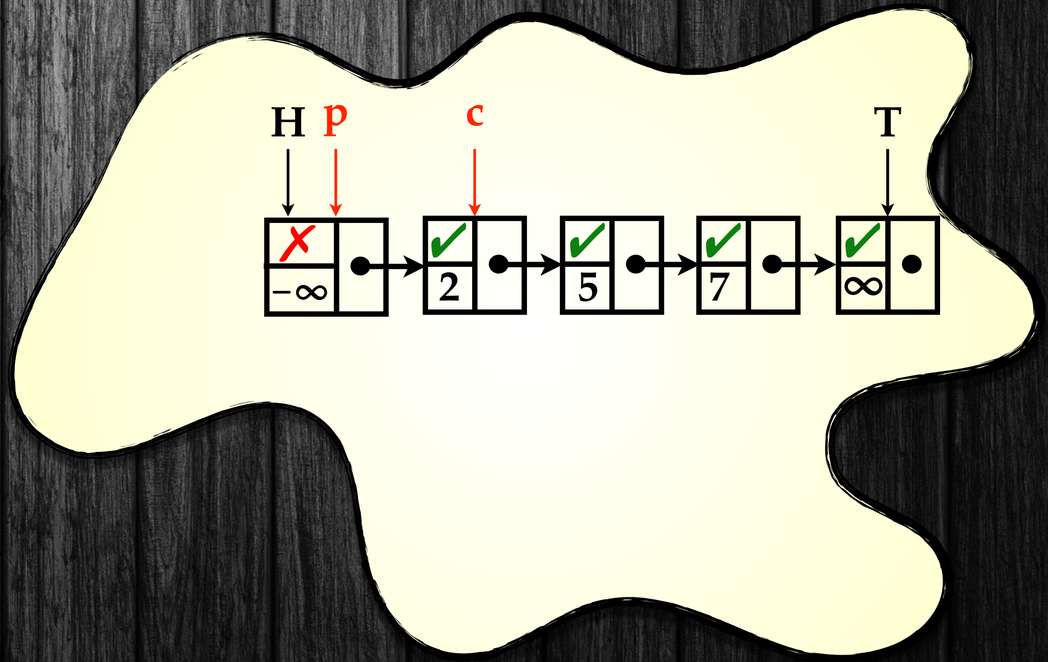


# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

add(4)



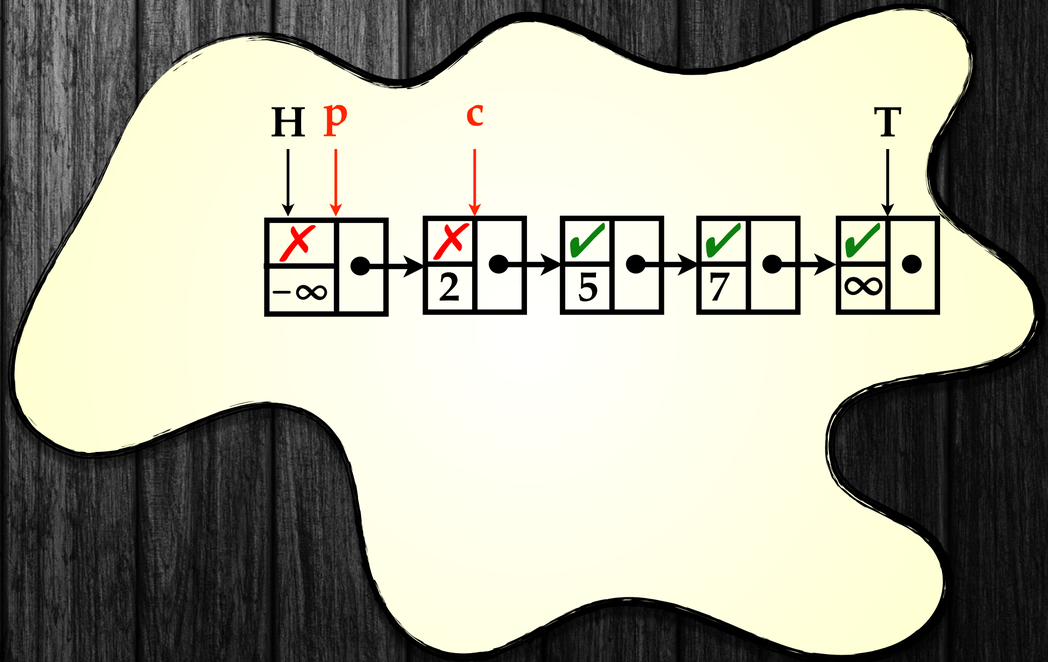
# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```



add(4)



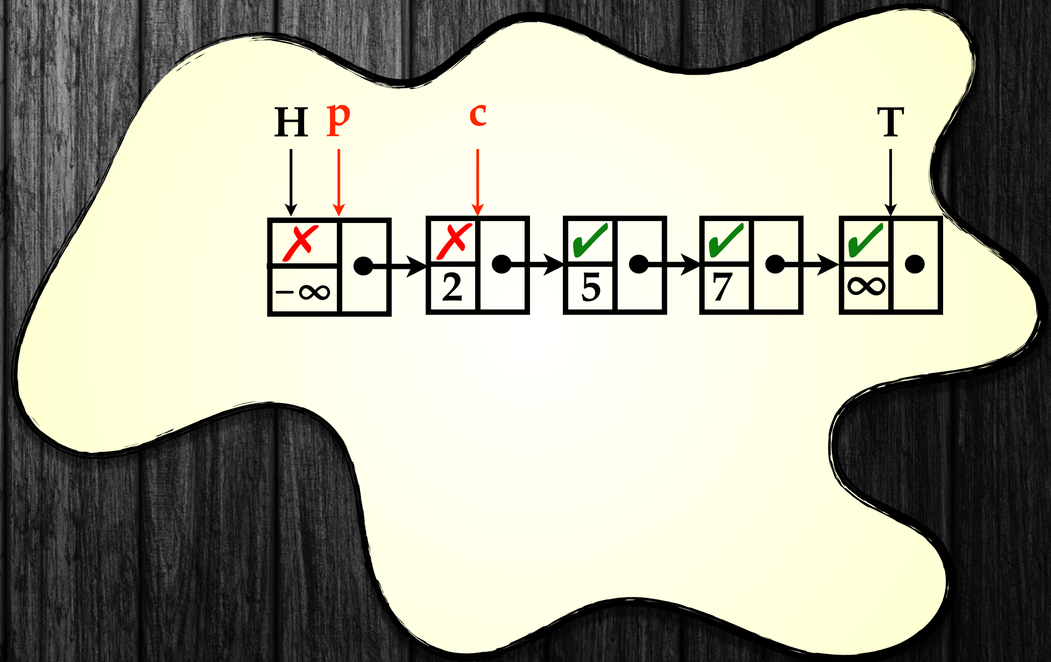
# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```



add(4)

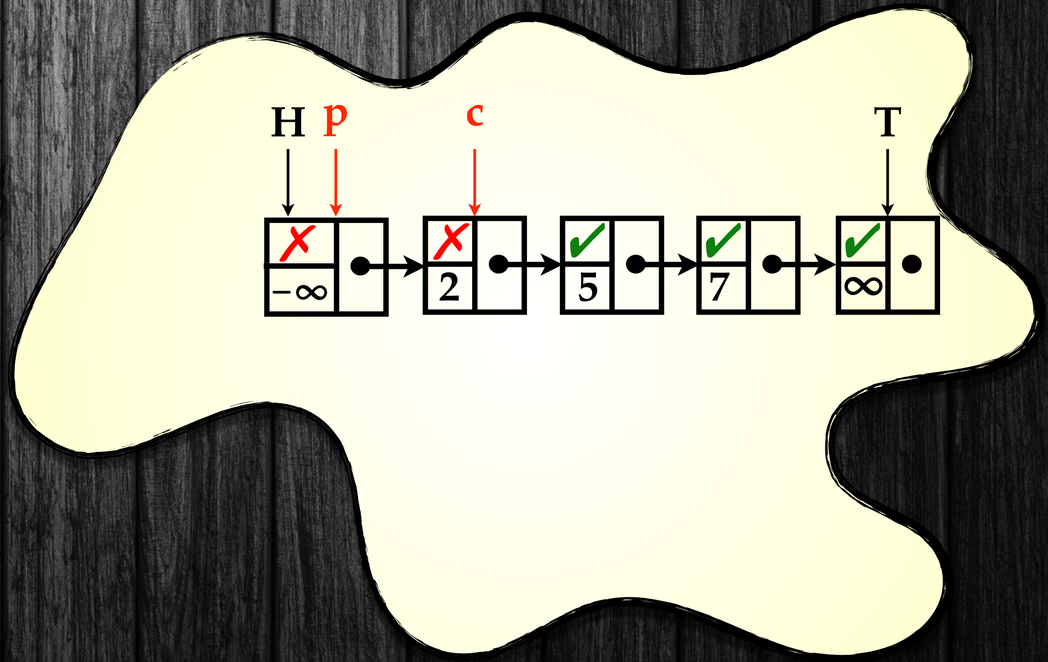


# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

add(4)

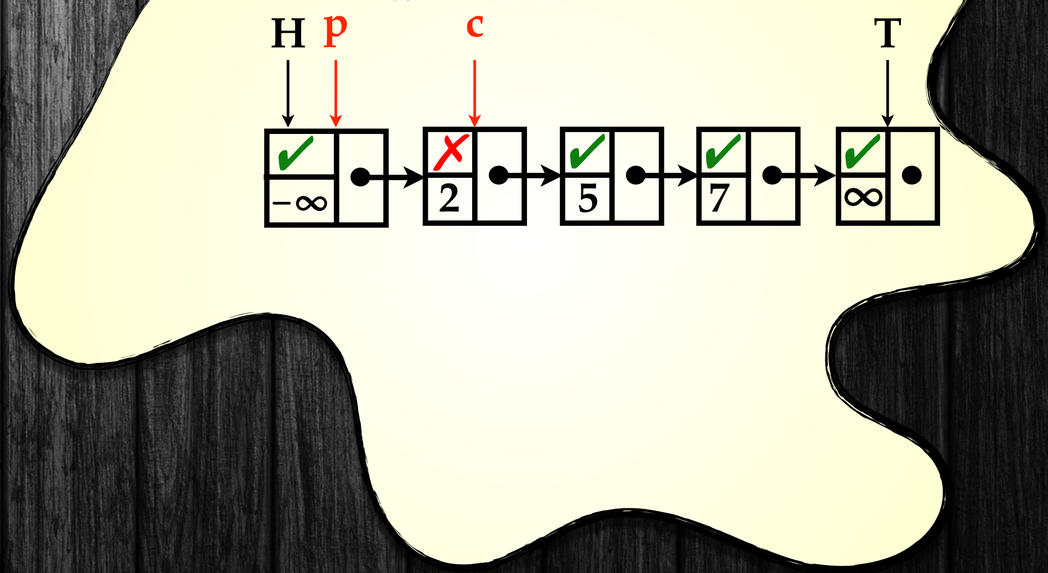


# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

add(4)

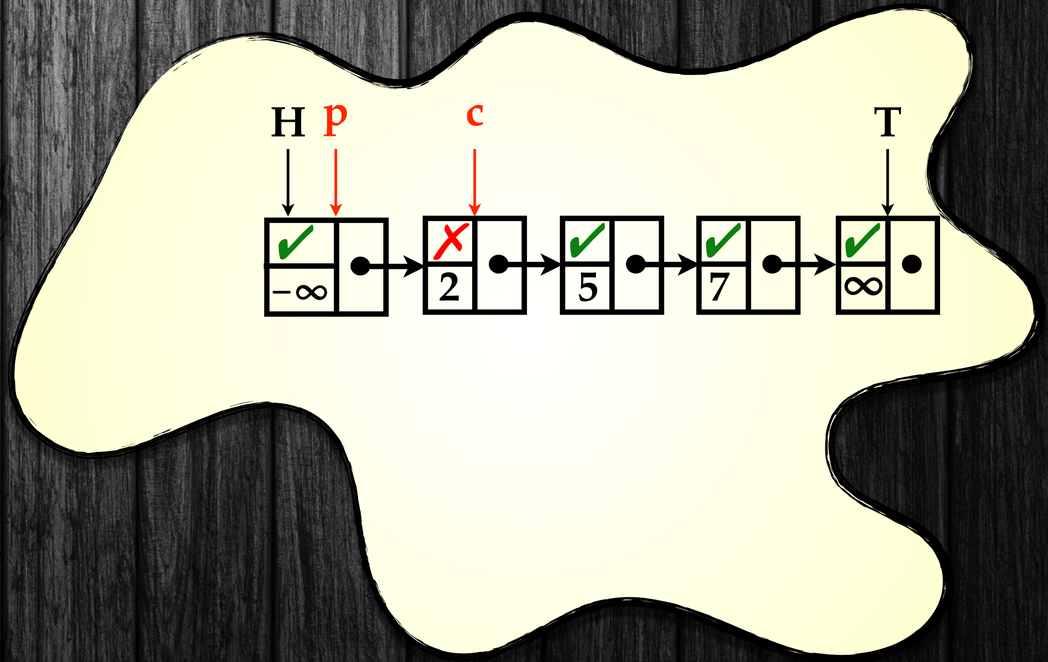


# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

add(4)



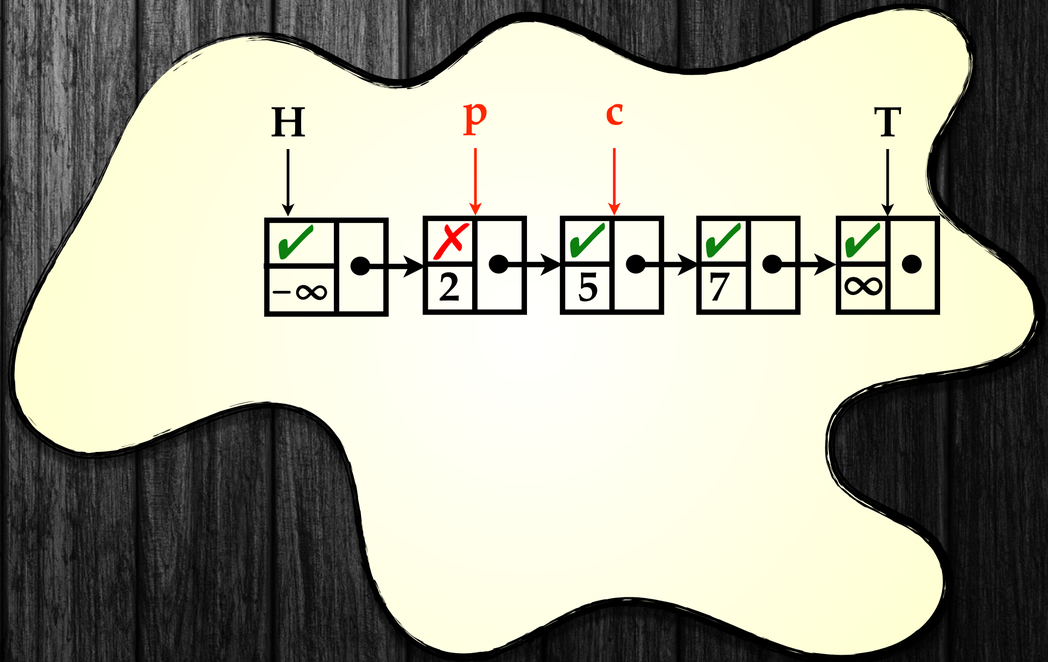
# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```



add(4)

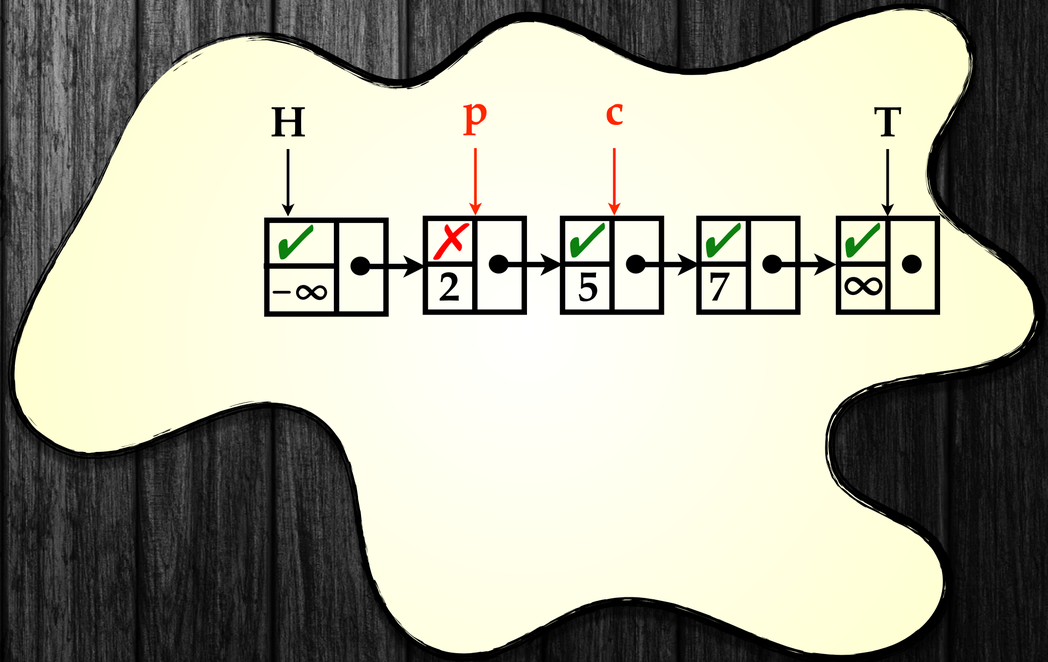


# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

add(4)

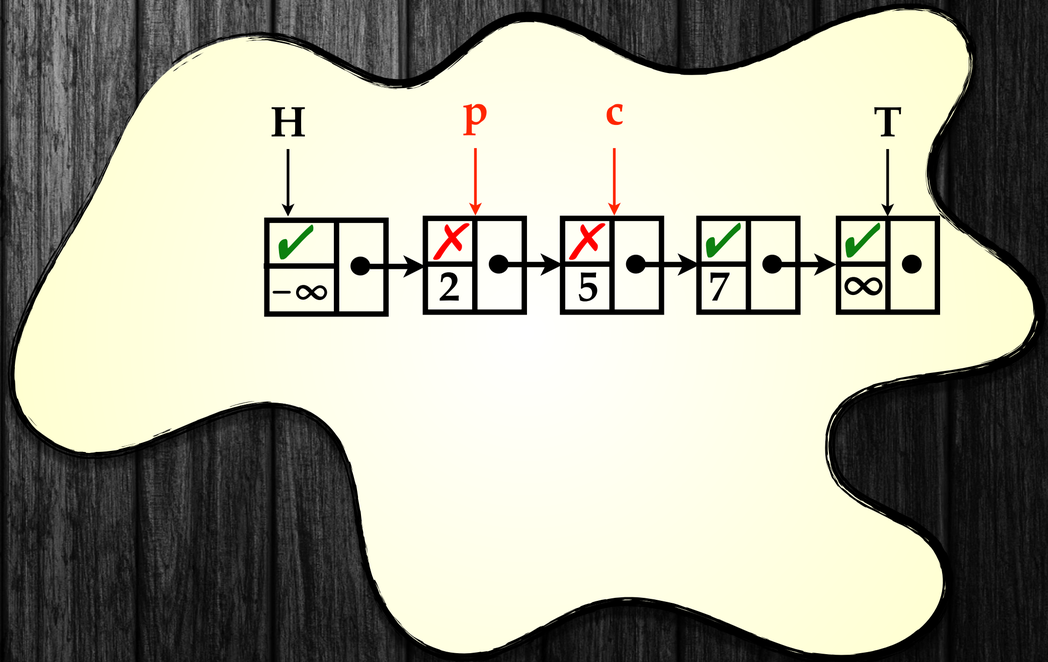


# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

add(4)

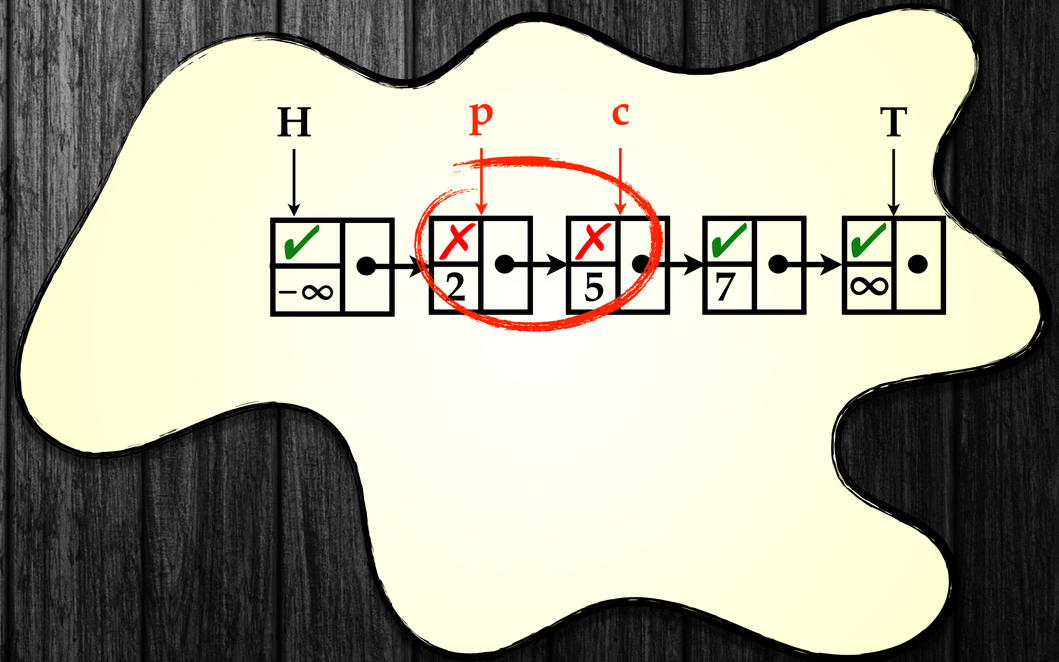


# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

add(4)

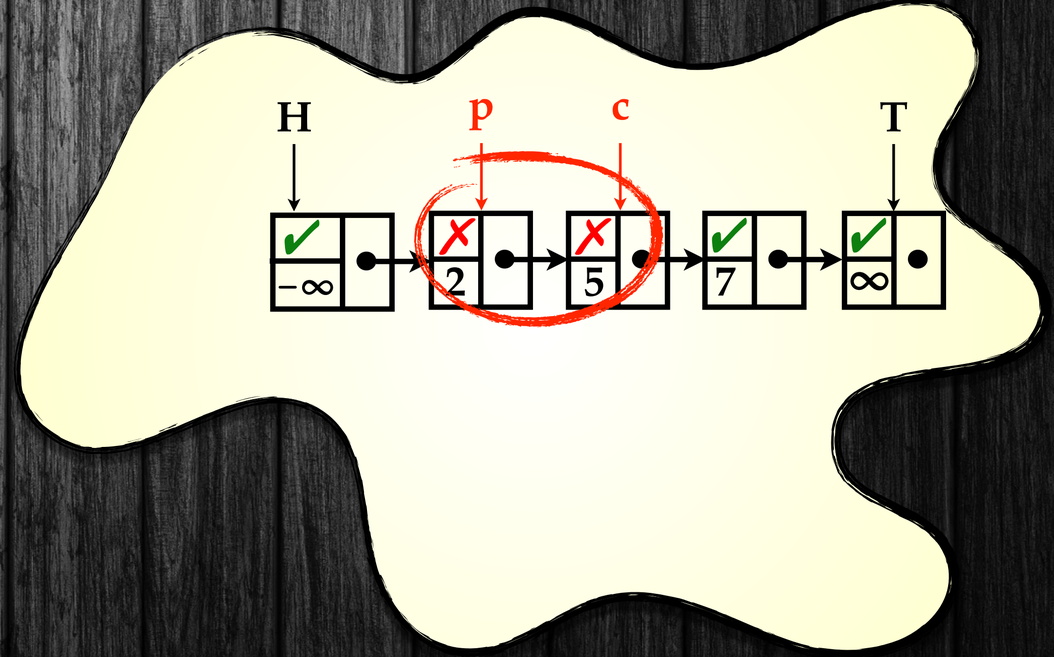


# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

add(4)



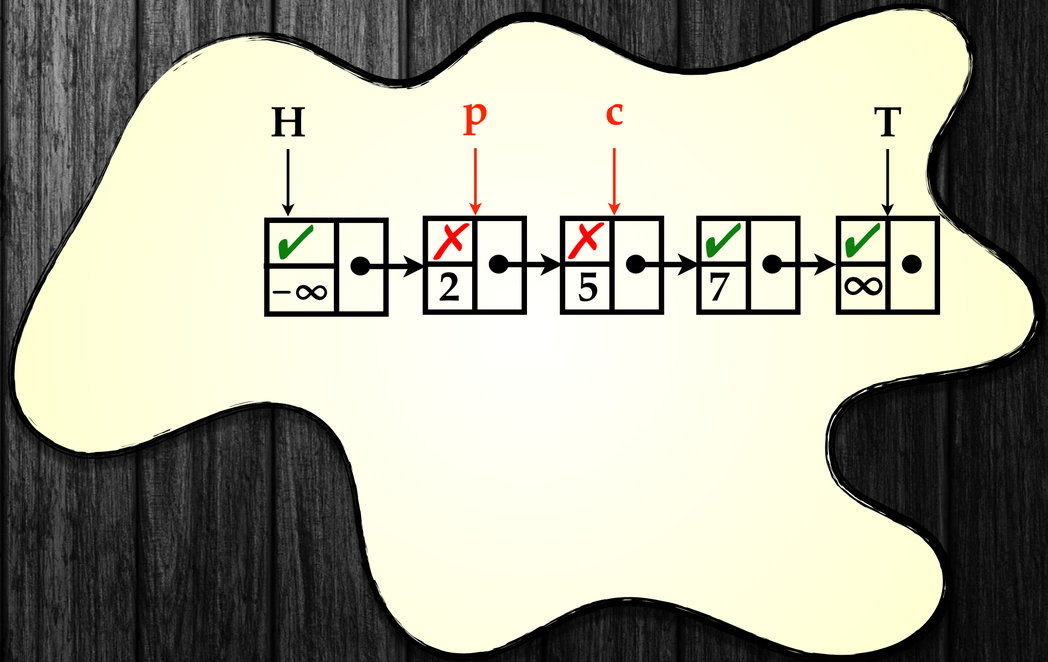
# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```



add(4)



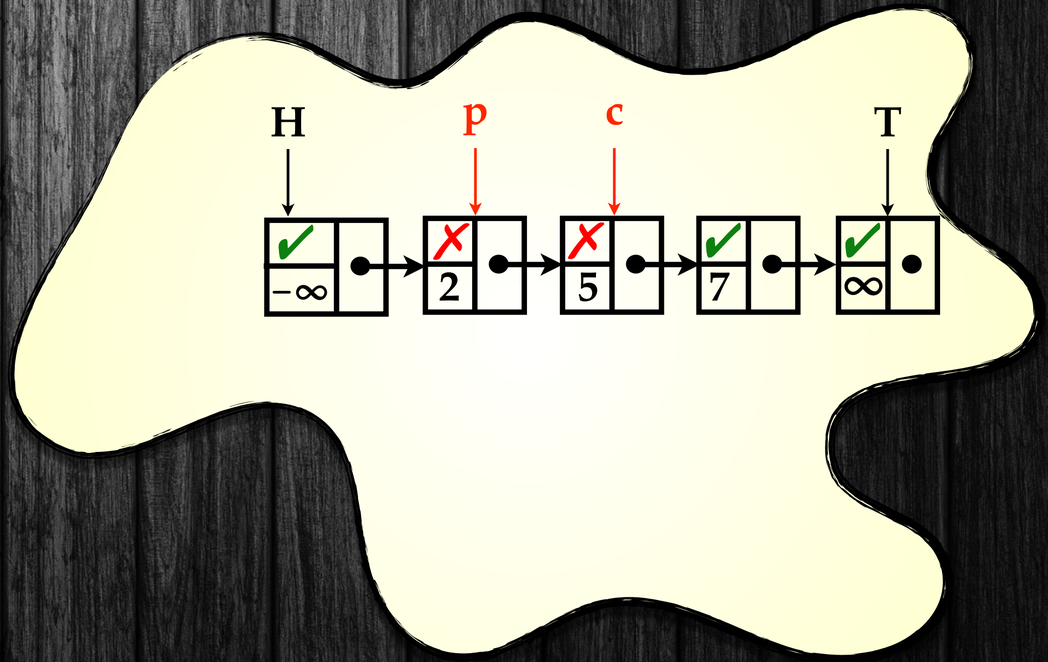
# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```



add(4)

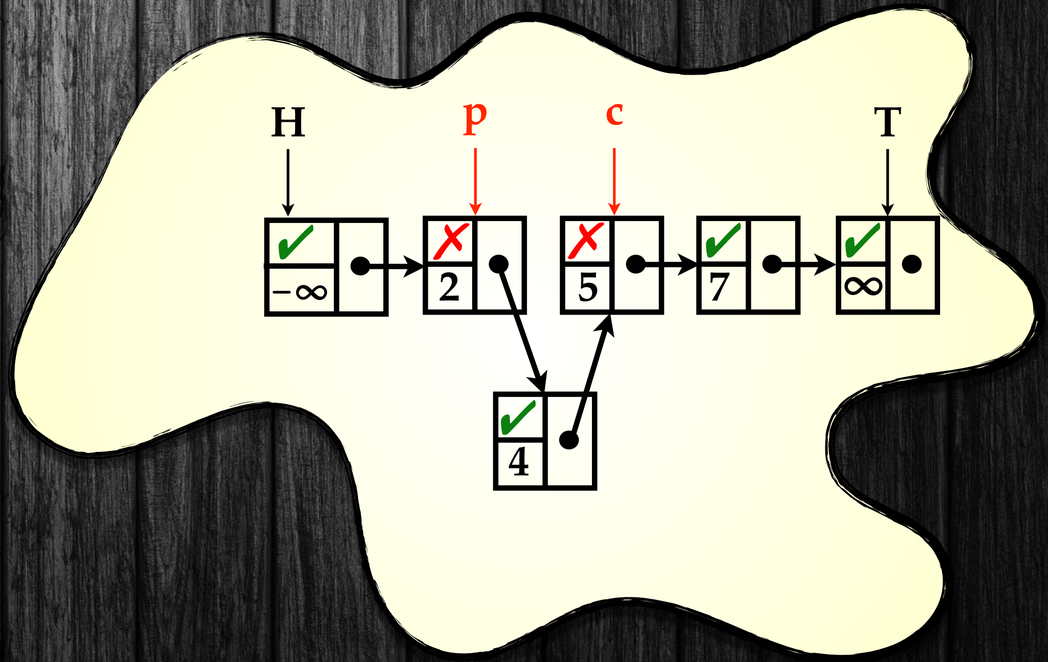


# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

add(4)

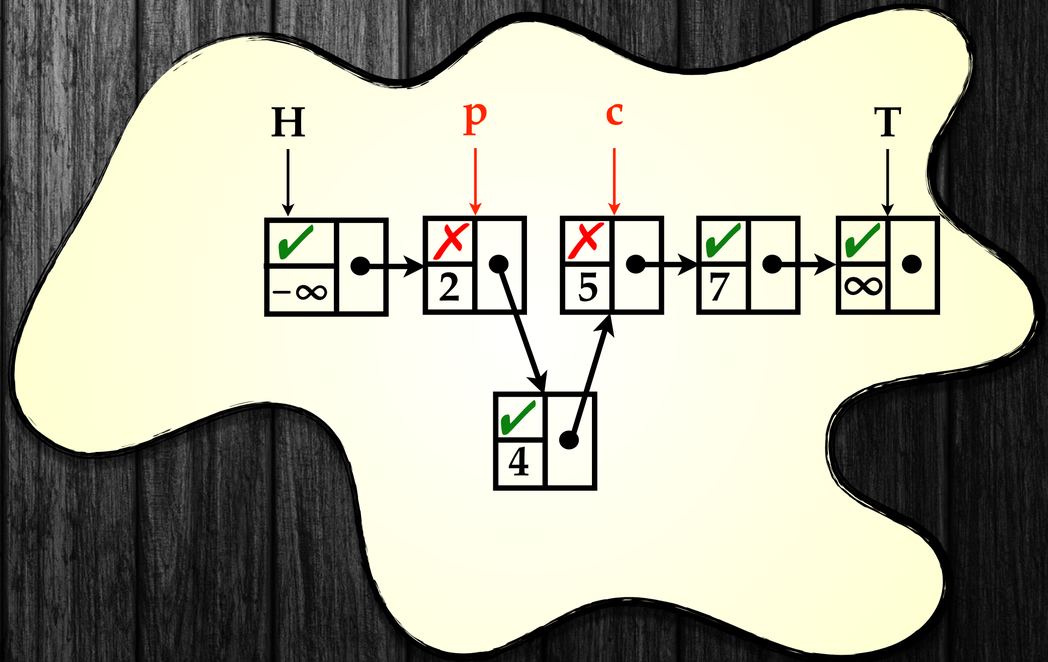


# Sets | Fine-Grained Synchronization

"hand-over-hand lock acquirement"

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

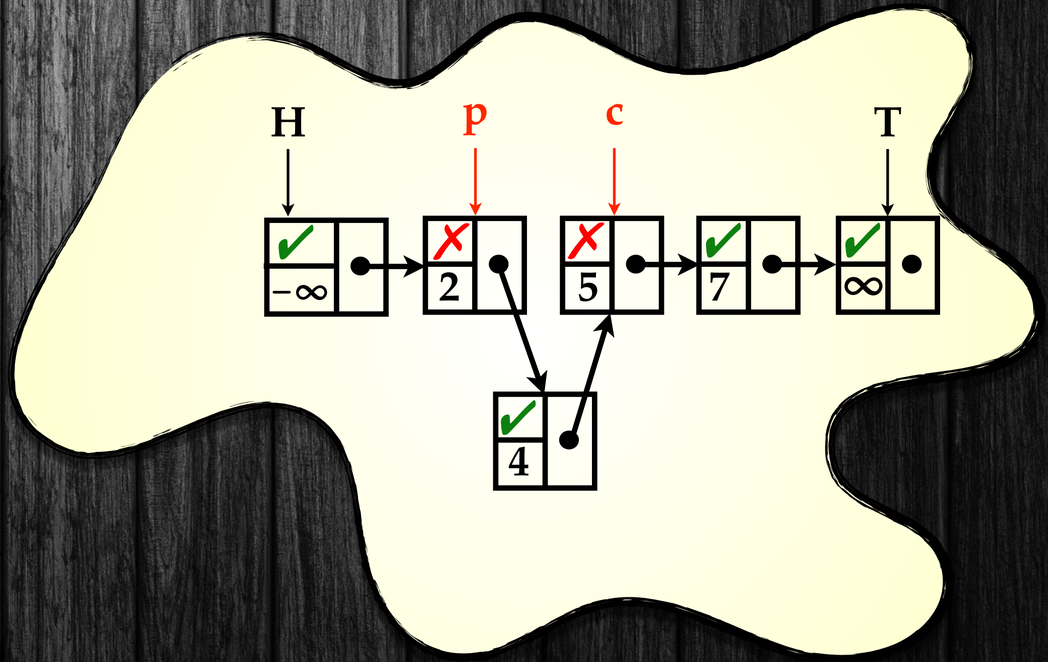
add(4)



# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

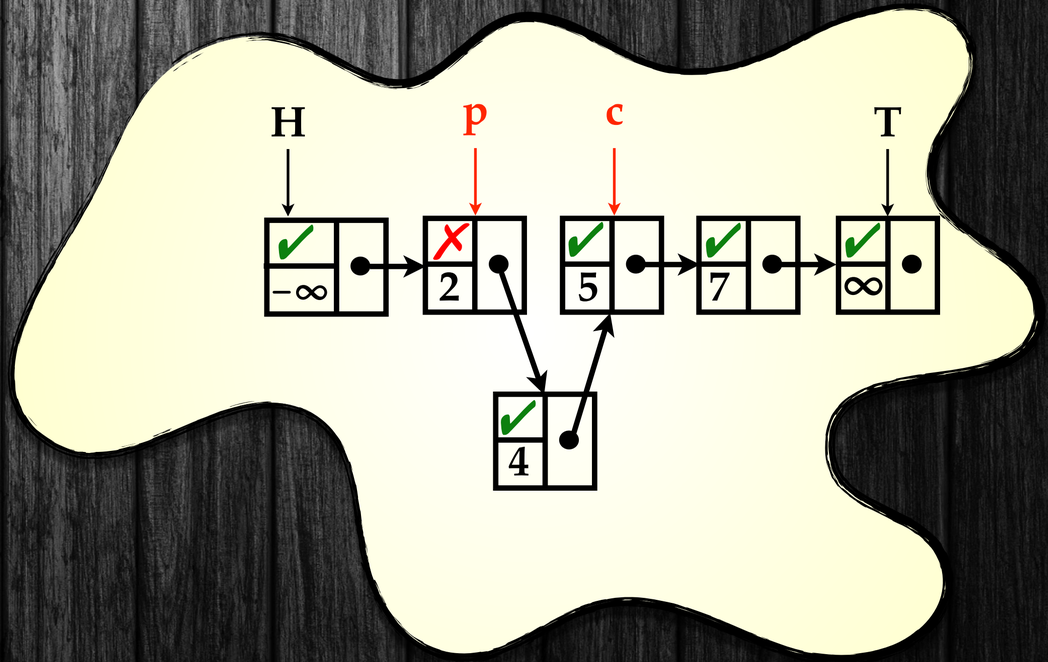
add(4)



# Sets Fine-Grained Synchronization

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

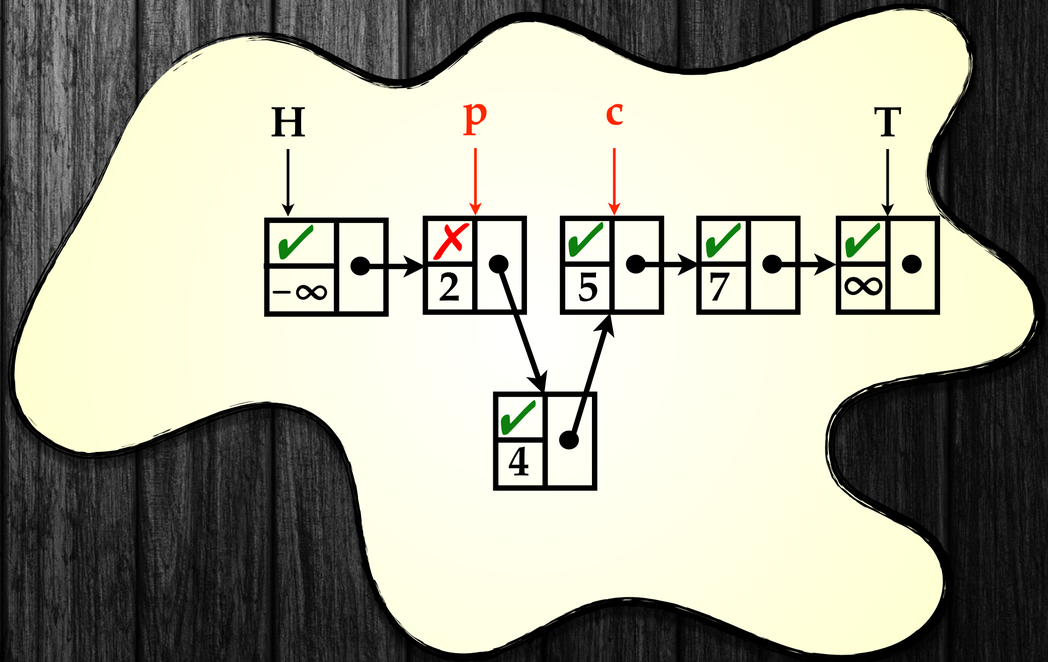
add(4)



# Sets Fine-Grained Synchronization

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

add(4)

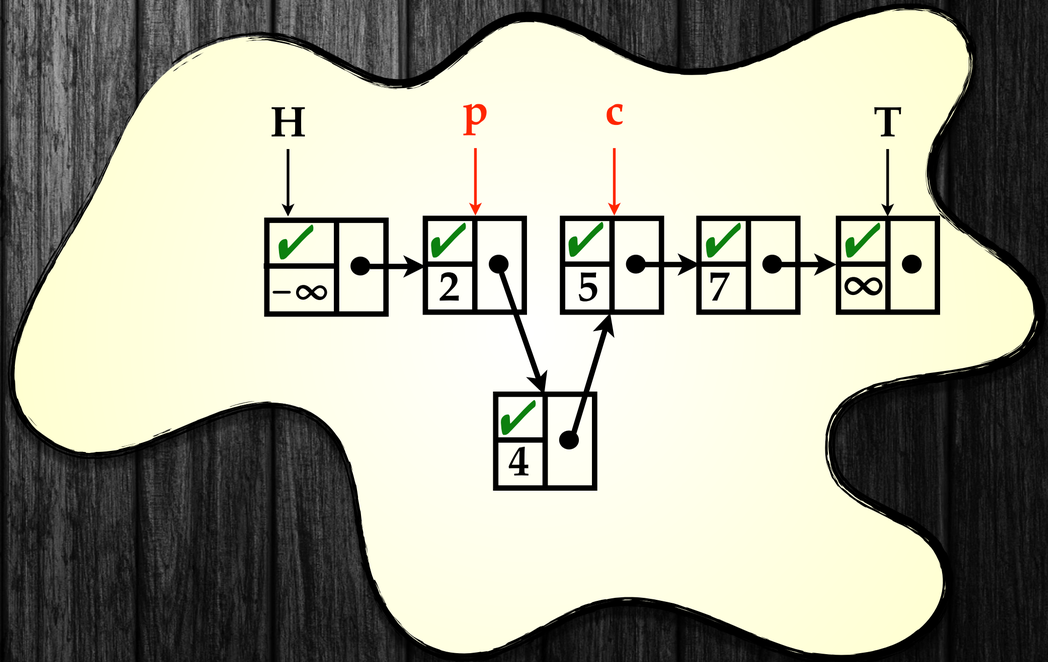


# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```



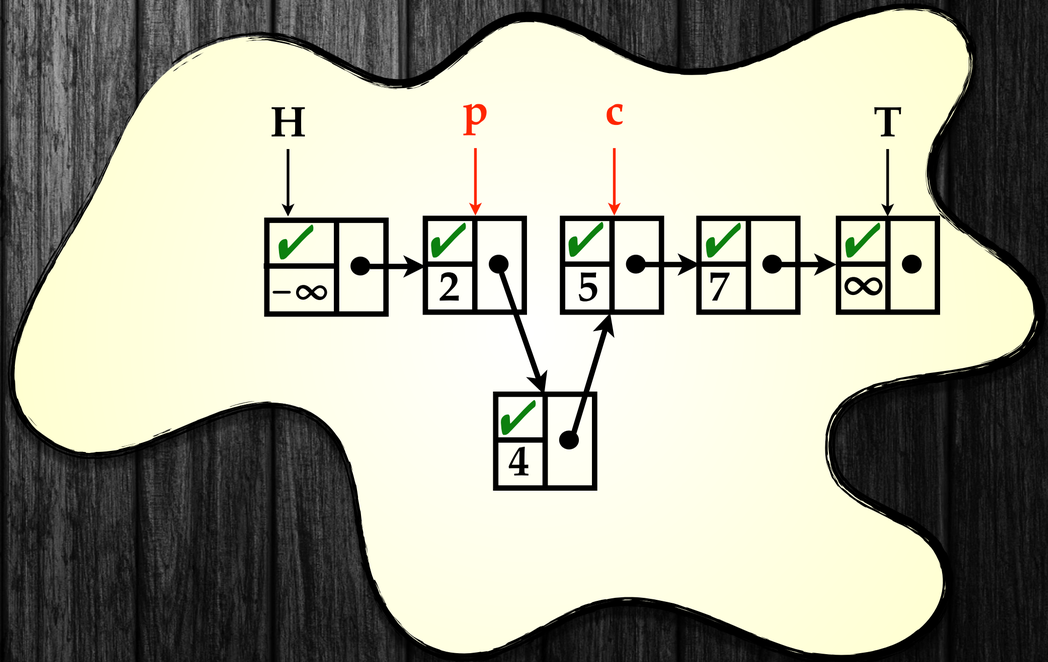
add(4)



# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17  unlock(c)
18  unlock(p)
```

add(4)



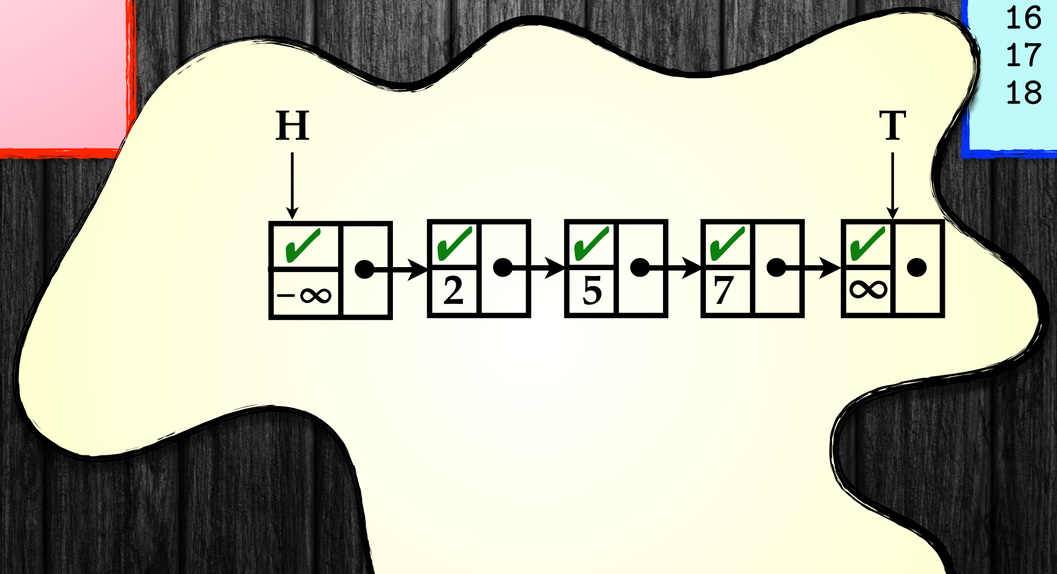
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)


```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

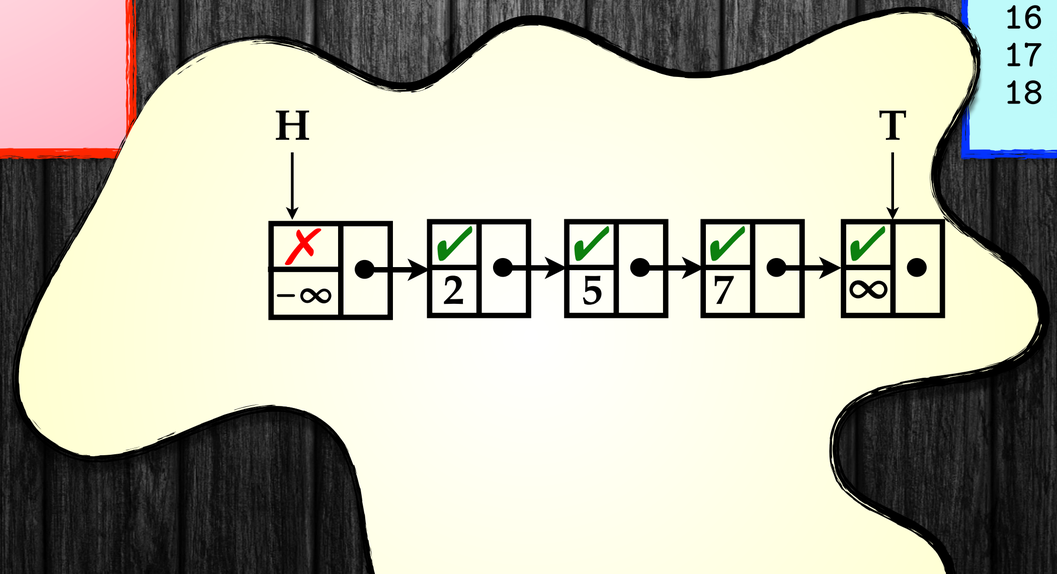
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)


```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

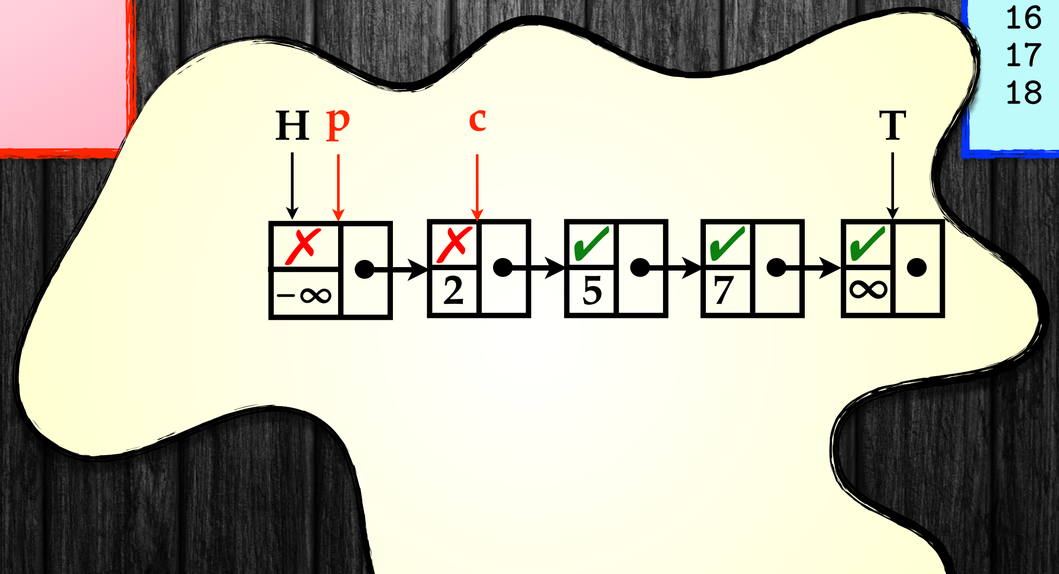
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



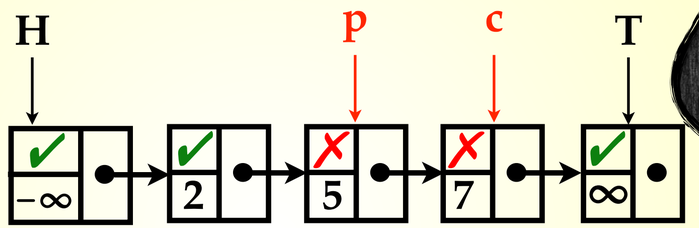
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



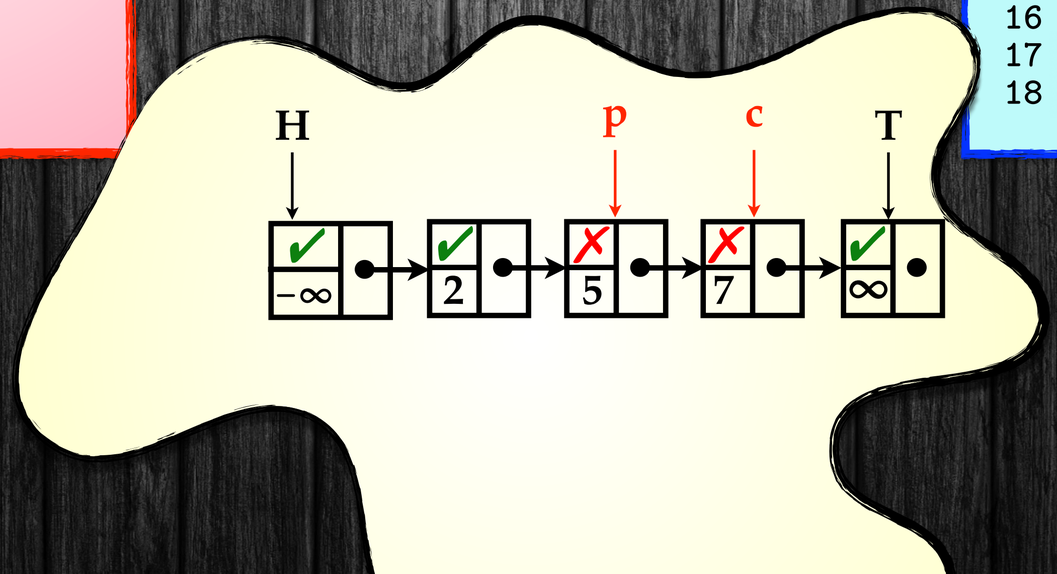
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



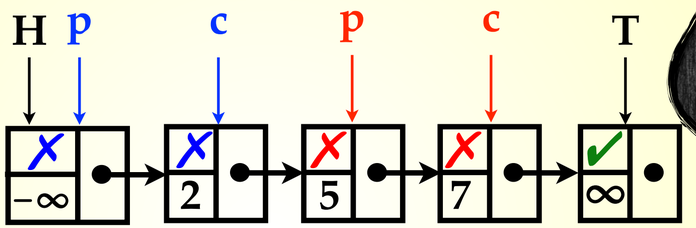
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



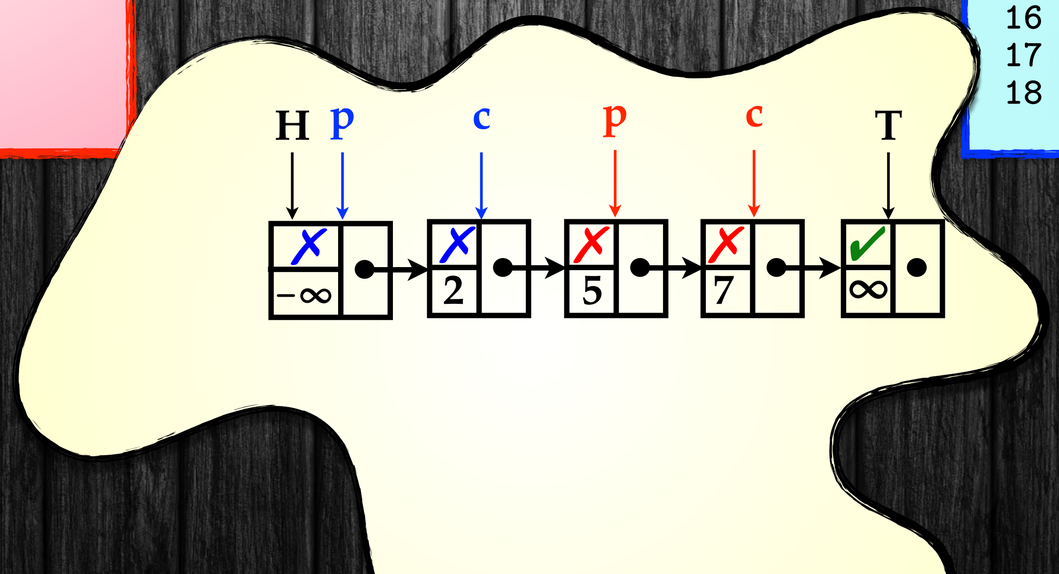
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



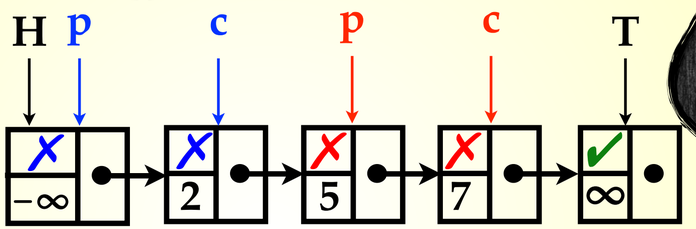
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



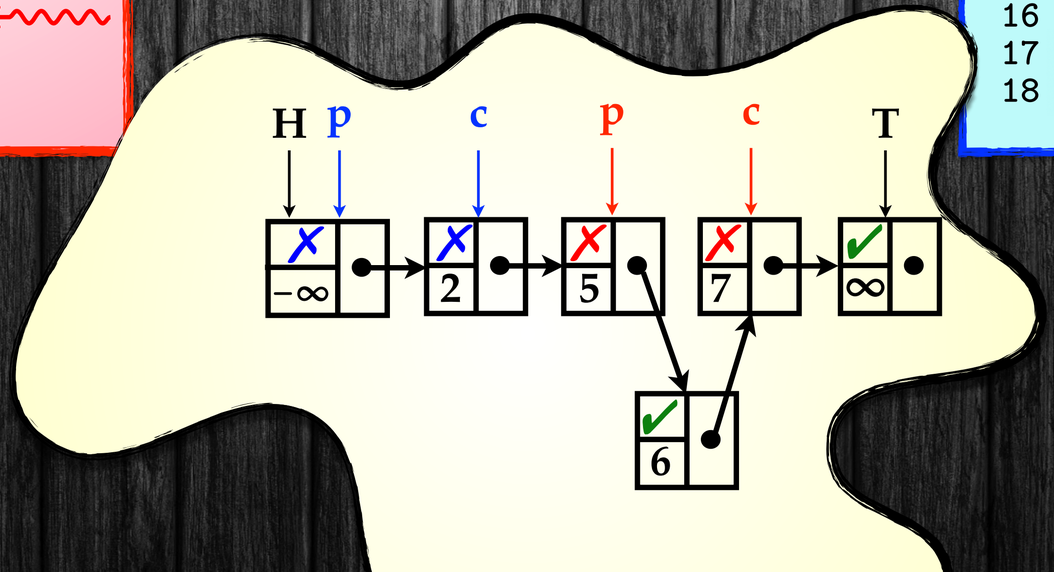
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



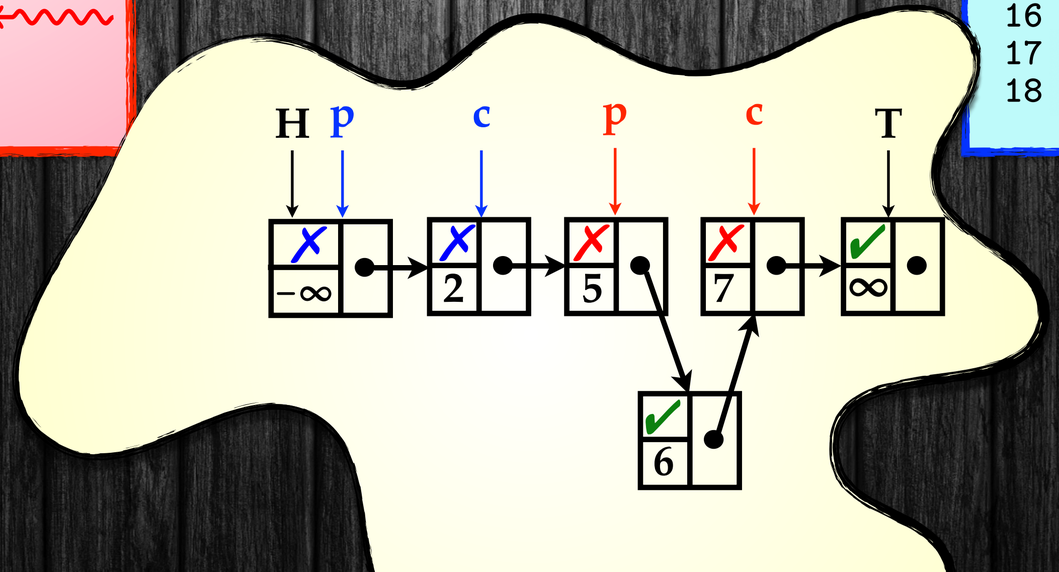
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



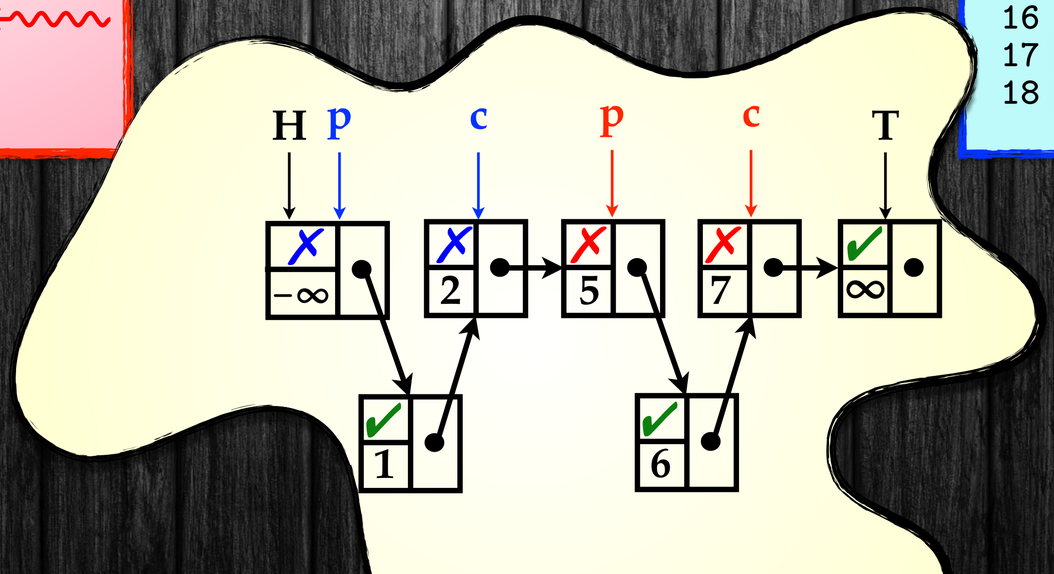
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



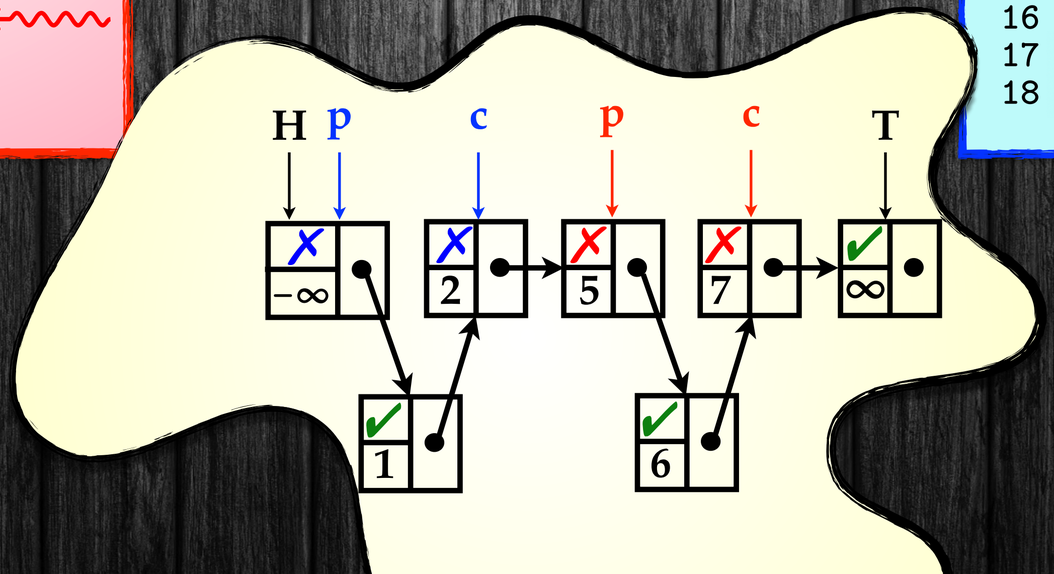
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



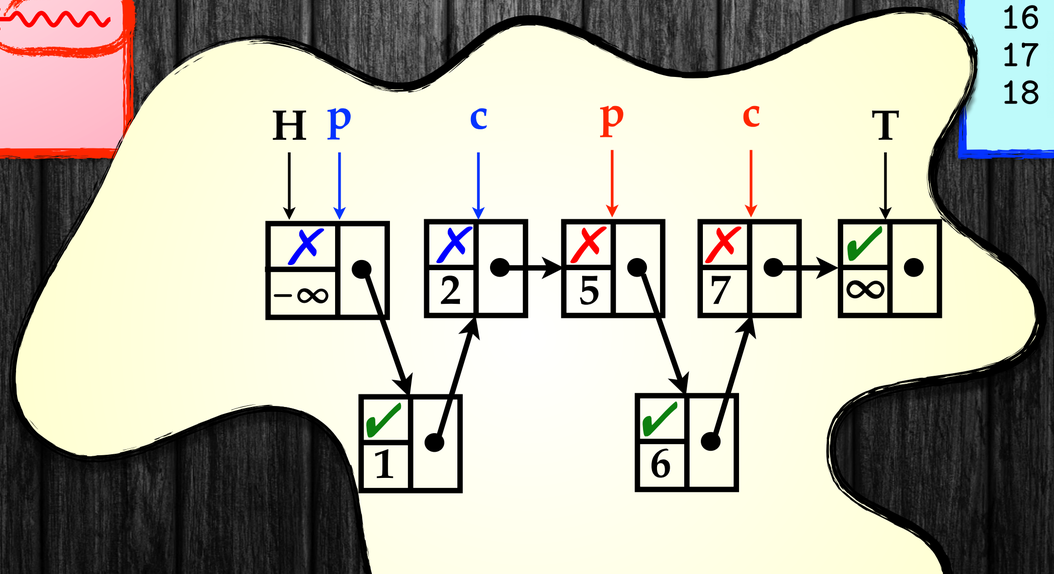
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



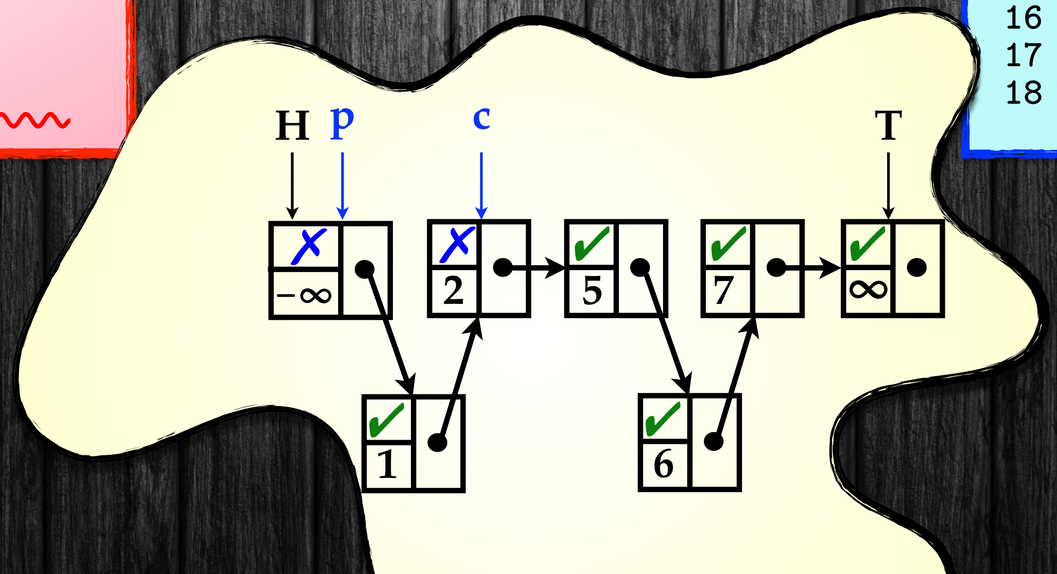
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



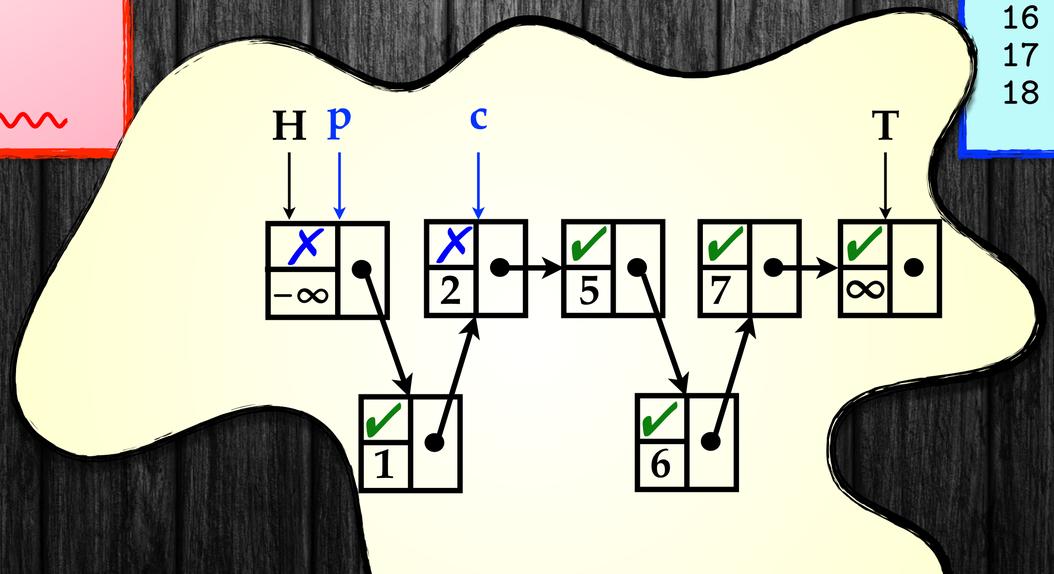
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



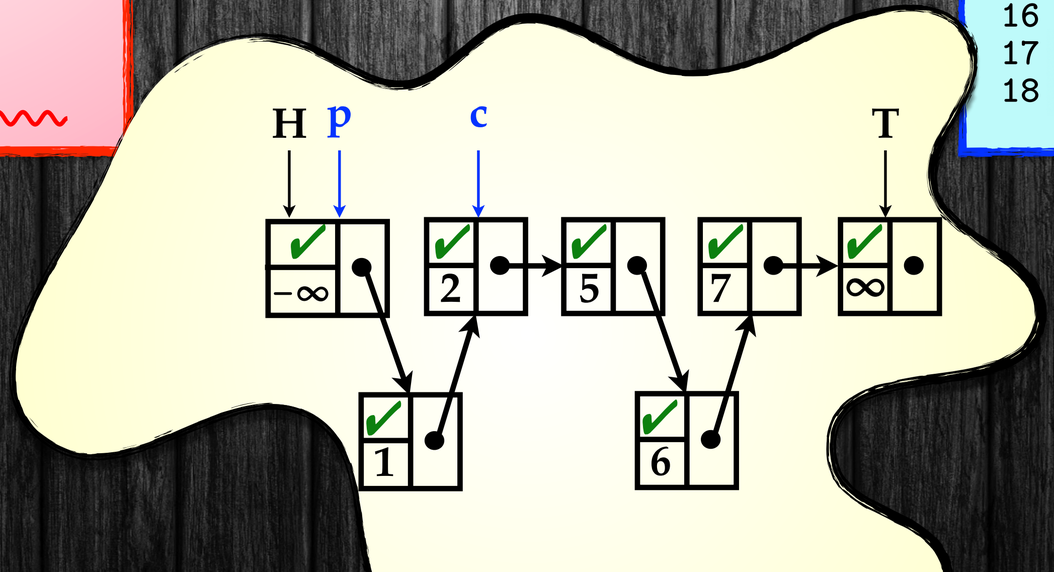
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



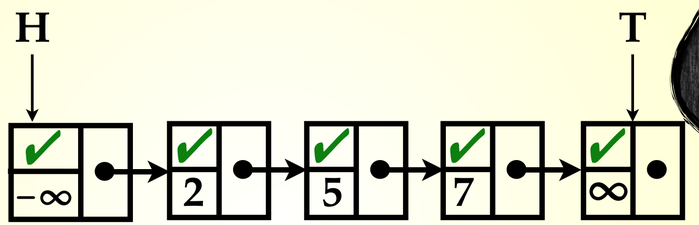
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



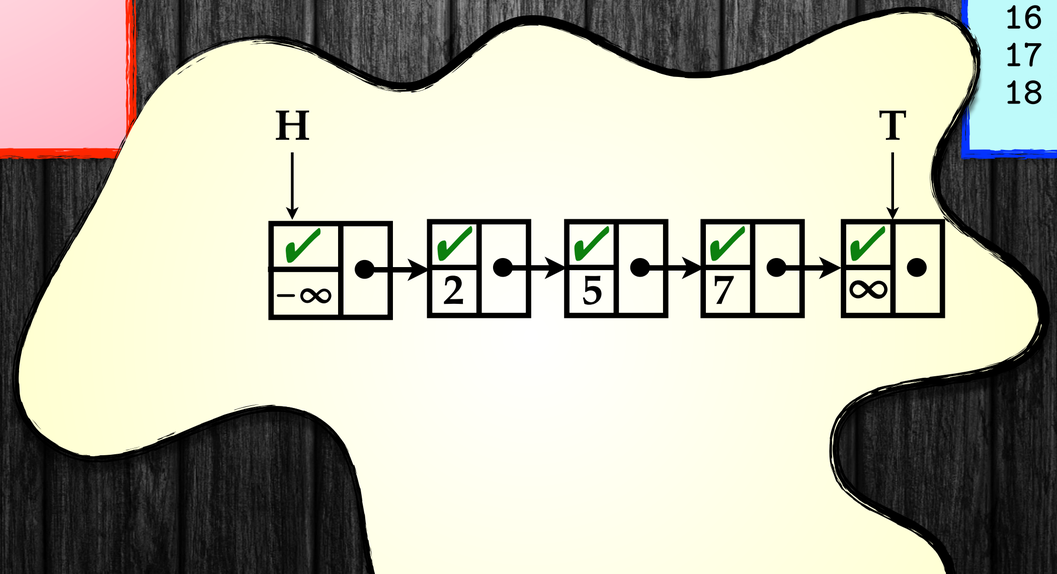
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



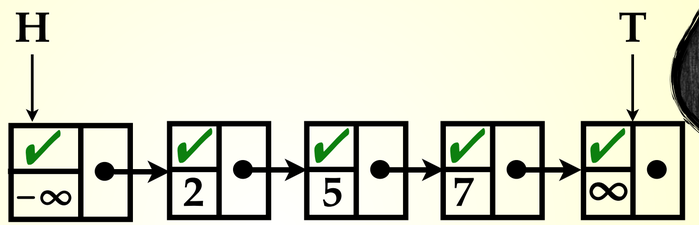
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



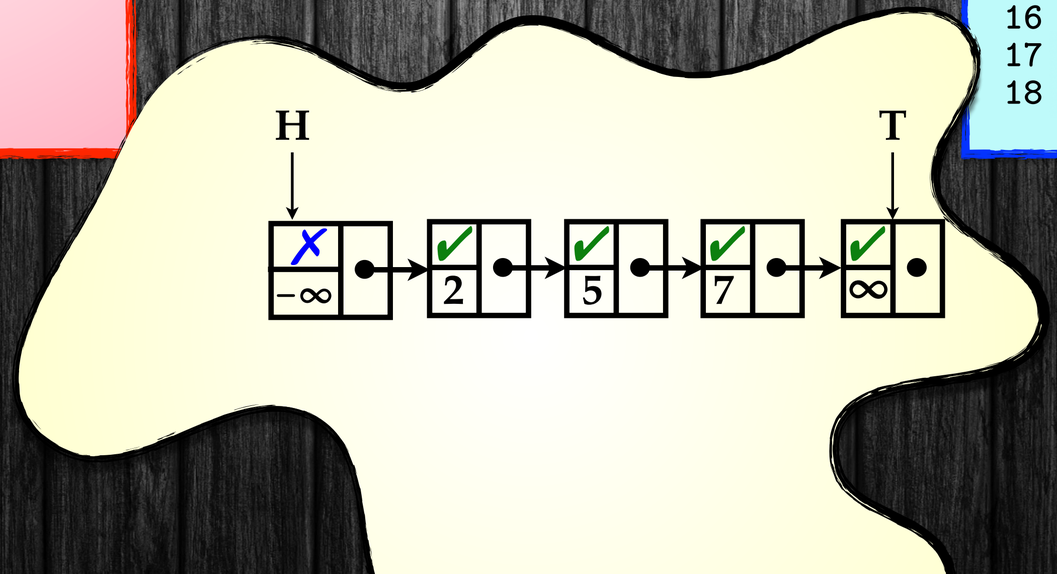
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



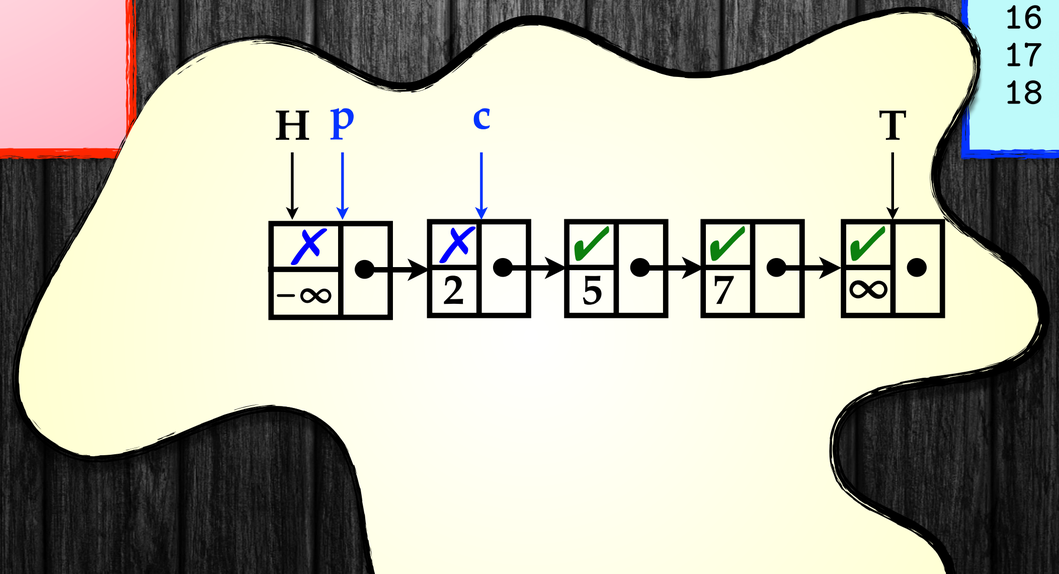
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



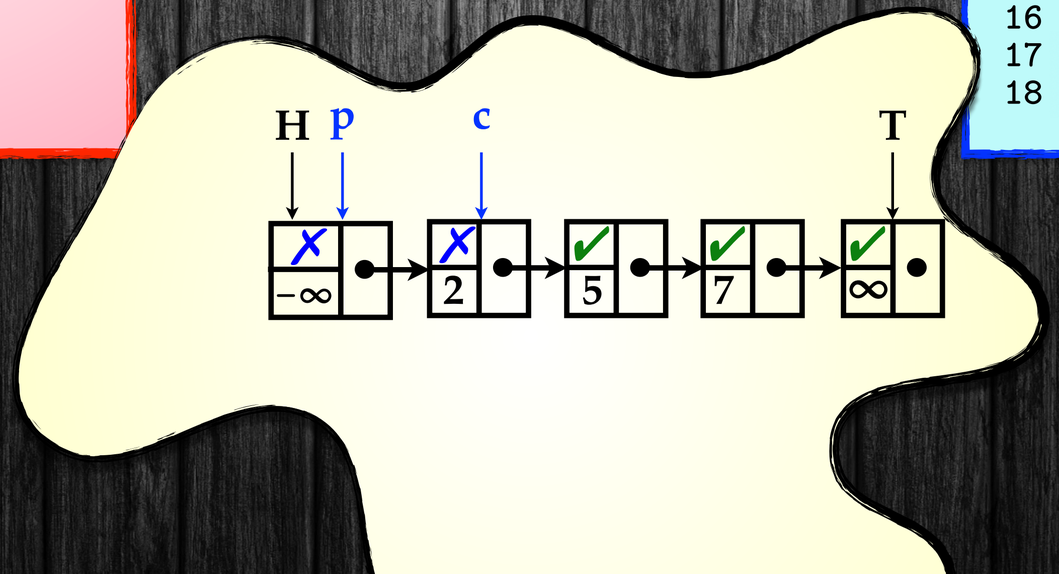
# Sets Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



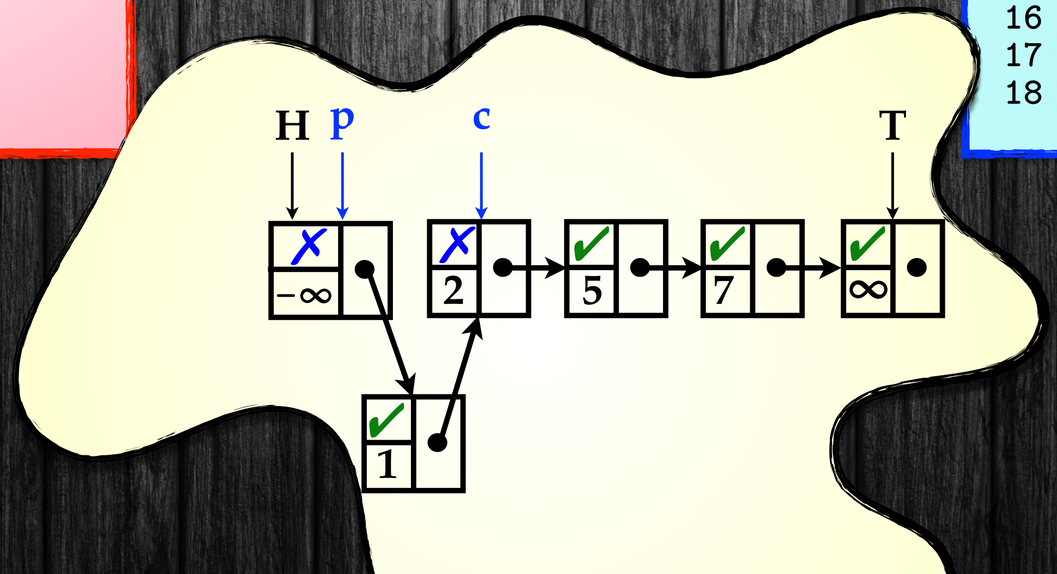
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



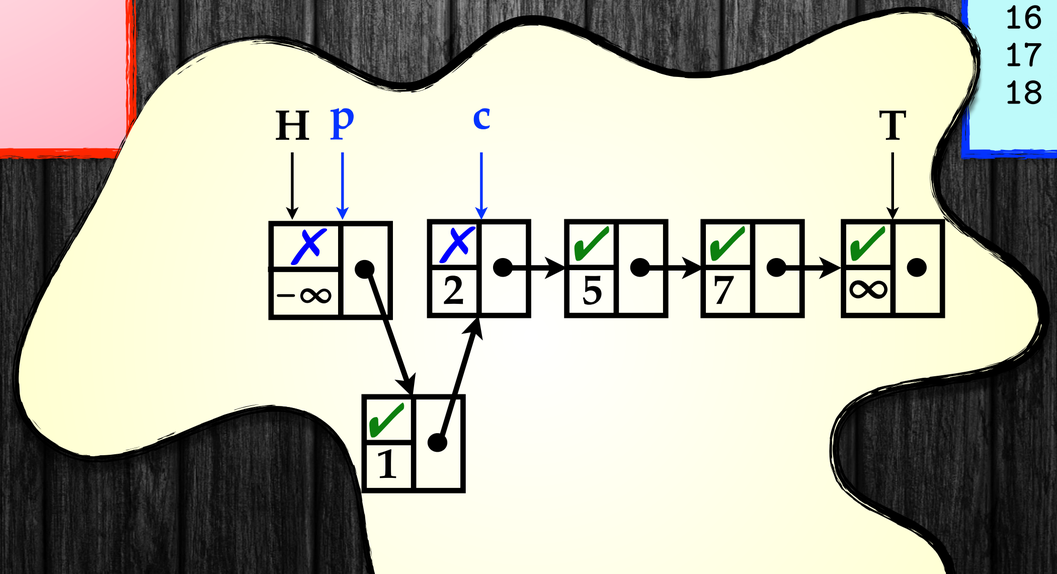
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



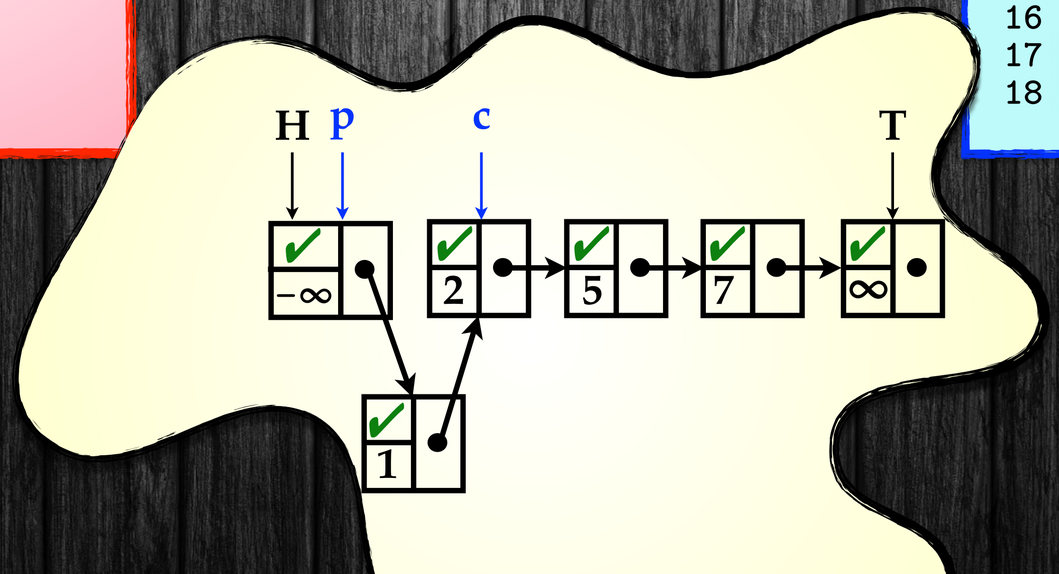
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



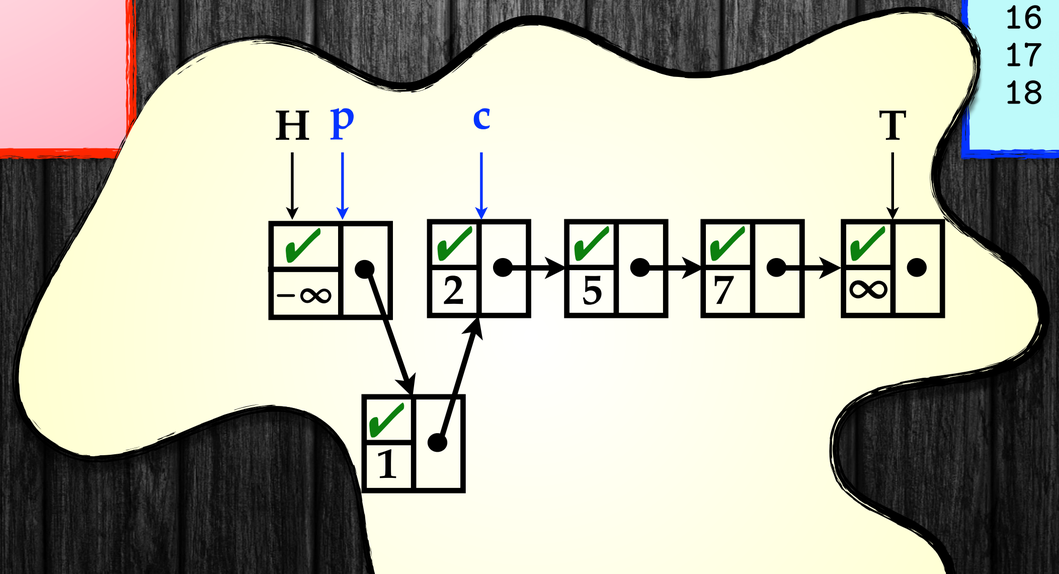
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



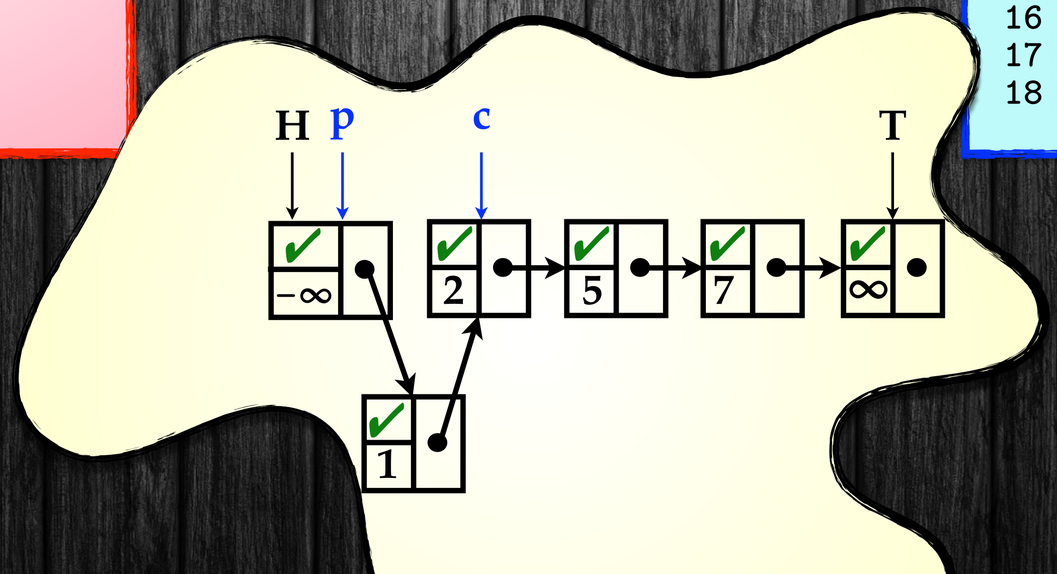
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



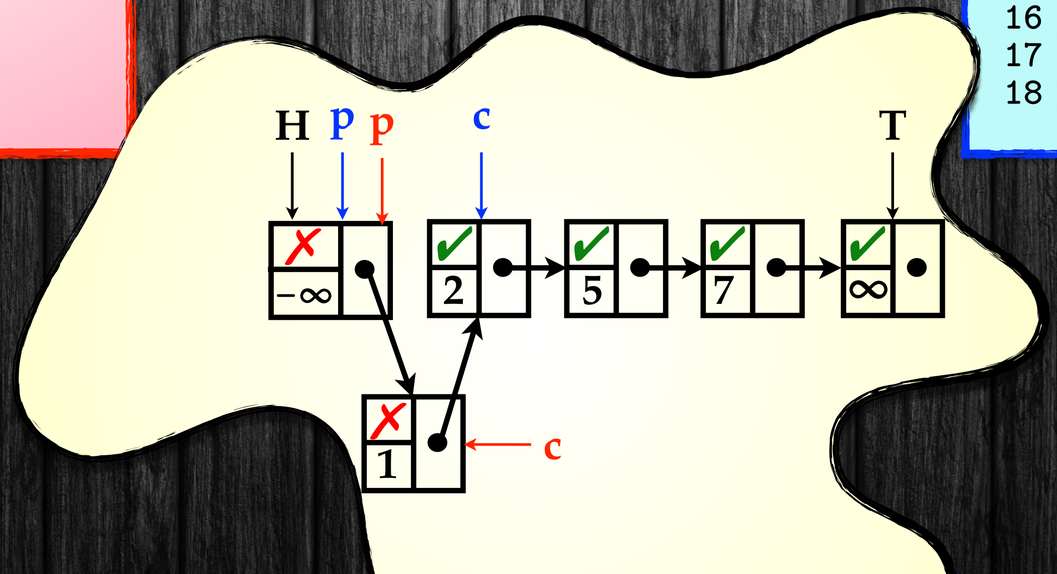
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



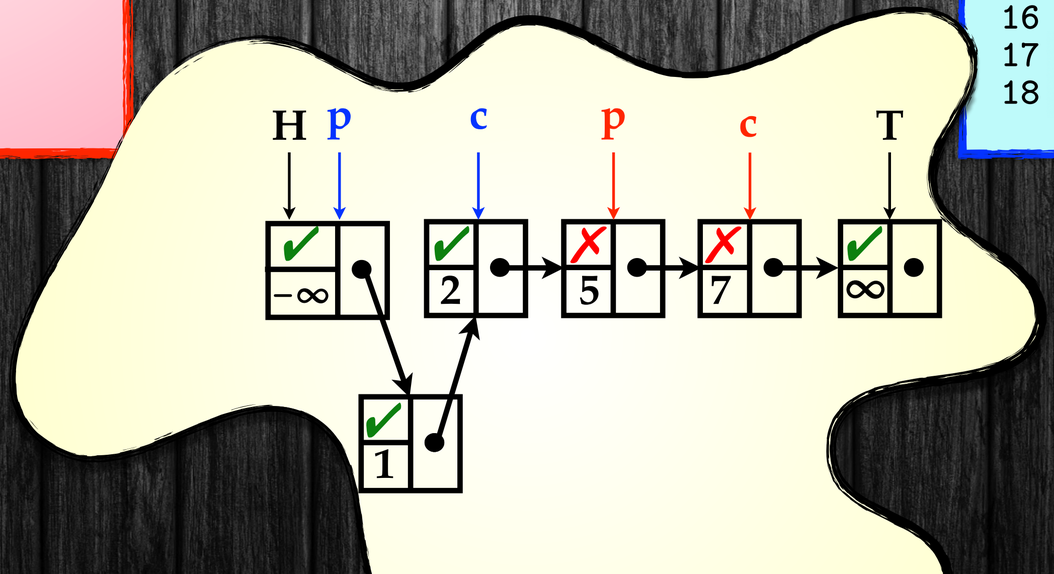
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



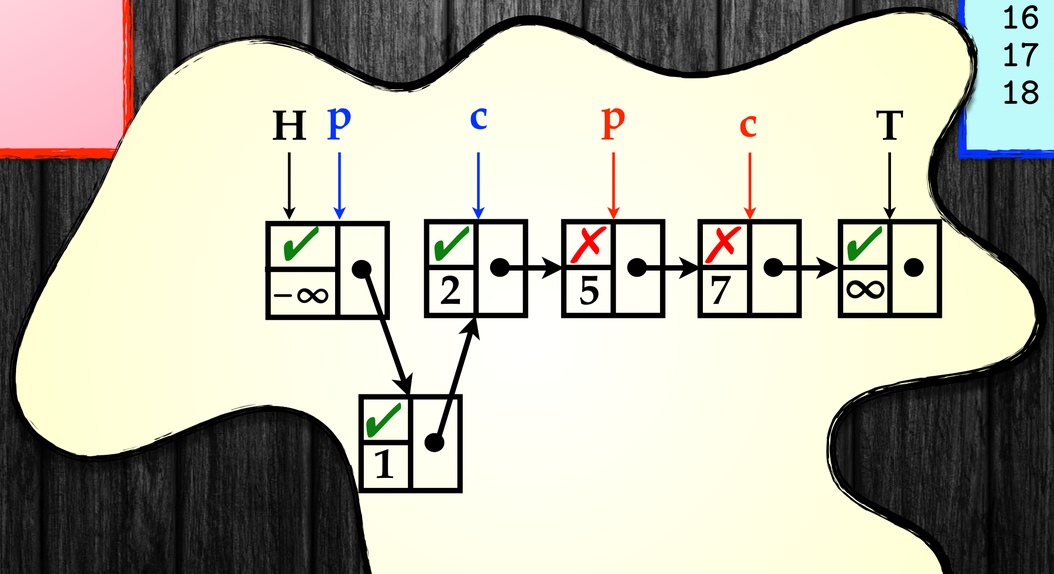
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



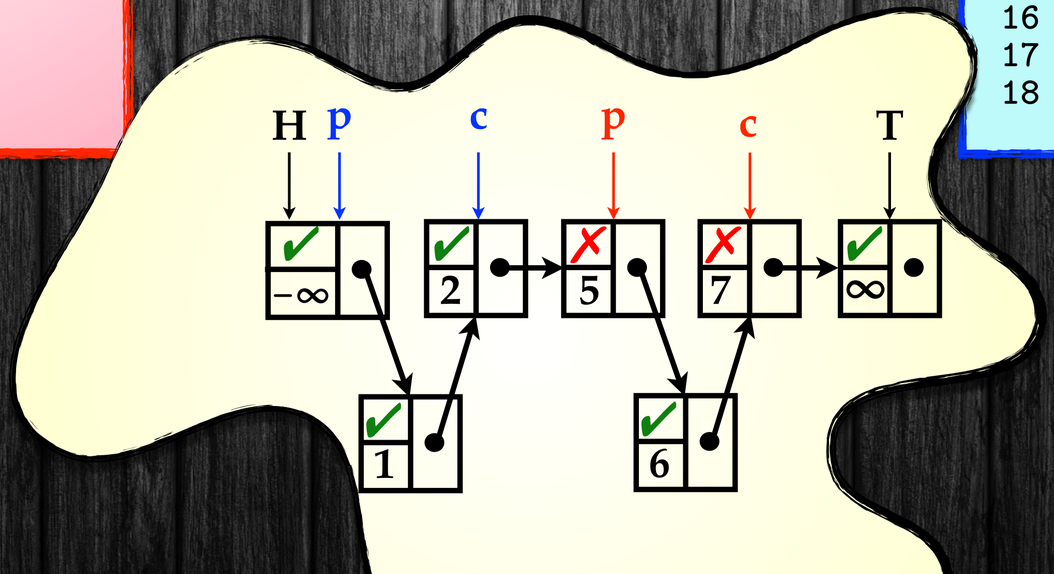
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



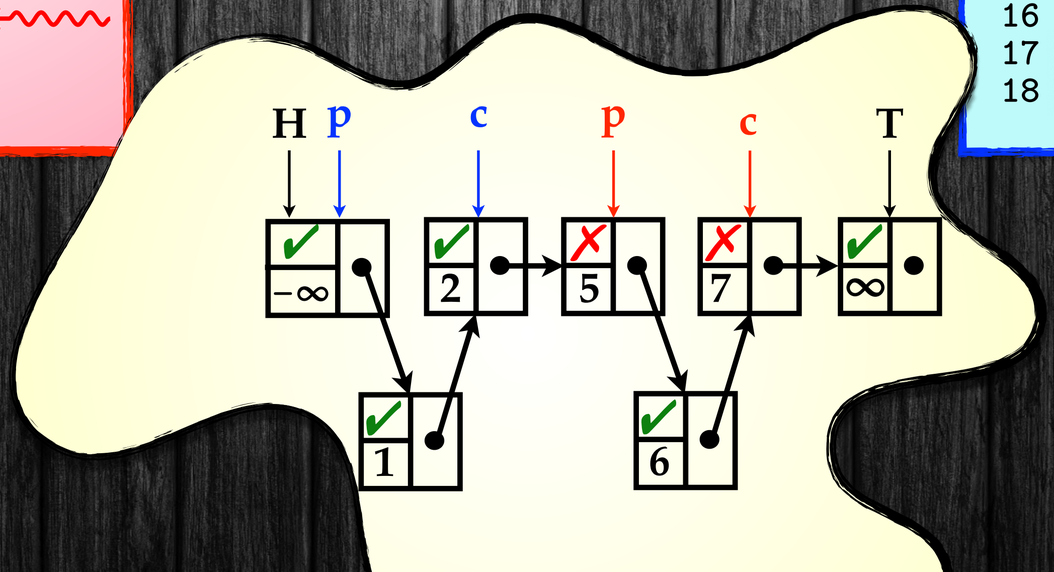
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



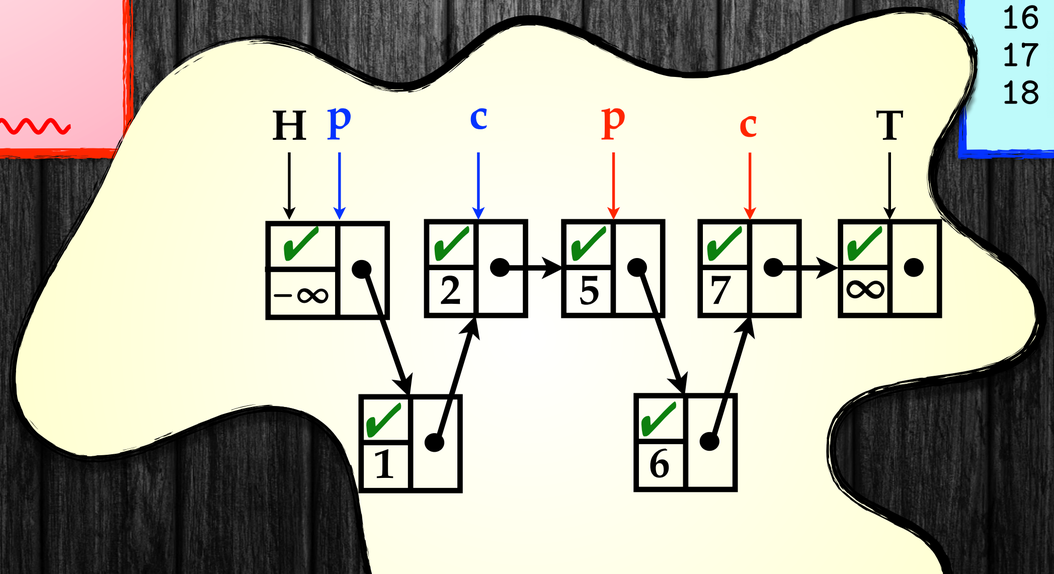
# Sets | Fine-Grained Synchronization

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```

add(6)

add(1)

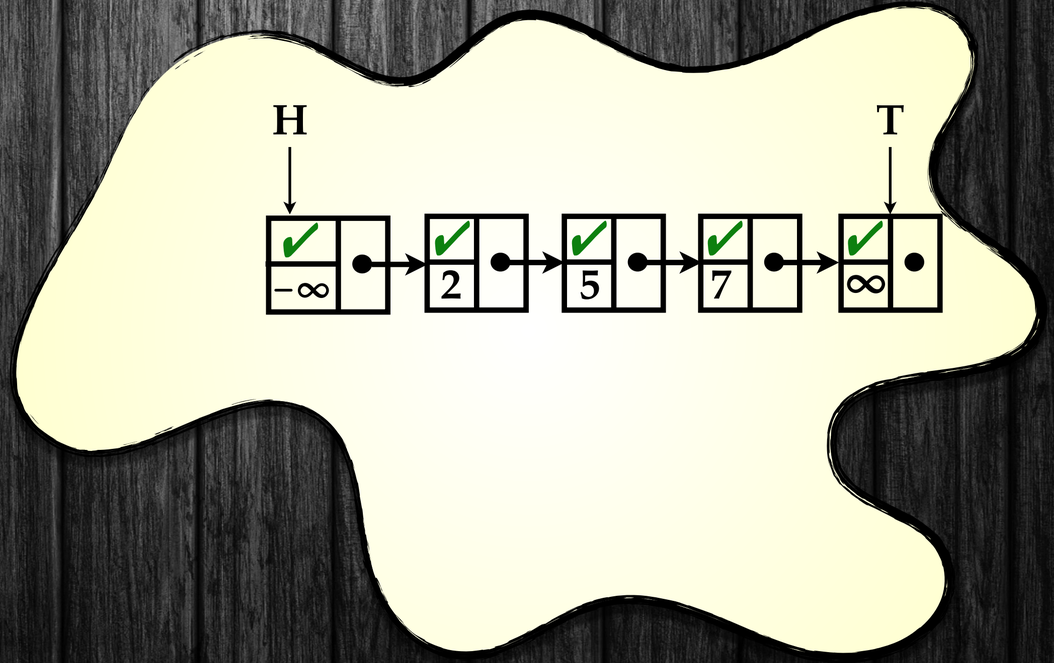
```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17 unlock(c)
18 unlock(p)
```



# Sets | Fine-Grained Synchronization

```
rmv(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    p.next=c.next;
12    return true
13  else
14    return false
15  unlock(c)
16  unlock(p)
```

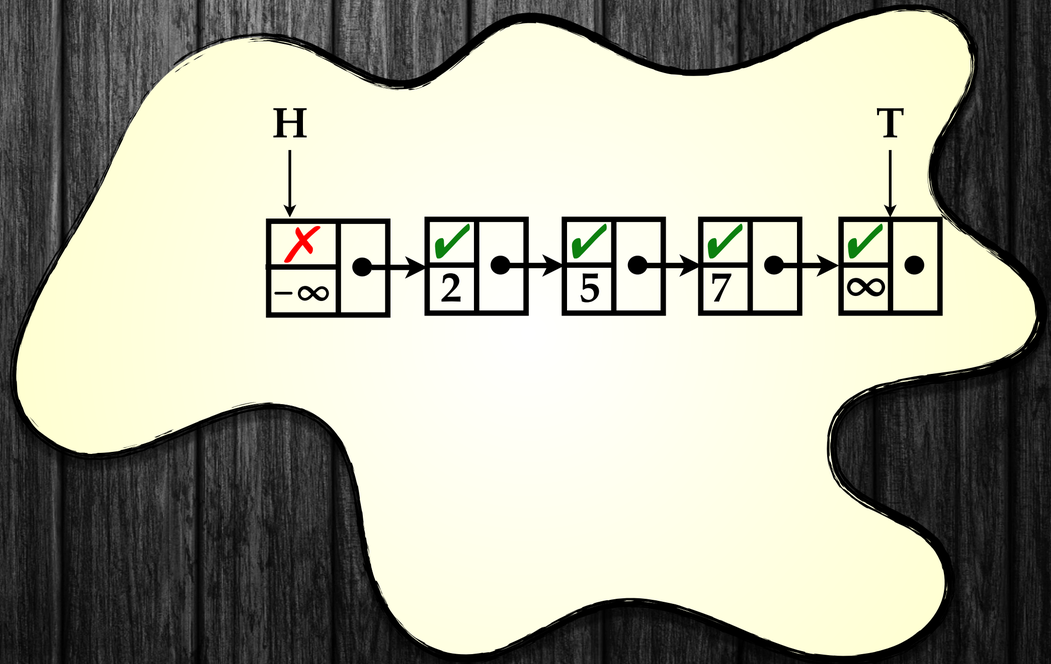


# Sets | Fine-Grained Synchronization

```
rmv(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   p.next=c.next;
12   return true
13 else
14   return false
15 unlock(c)
16 unlock(p)
```

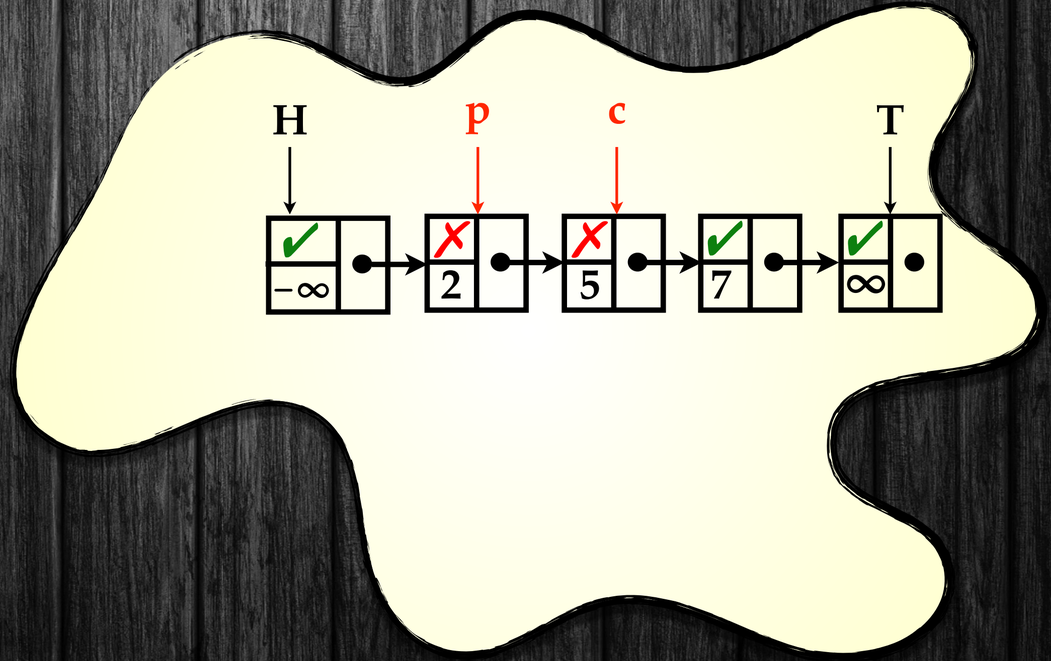


# Sets | Fine-Grained Synchronization

```
rmv(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   p.next=c.next;
12   return true
13 else
14   return false
15 unlock(c)
16 unlock(p)
```

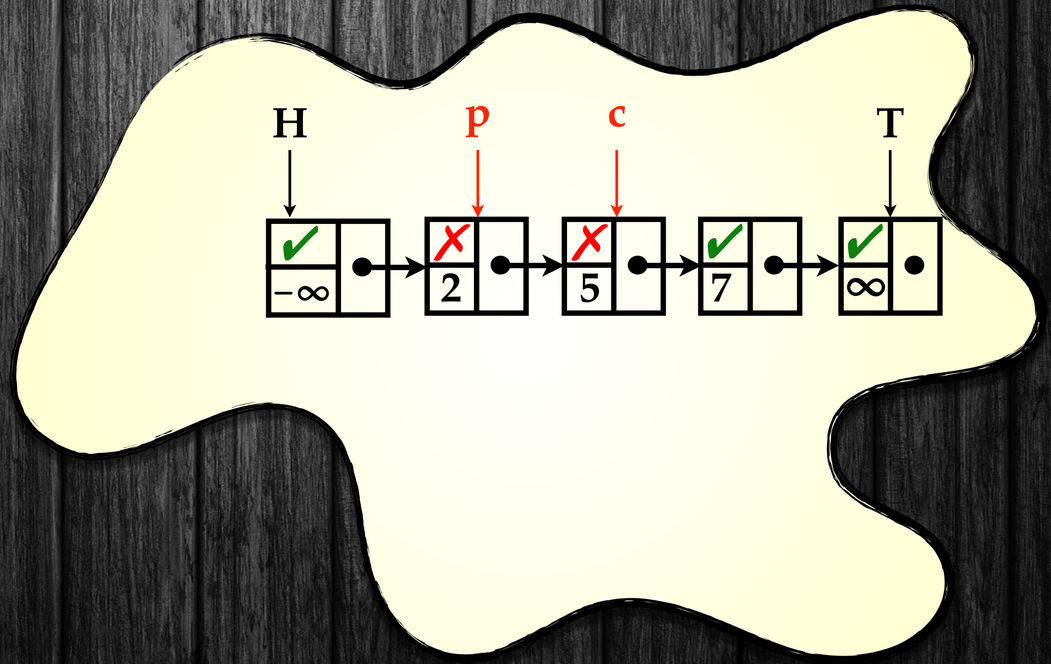


# Sets | Fine-Grained Synchronization

```
rmv(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   p.next=c.next;
12   return true
13 else
14   return false
15 unlock(c)
16 unlock(p)
```

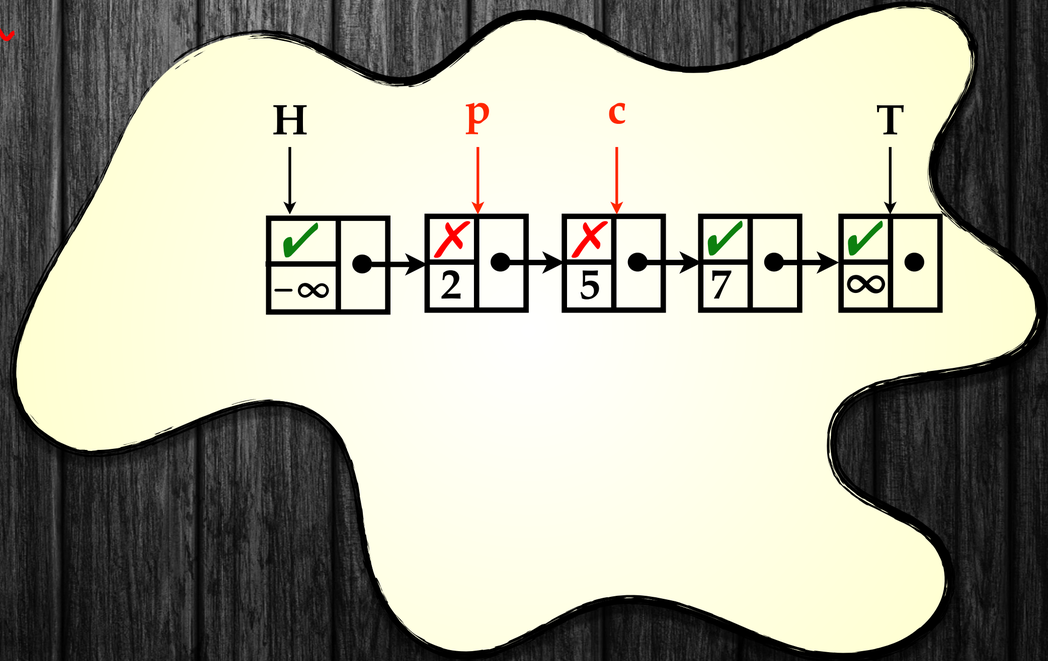


# Sets | Fine-Grained Synchronization

```
rmv(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   p.next=c.next;
12   return true
13 else
14   return false
15 unlock(c)
16 unlock(p)
```

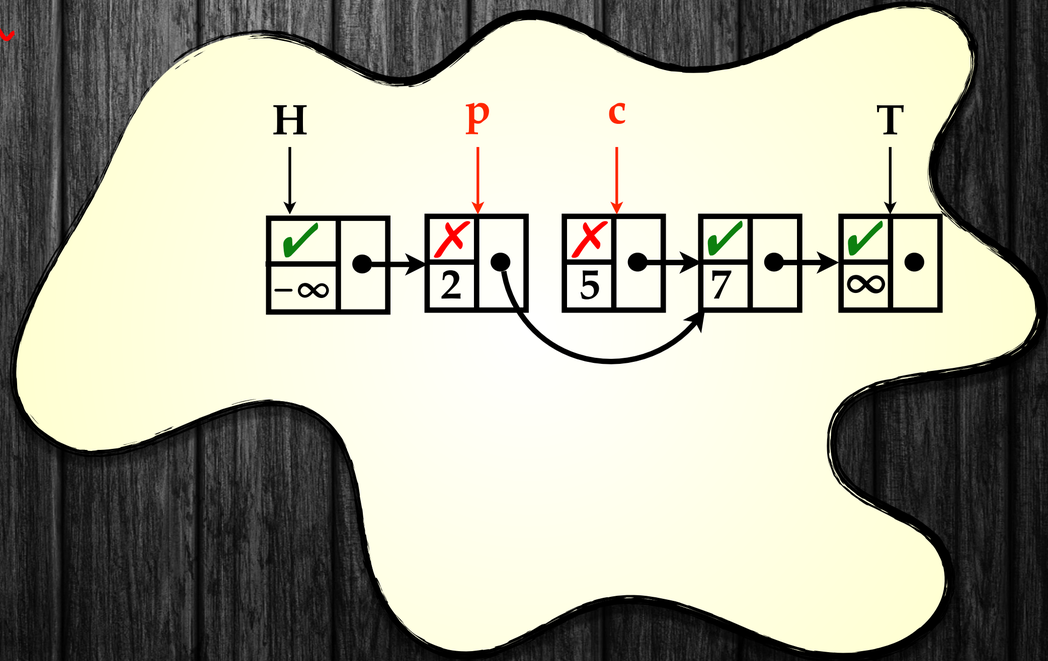


# Sets | Fine-Grained Synchronization

```
rmv(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    p.next=c.next;
12    return true
13  else
14    return false
15  unlock(c)
16  unlock(p)
```

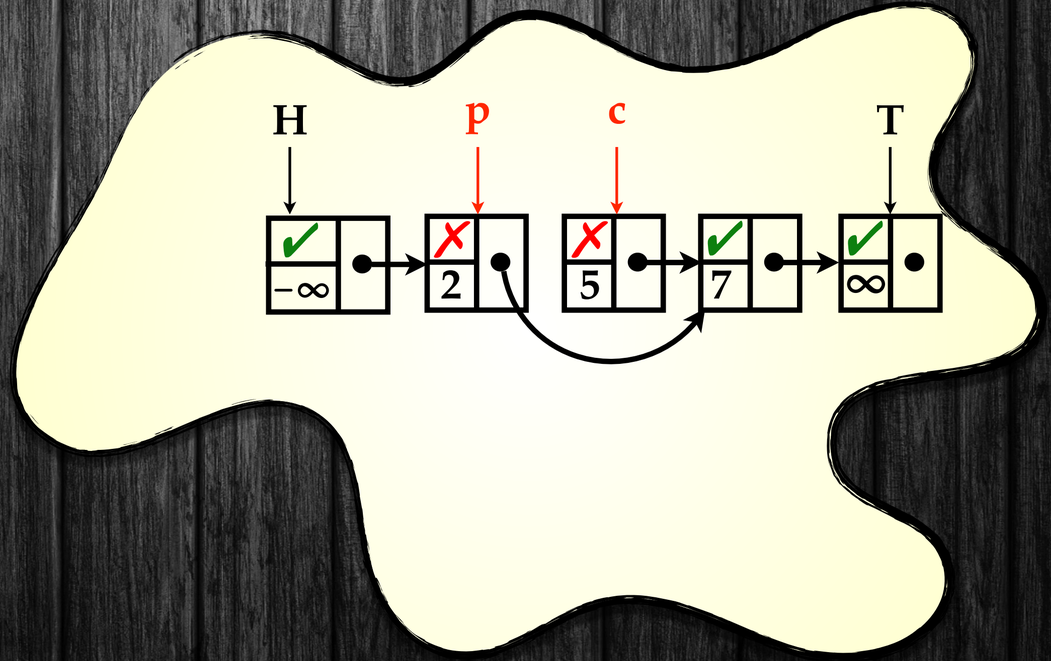


# Sets | Fine-Grained Synchronization

```
rmv(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   p.next=c.next;
12   return true
13 else
14   return false
15 unlock(c)
16 unlock(p)
```

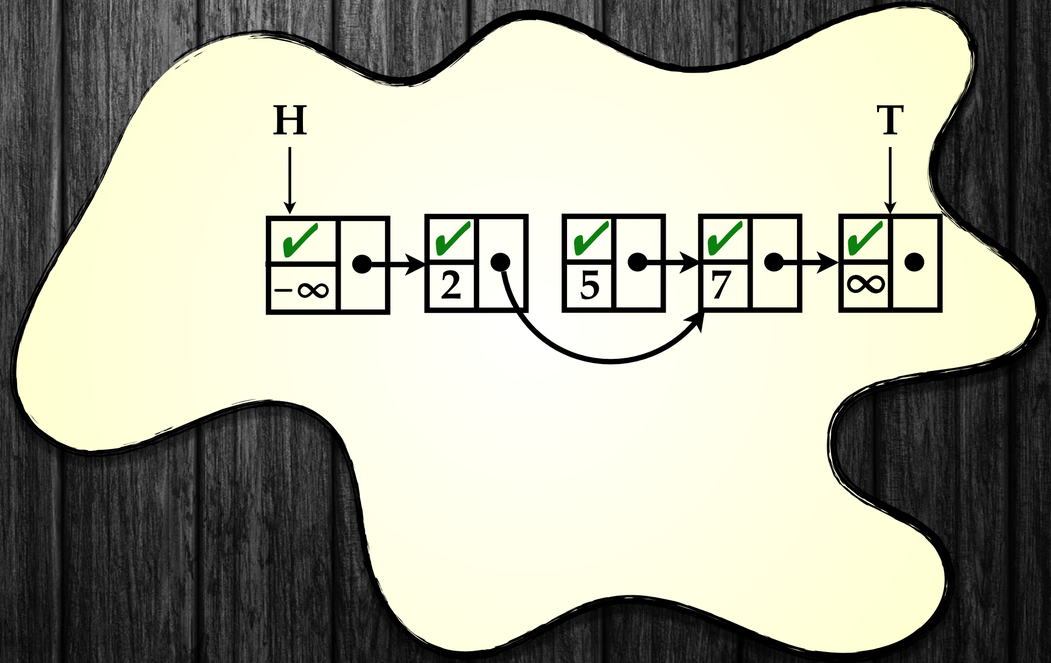


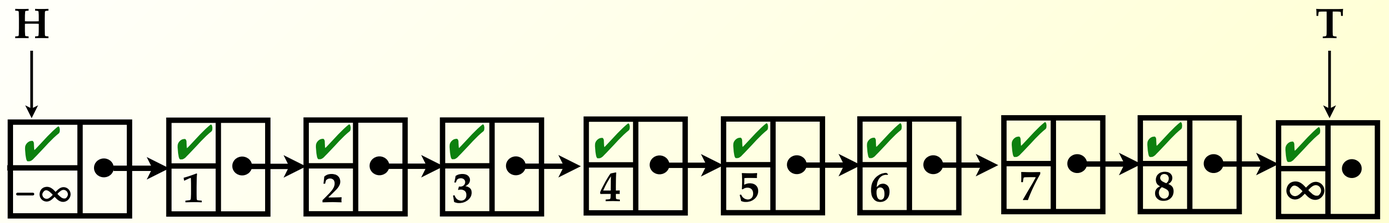
# Sets | Fine-Grained Synchronization

```
rmv(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   p.next=c.next;
12   return true
13 else
14   return false
15 unlock(c)
16 unlock(p)
```



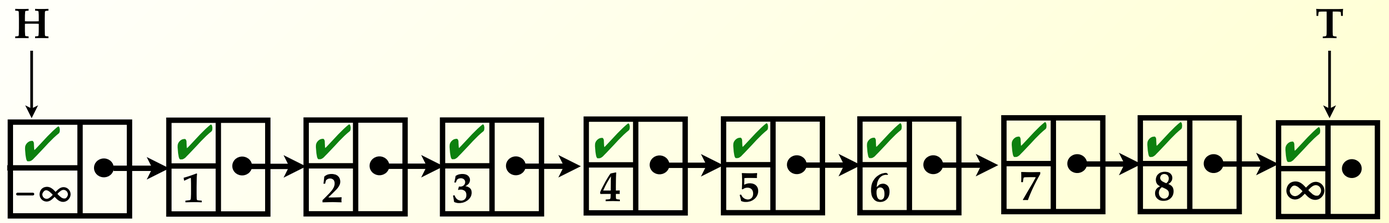



# Sets Fine-Grained Synchronization



# Sets | Fine-Grained Synchronization

**Lock Coupling**

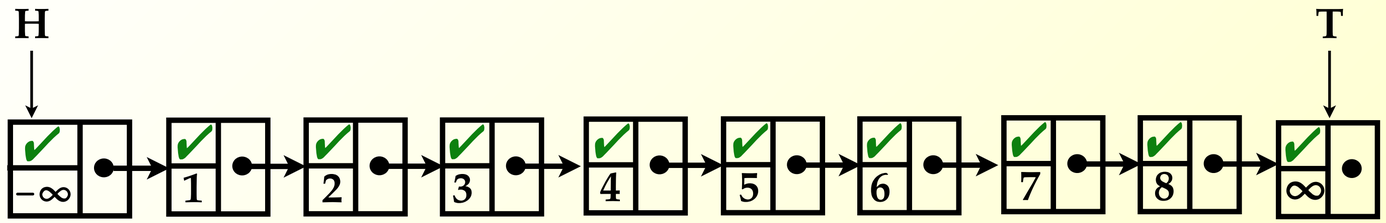


# Sets Fine-Grained Synchronization



rmv(4)

**Lock Coupling**

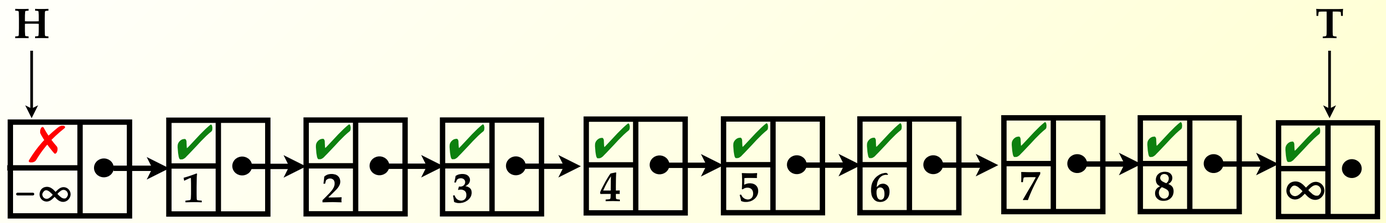


# Sets Fine-Grained Synchronization



rmv(4)

**Lock Coupling**

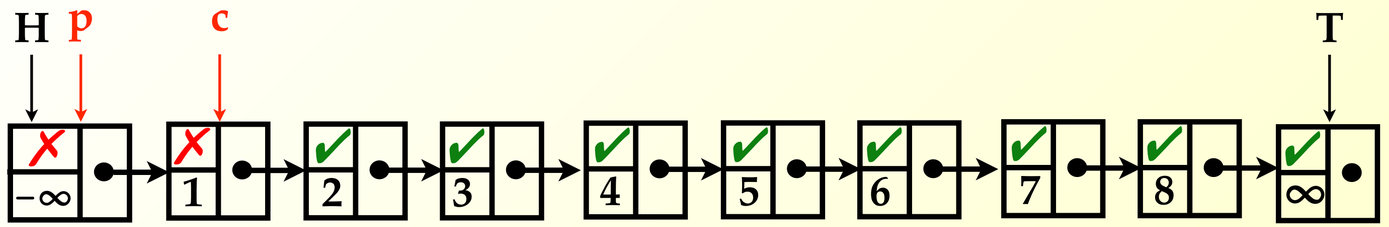


# Sets Fine-Grained Synchronization



rmv(4)

**Lock Coupling**

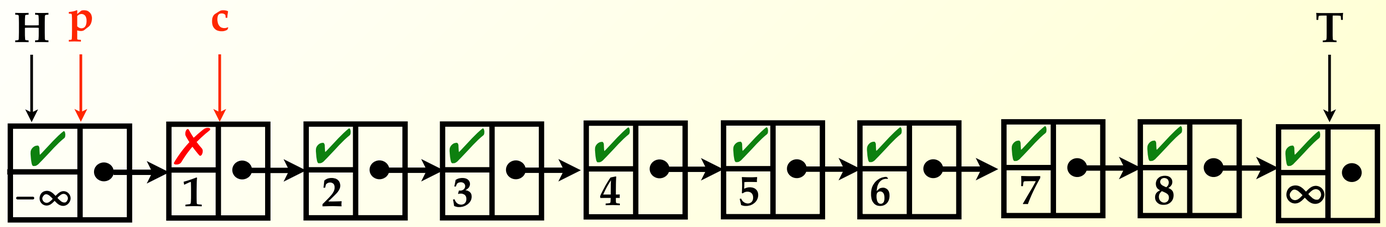


# Sets Fine-Grained Synchronization



rmv(4)

**Lock Coupling**

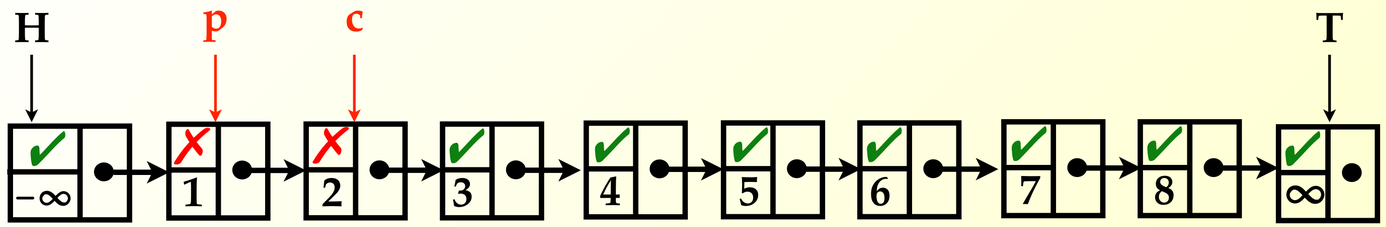


# Sets | Fine-Grained Synchronization



rmv(4)

**Lock Coupling**

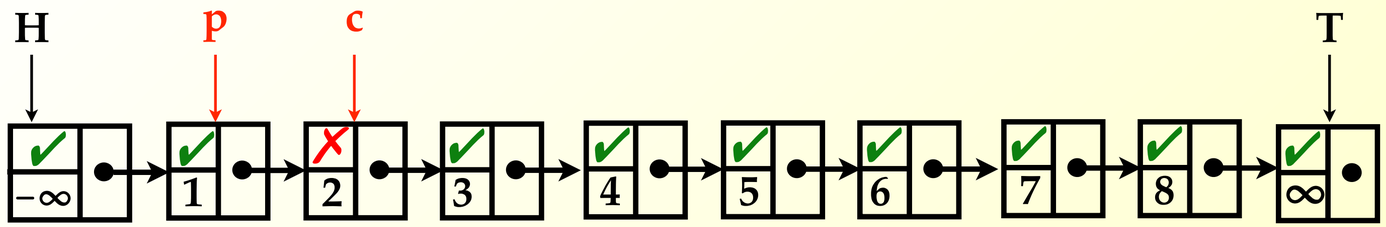


# Sets | Fine-Grained Synchronization



rmv(4)

**Lock Coupling**

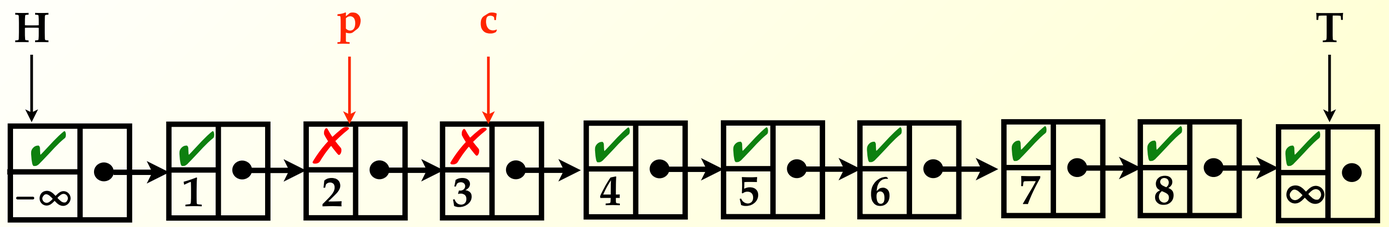


# Sets Fine-Grained Synchronization



rmv(4)

**Lock Coupling**

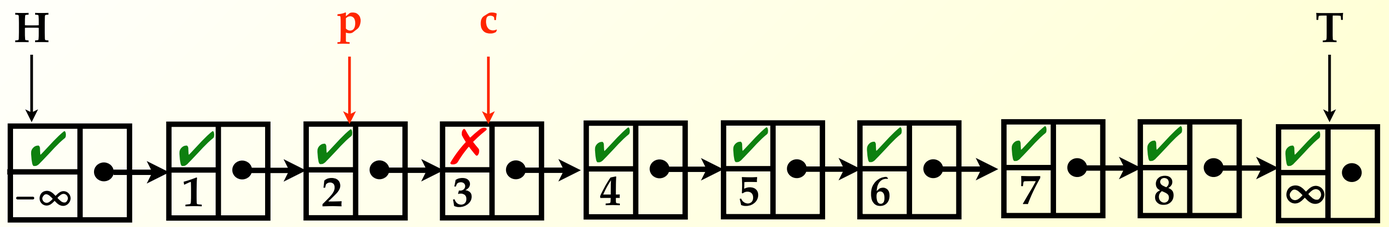


# Sets Fine-Grained Synchronization



rmv(4)

**Lock Coupling**

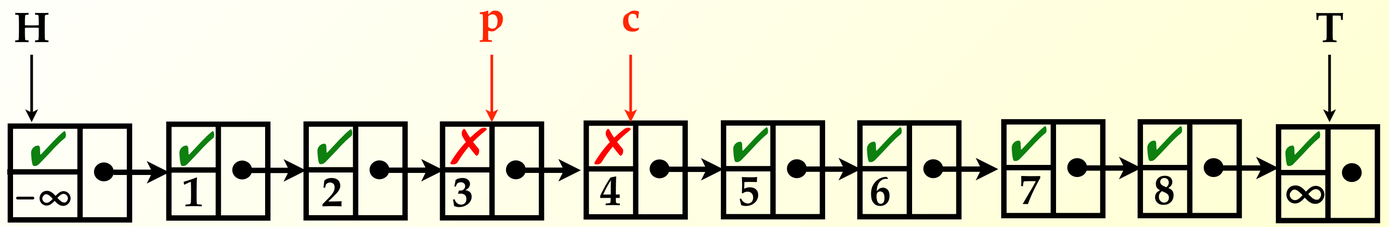


# Sets | Fine-Grained Synchronization



rmv(4)

**Lock Coupling**

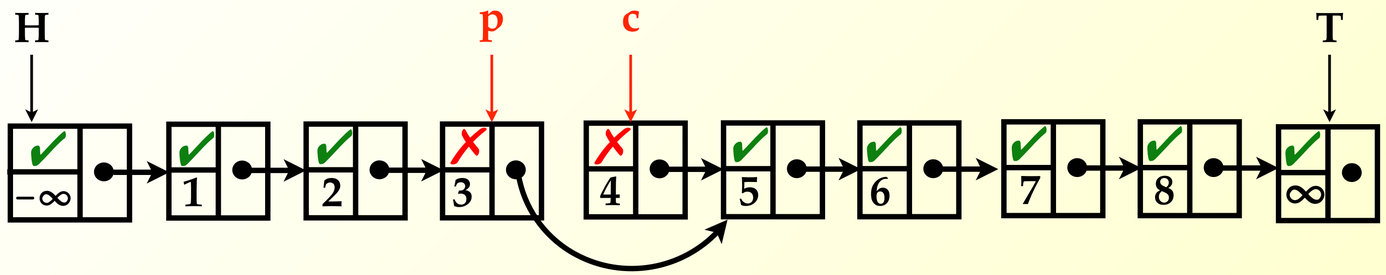


# Sets | Fine-Grained Synchronization



rmv(4)

**Lock Coupling**

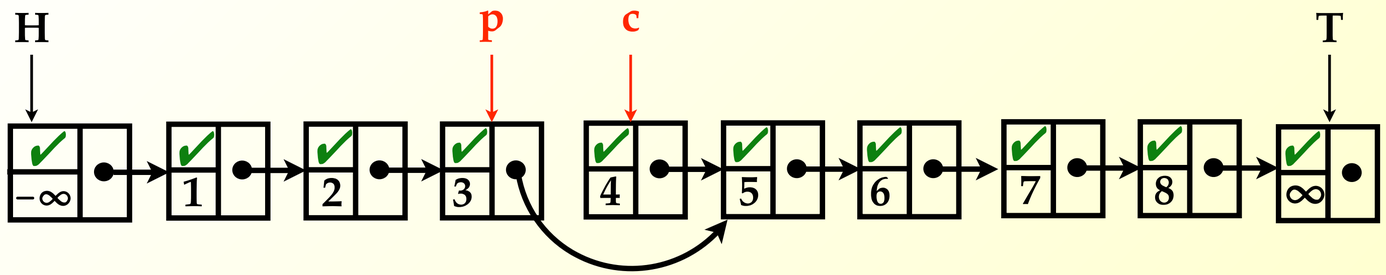


# Sets Fine-Grained Synchronization

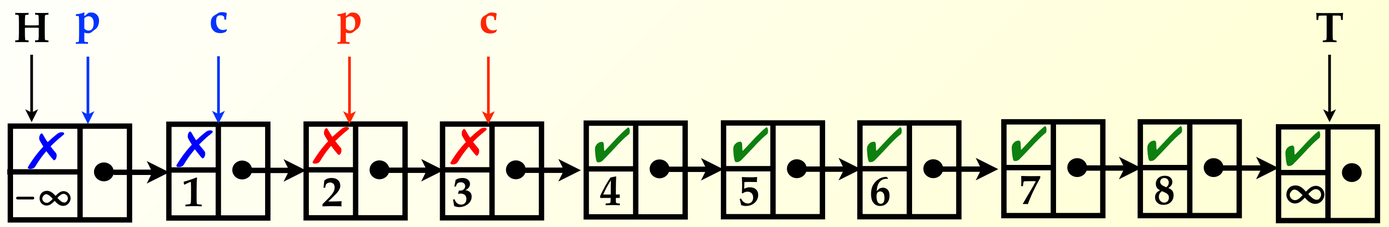


rmv(4)

**Lock Coupling**

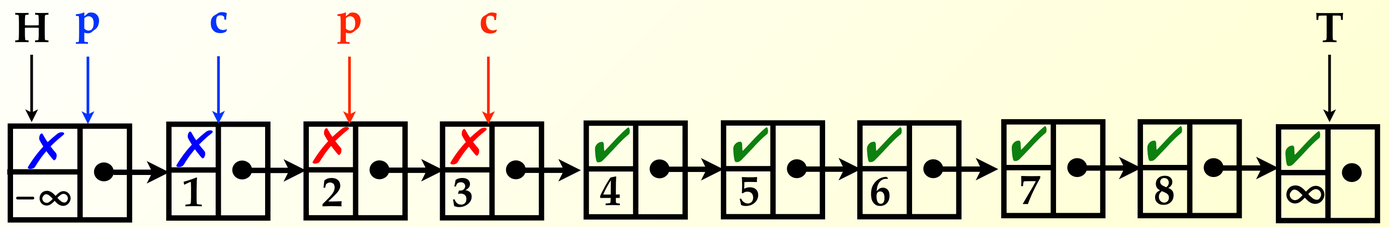


# Sets Fine-Grained Synchronization



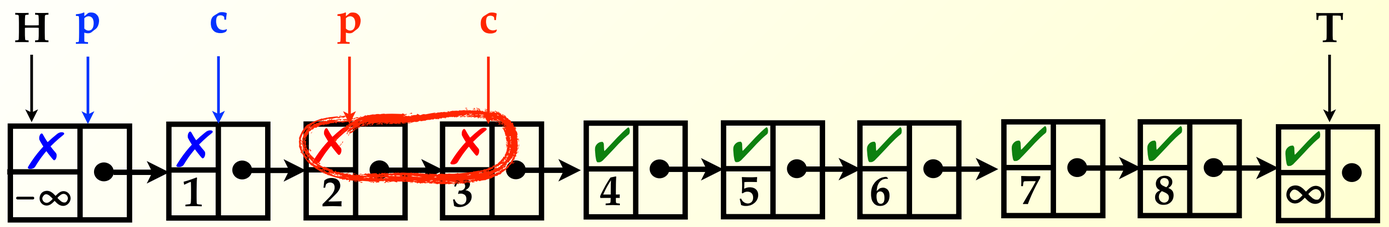
# Sets | Fine-Grained Synchronization

**Lock Coupling**



# Sets | Fine-Grained Synchronization

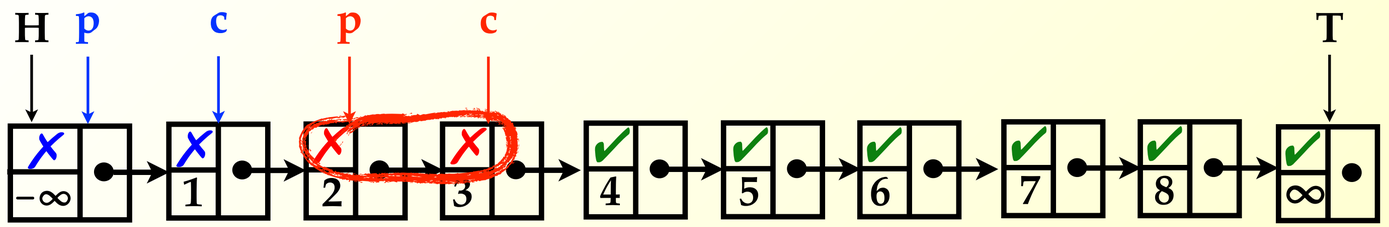
## Lock Coupling



# Sets Fine-Grained Synchronization

threads cannot overtake each other

Lock Coupling

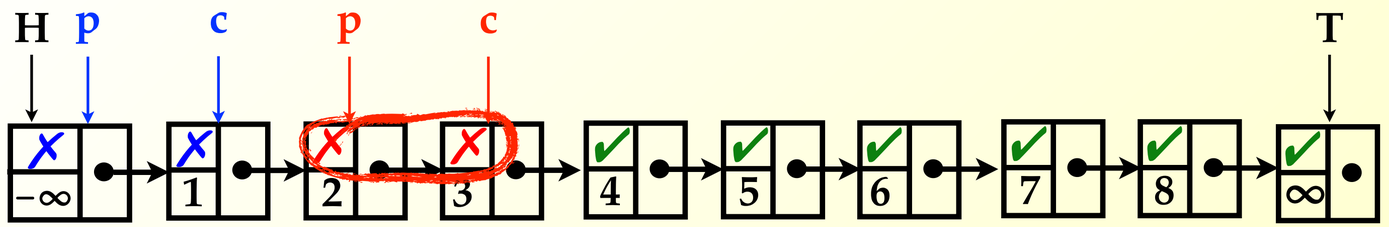


# Sets | Fine-Grained Synchronization

threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

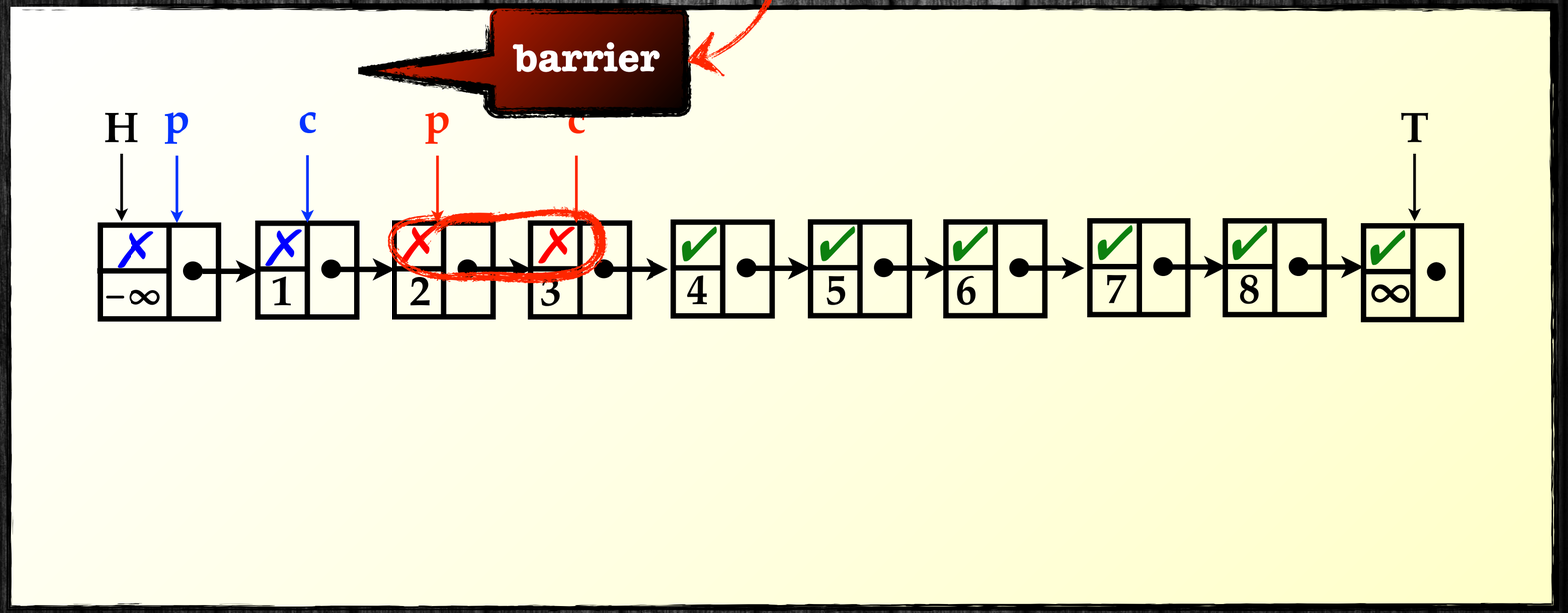


# Sets | Fine-Grained Synchronization

threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

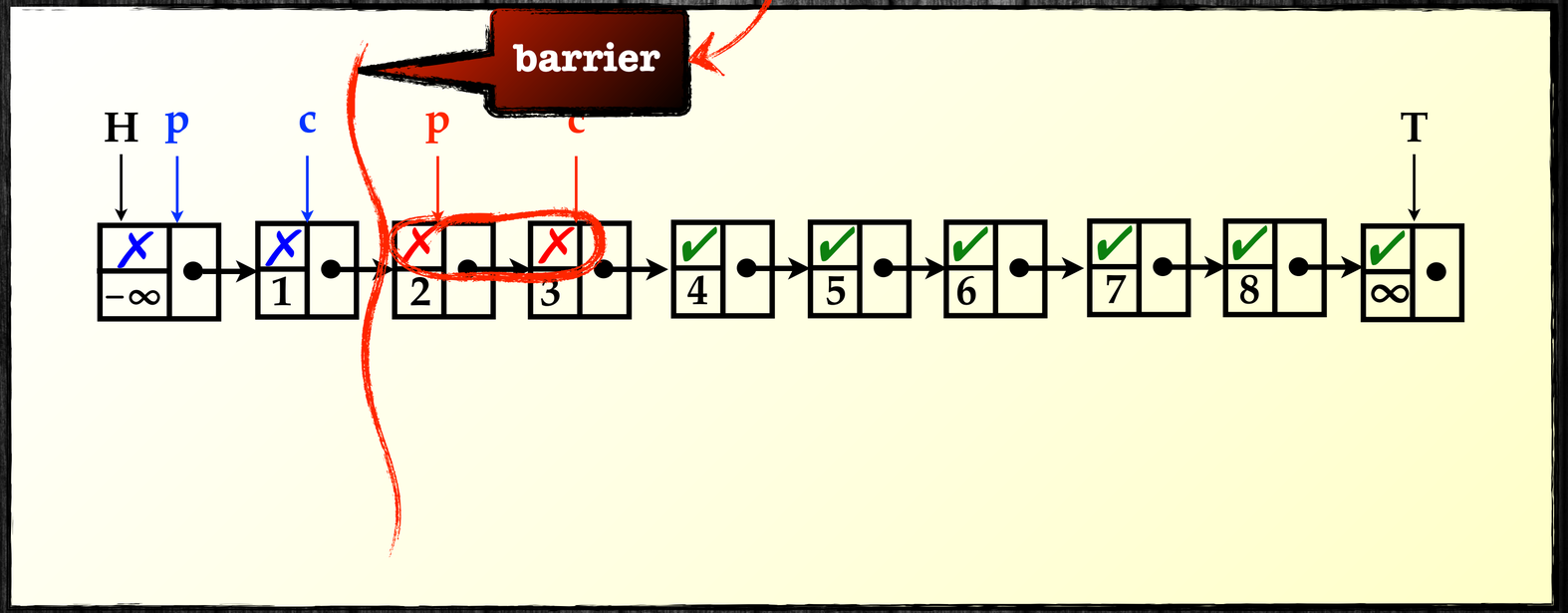


# Sets Fine-Grained Synchronization

threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

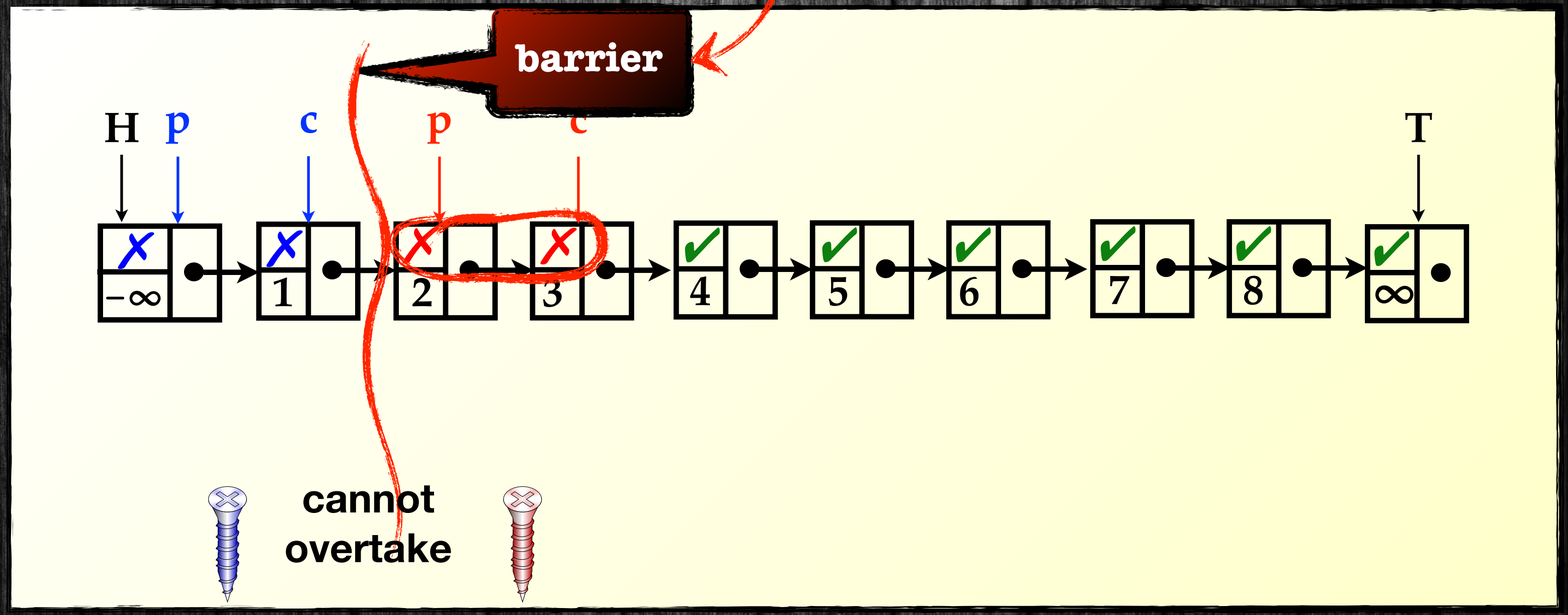


# Sets Fine-Grained Synchronization

threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

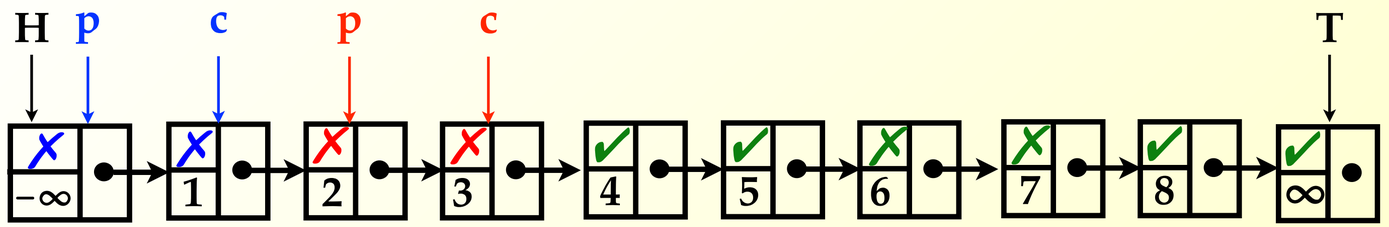


# Sets Fine-Grained Synchronization

threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

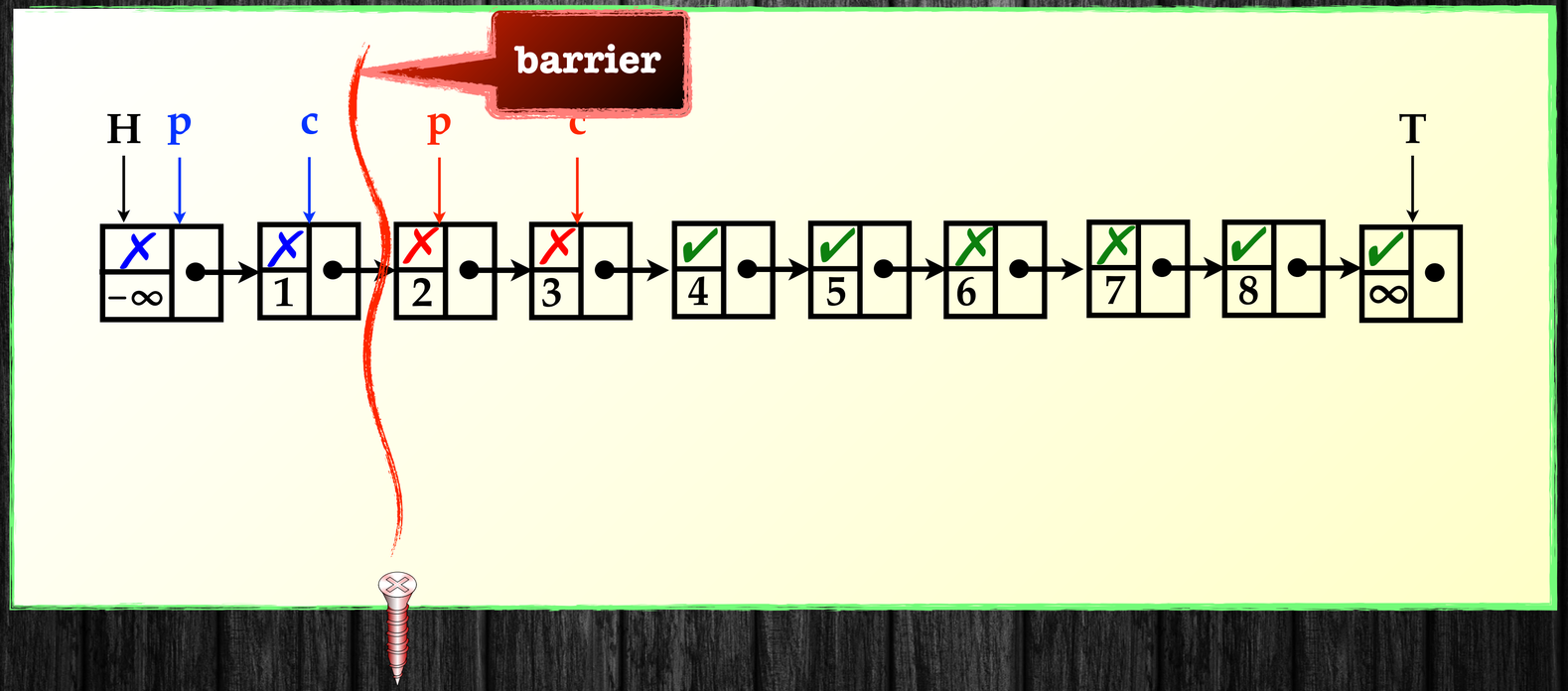


# Sets Fine-Grained Synchronization

threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

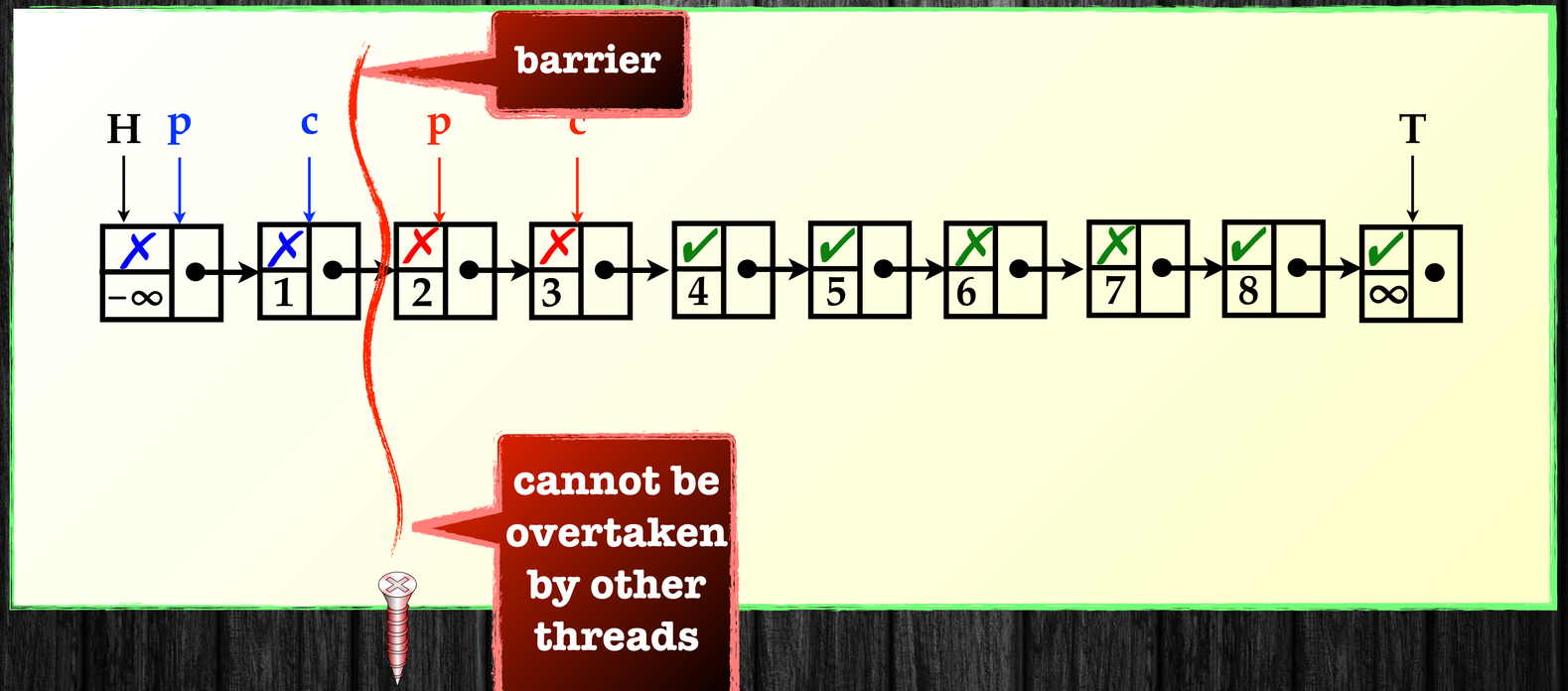


# Sets | Fine-Grained Synchronization

threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

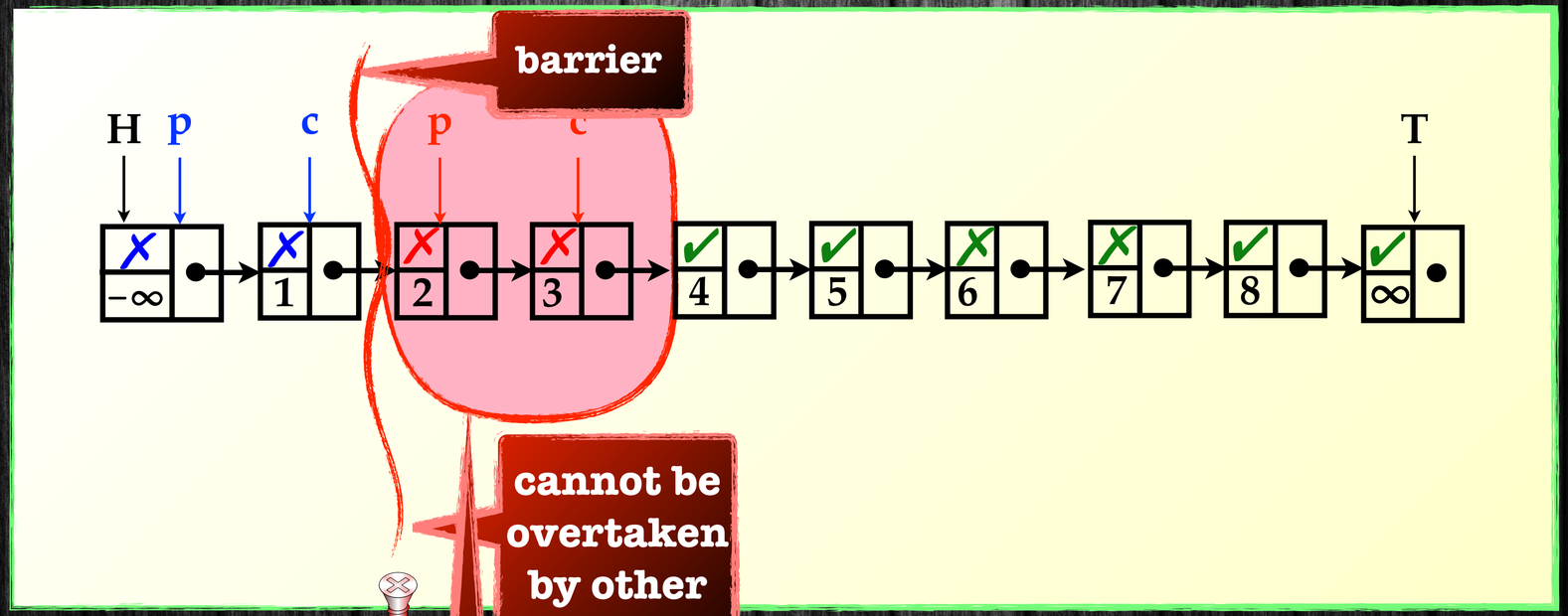


# Sets | Fine-Grained Synchronization

threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

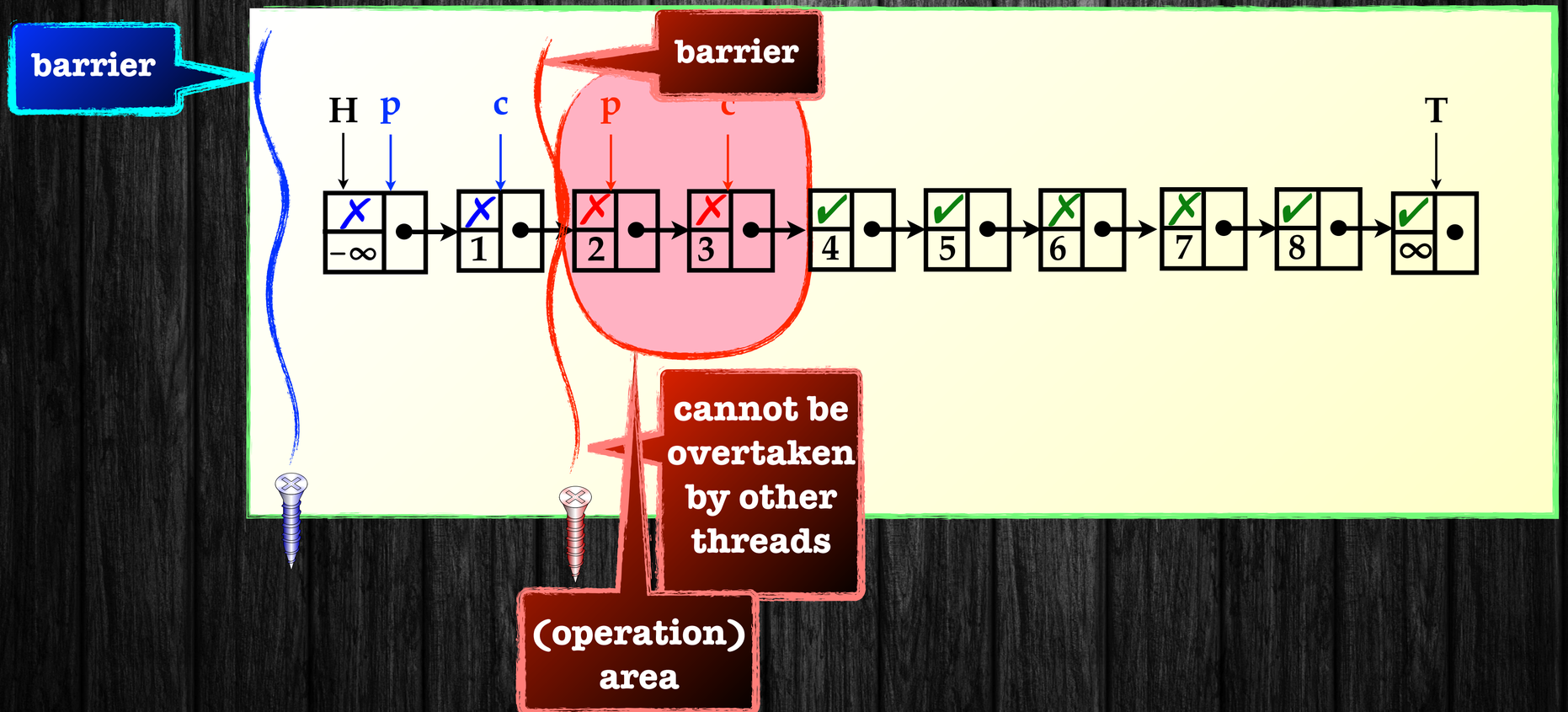


# Sets | Fine-Grained Synchronization

threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

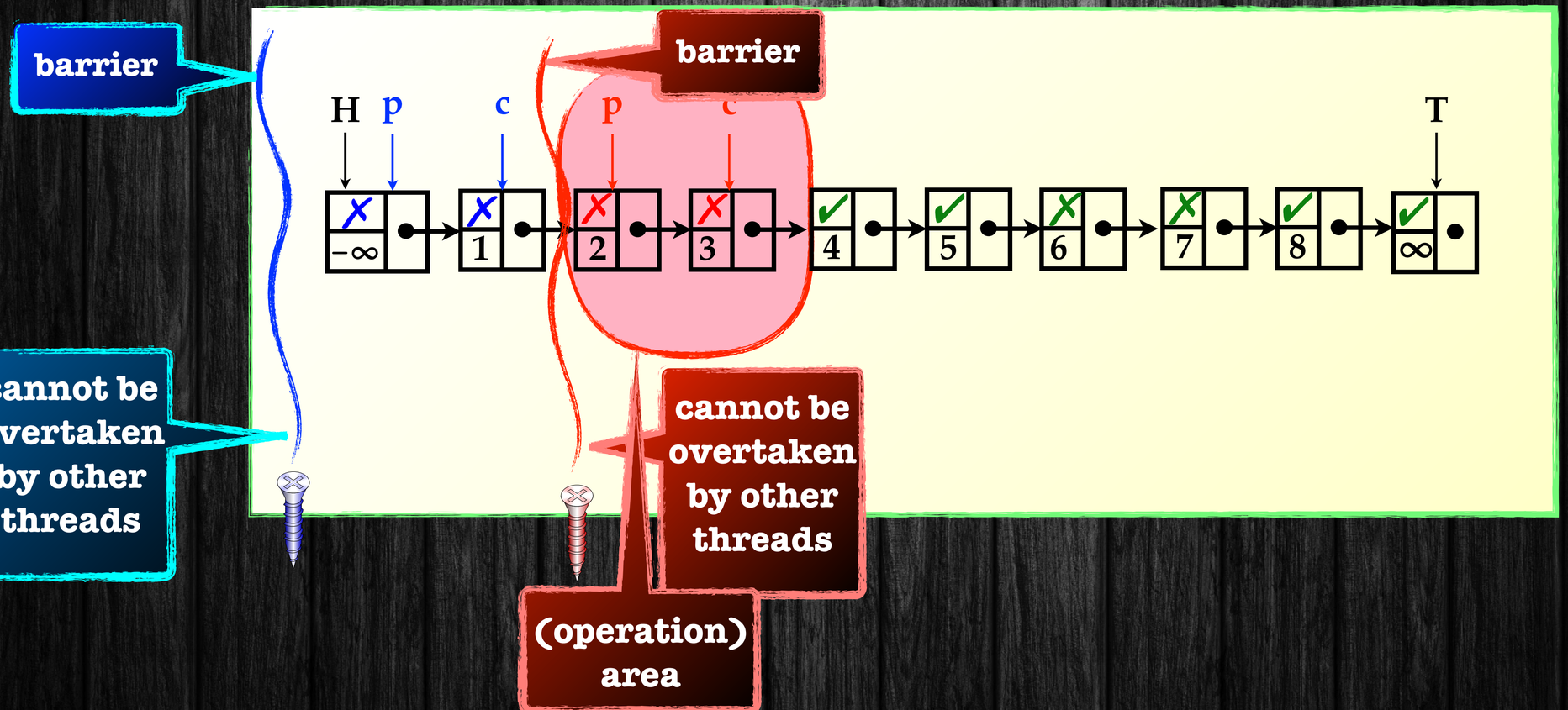


# Sets | Fine-Grained Synchronization

threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

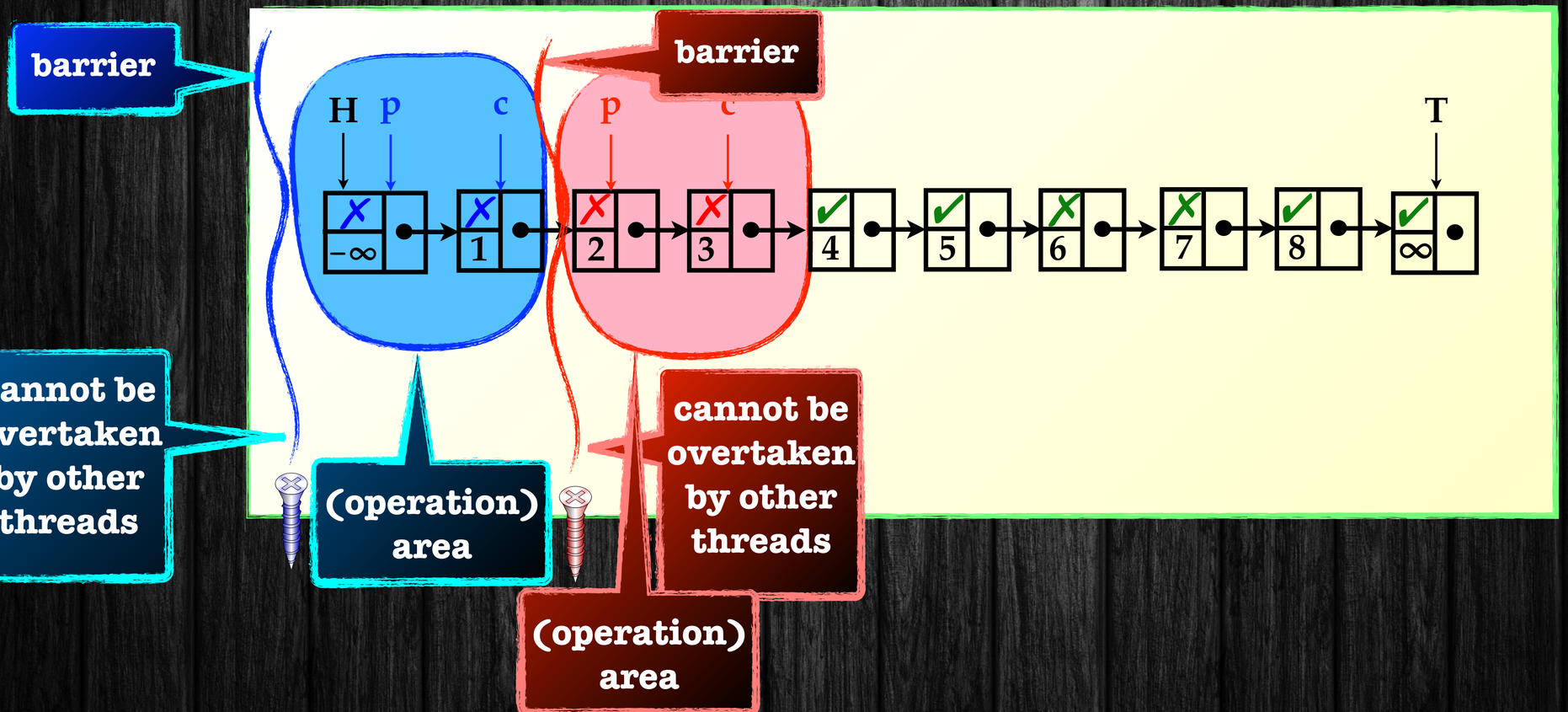


# Sets | Fine-Grained Synchronization

threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

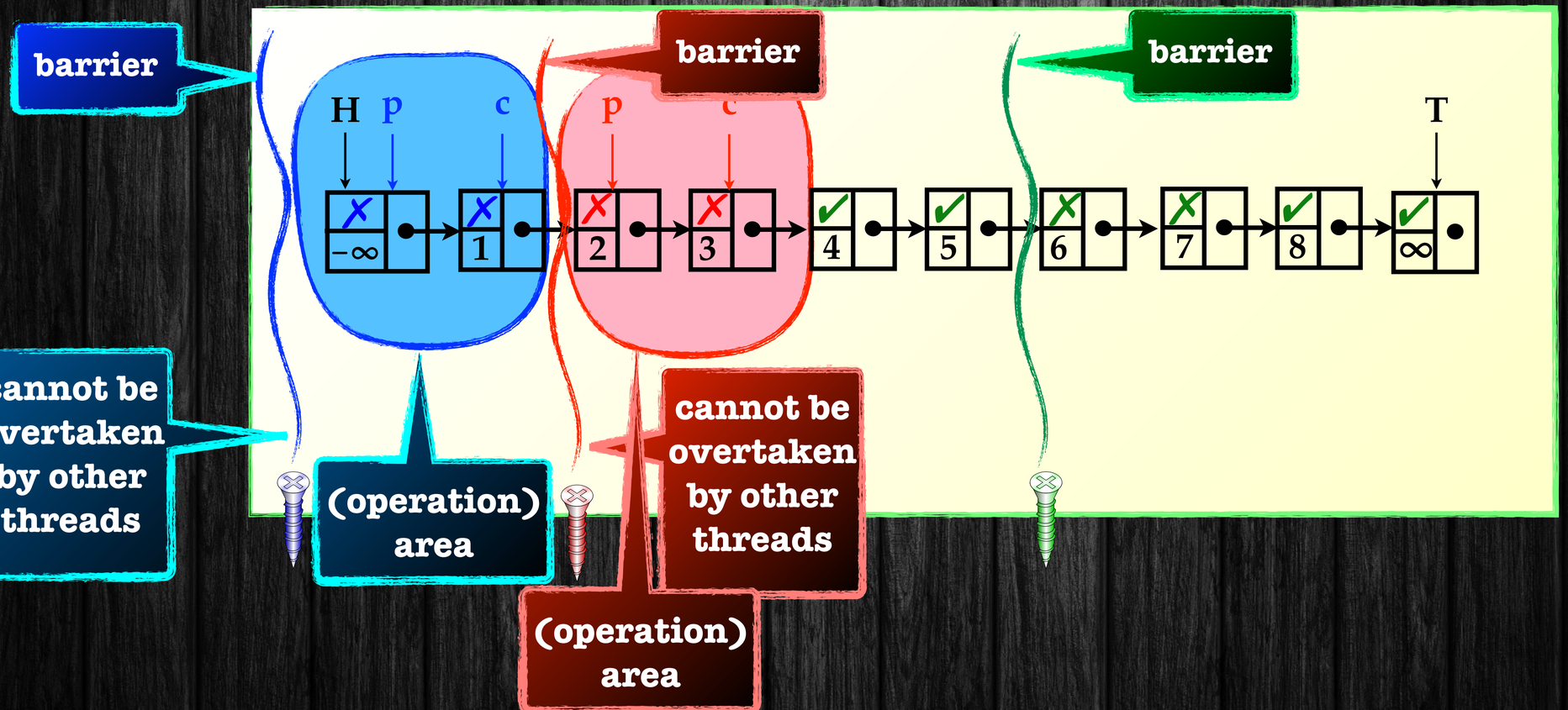


# Sets | Fine-Grained Synchronization

threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

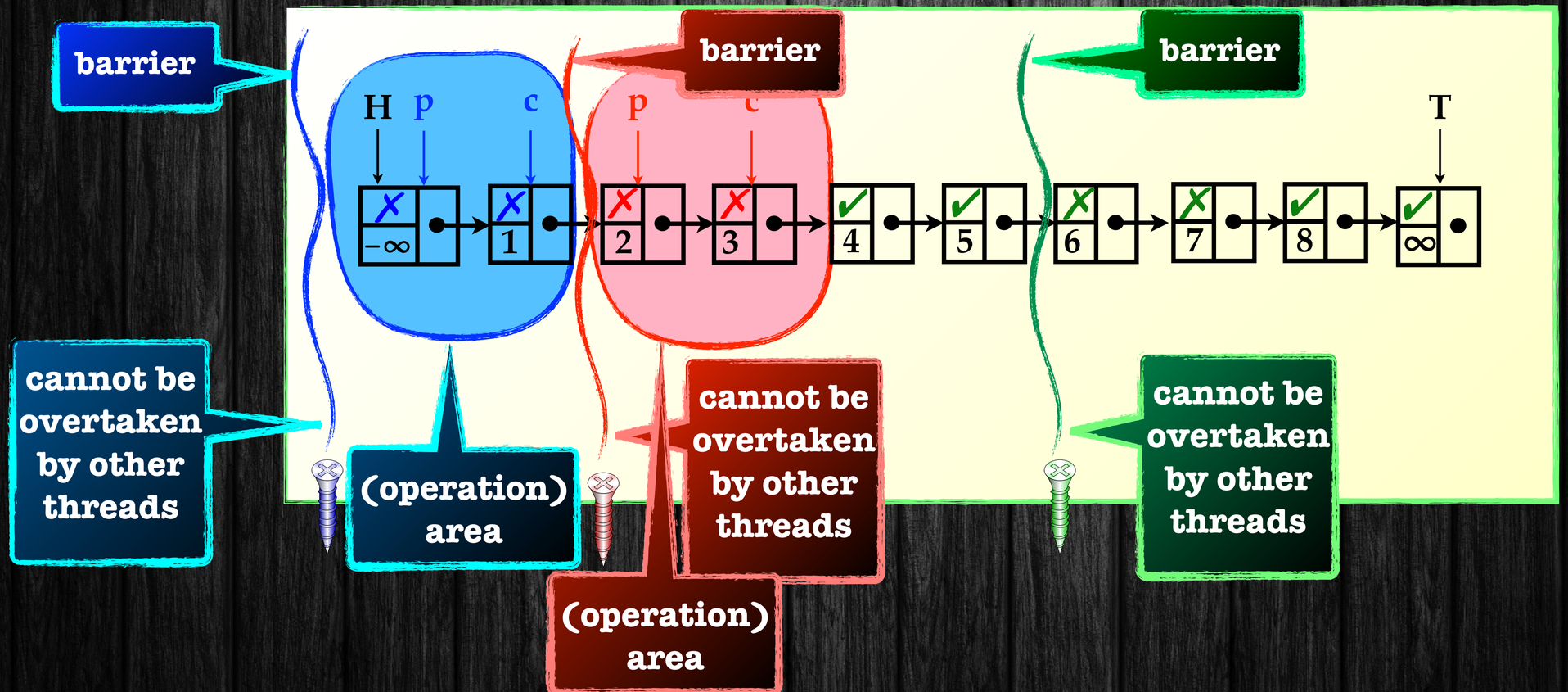


# Sets | Fine-Grained Synchronization

threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

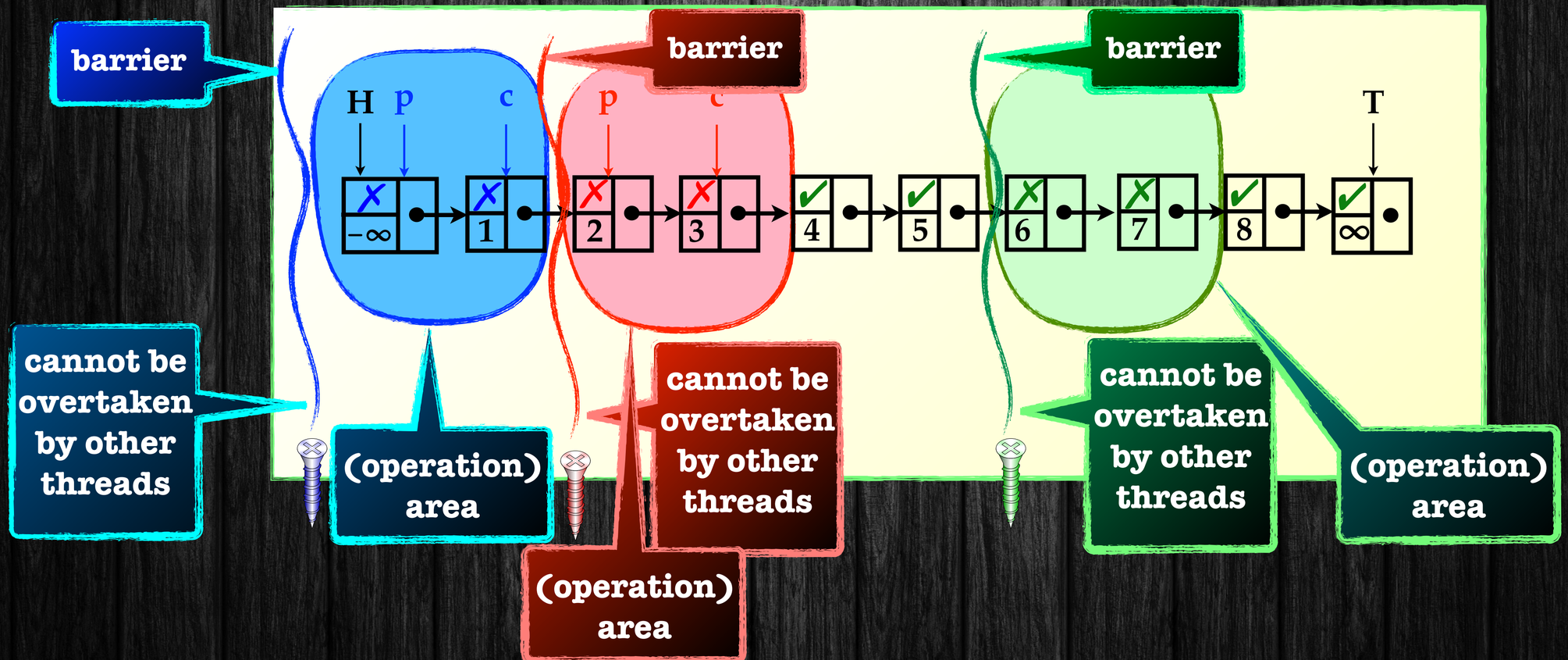


# Sets | Fine-Grained Synchronization

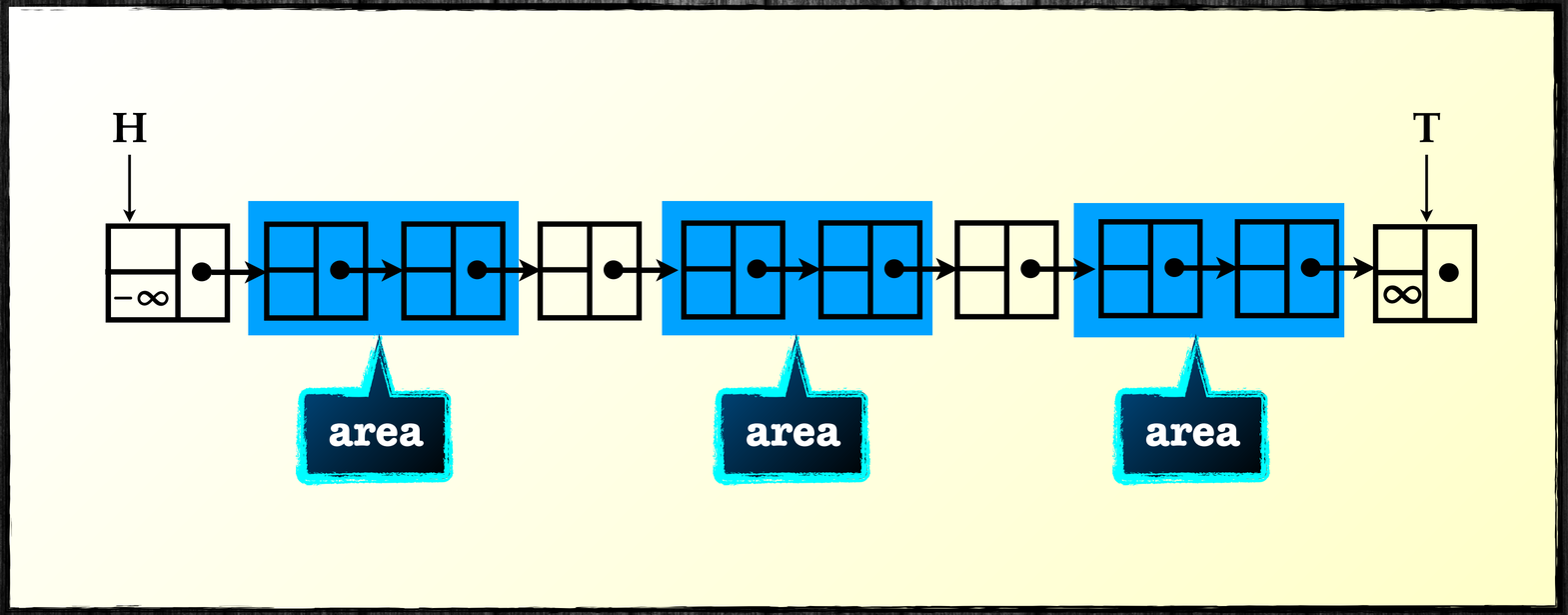
threads cannot overtake each other

threads do not interfere with each other

Lock Coupling

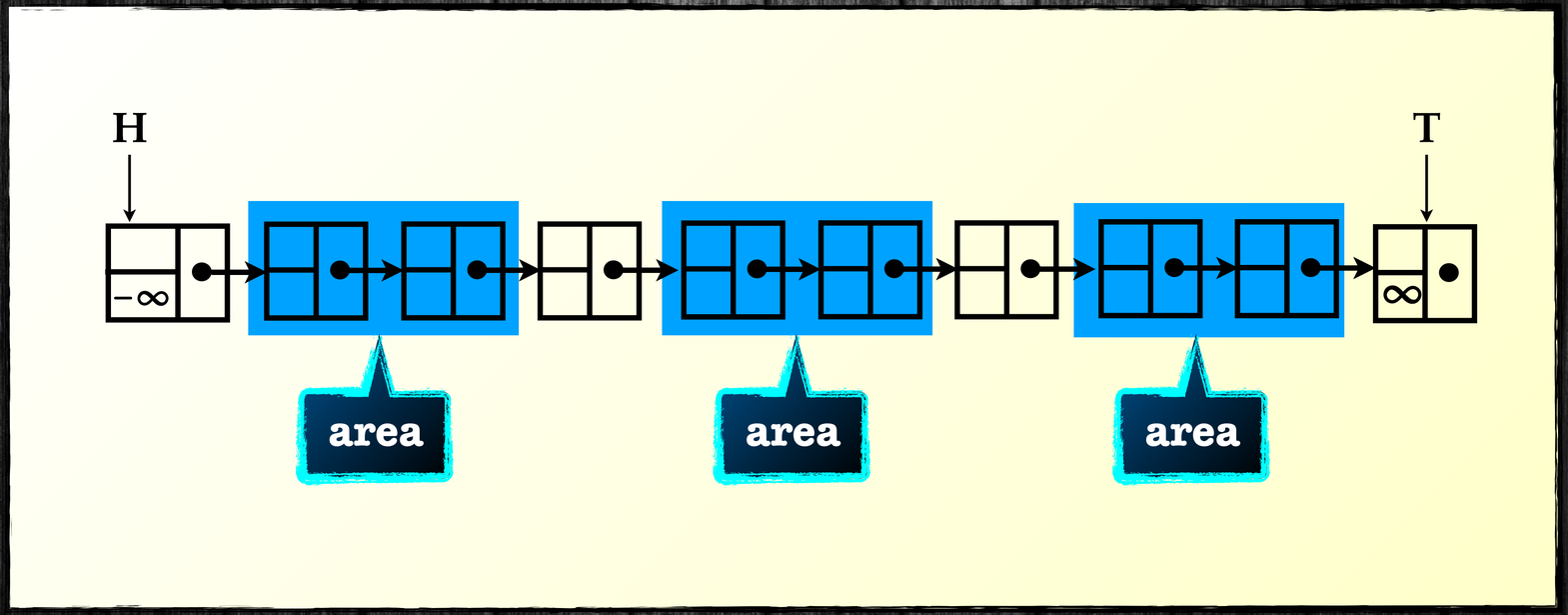


# Sets Fine-Grained Synchronization



# Sets | Fine-Grained Synchronization

**Lock Coupling**

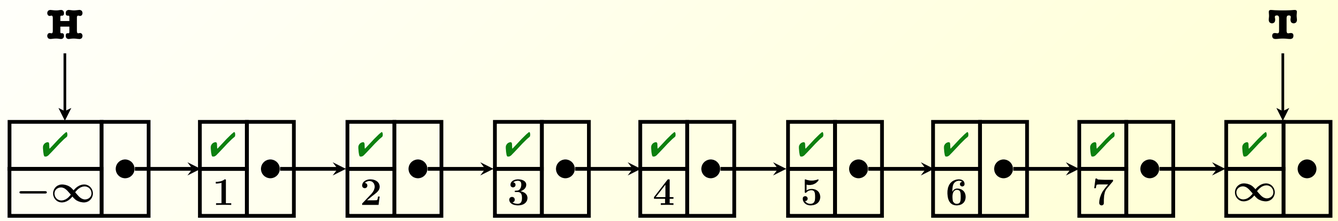


# Sets Fine-Grained Synchronization



rmv(4)

**Lock Coupling**

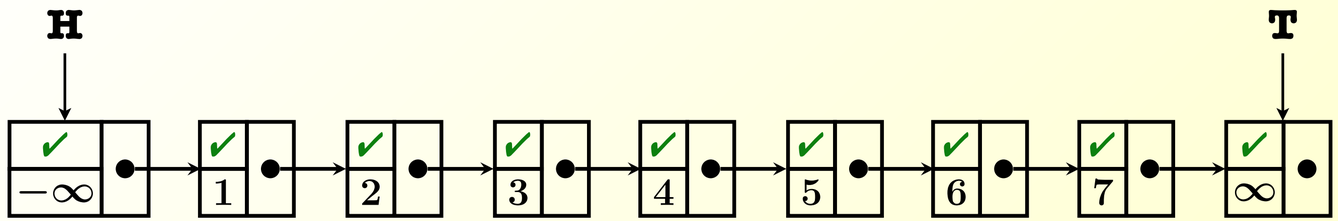


# Sets Fine-Grained Synchronization



rmv(4)

~~Lock Coupling~~



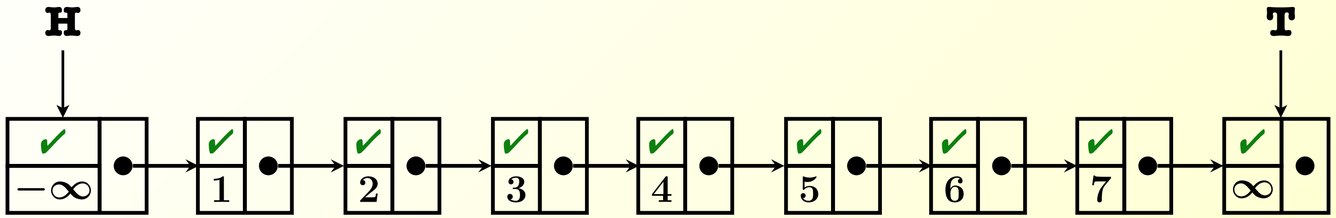
# Sets Fine-Grained Synchronization



rmv(4)

~~Lock Coupling~~

Lock only one cell

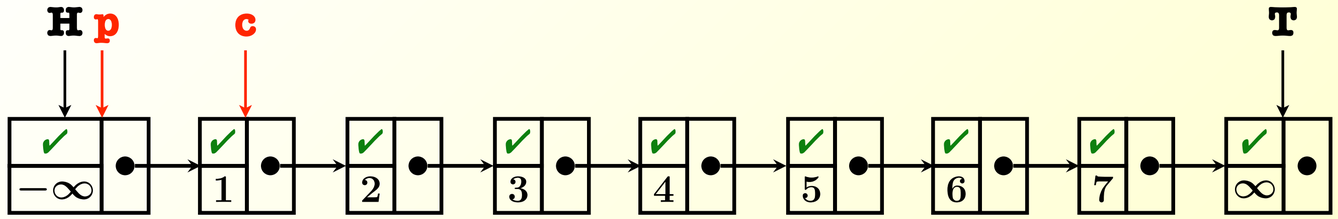


# Sets Fine-Grained Synchronization

rmv(4)

~~Lock Coupling~~

Lock only one cell



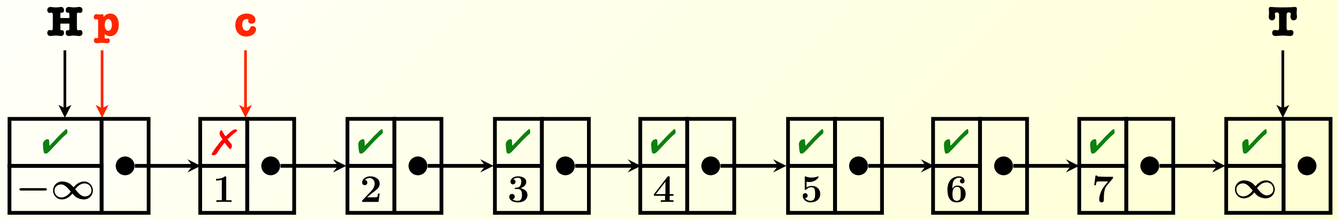
# Sets Fine-Grained Synchronization



rmv(4)

~~Lock Coupling~~

Lock only one cell

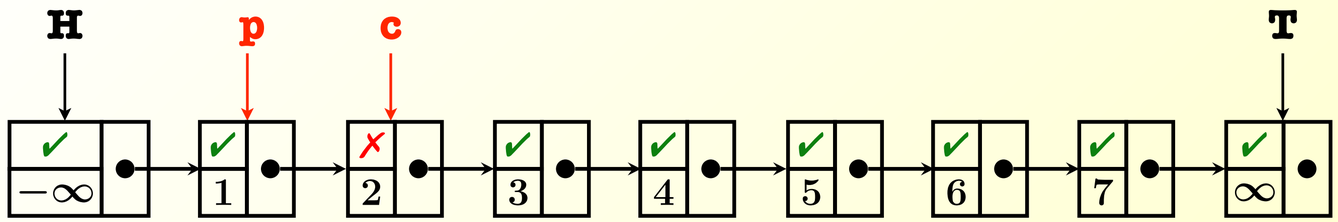


# Sets Fine-Grained Synchronization

rmv(4)

~~Lock Coupling~~

Lock only one cell



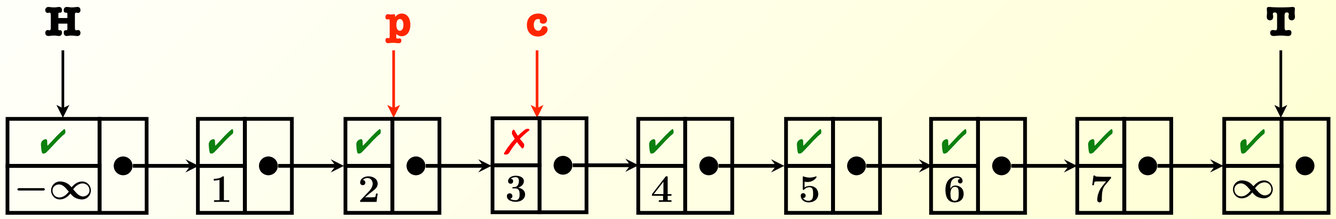
# Sets Fine-Grained Synchronization



rmv(4)

~~Lock Coupling~~

Lock only one cell



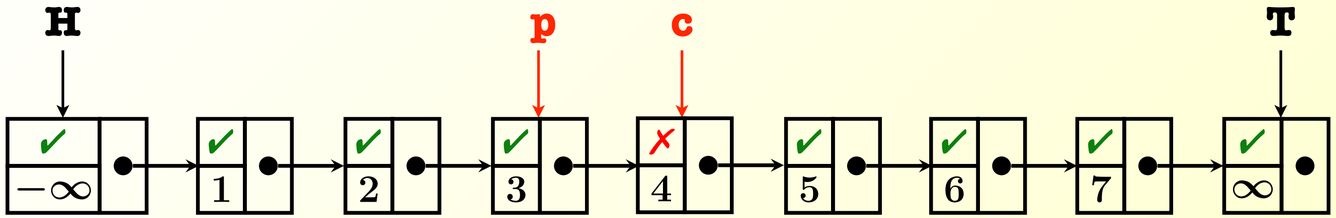
# Sets Fine-Grained Synchronization



rmv(4)

~~Lock Coupling~~

Lock only one cell



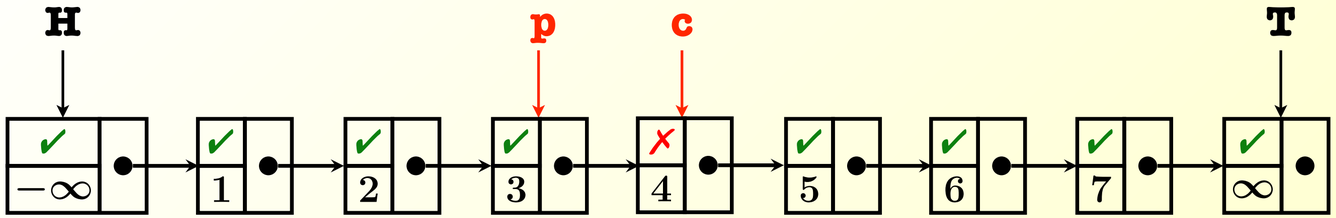
# Sets Fine-Grained Synchronization



rmv(4)

~~Lock Coupling~~

Lock only one cell



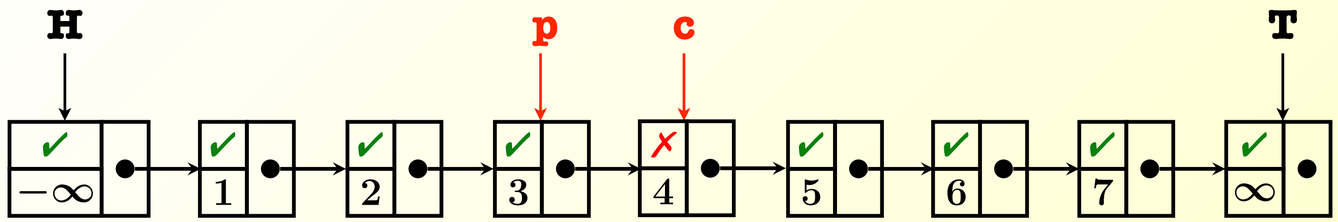
# Sets Fine-Grained Synchronization

rmv(4)

~~Lock Coupling~~

Lock only one cell

rmv(3)



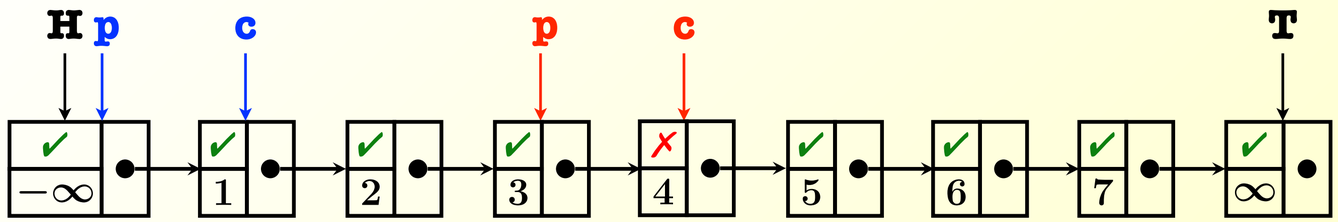
# Sets Fine-Grained Synchronization

rmv(4)

~~Lock Coupling~~

Lock only one cell

rmv(3)



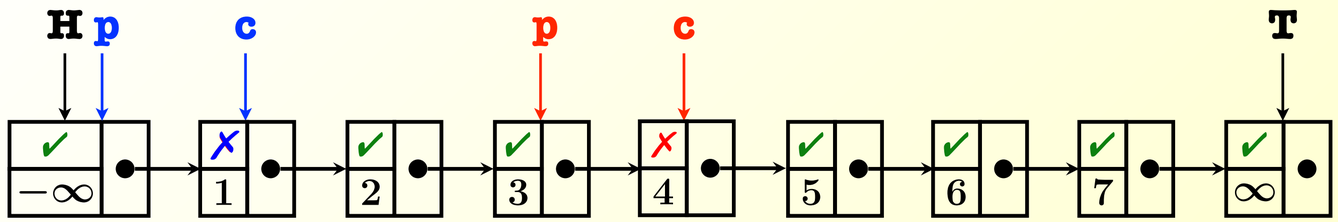
# Sets Fine-Grained Synchronization

rmv(4)

~~Lock Coupling~~

Lock only one cell

rmv(3)



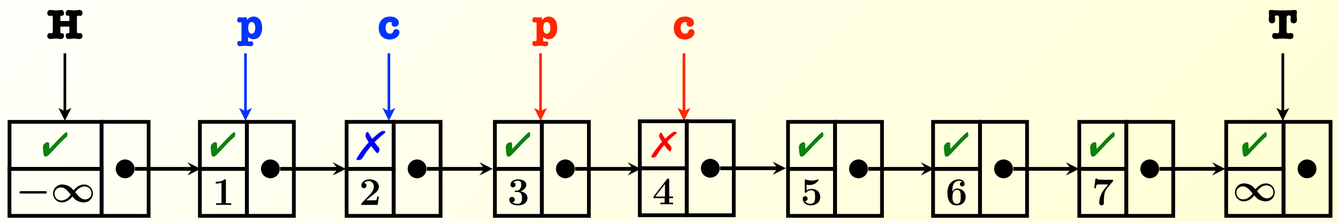
# Sets Fine-Grained Synchronization

rmv(4)

~~Lock Coupling~~

Lock only one cell

rmv(3)



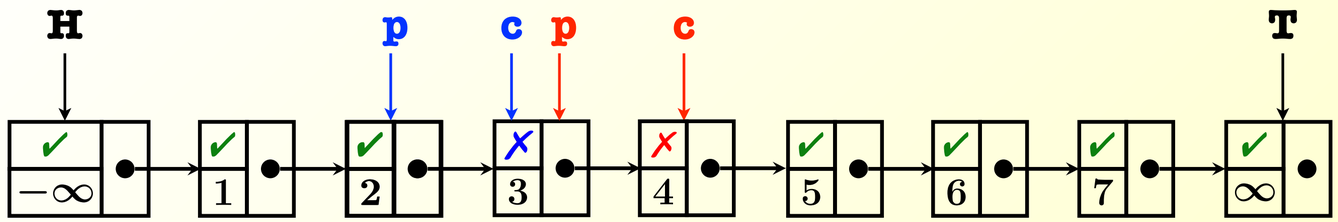
# Sets Fine-Grained Synchronization

rmv(4)

~~Lock Coupling~~

Lock only one cell

rmv(3)



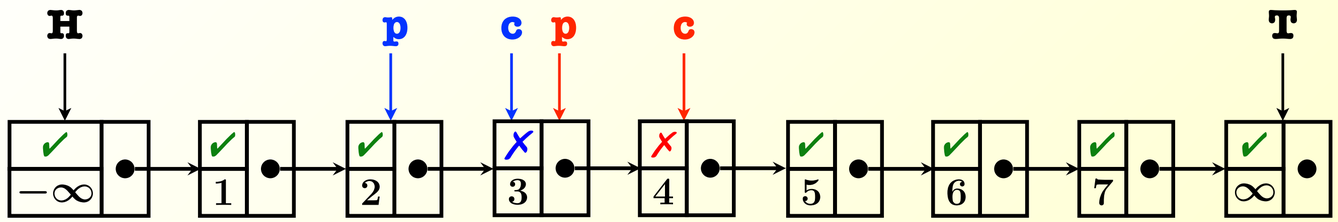
# Sets Fine-Grained Synchronization

rmv(4)

~~Lock Coupling~~

Lock only one cell

rmv(3)



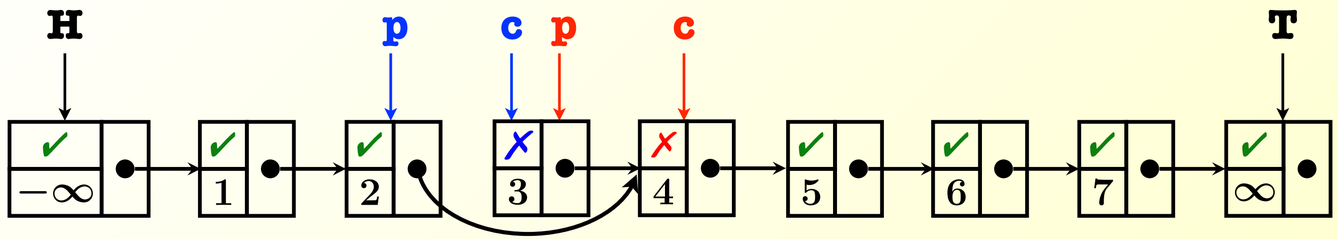
# Sets Fine-Grained Synchronization

rmv(4)

~~Lock Coupling~~

Lock only one cell

rmv(3)



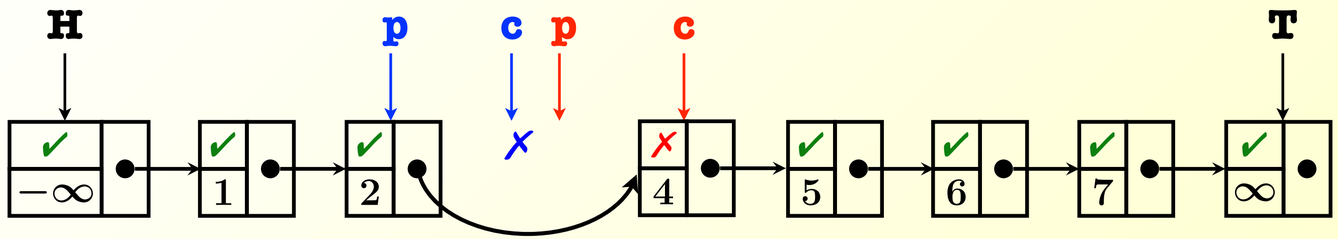
# Sets Fine-Grained Synchronization

rmv(4)

~~Lock Coupling~~

Lock only one cell

rmv(3)



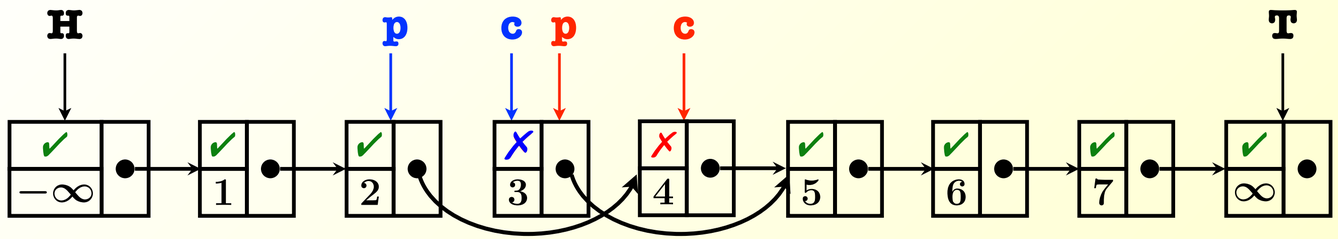
# Sets Fine-Grained Synchronization

rmv(4)

~~Lock Coupling~~

Lock only one cell

rmv(3)



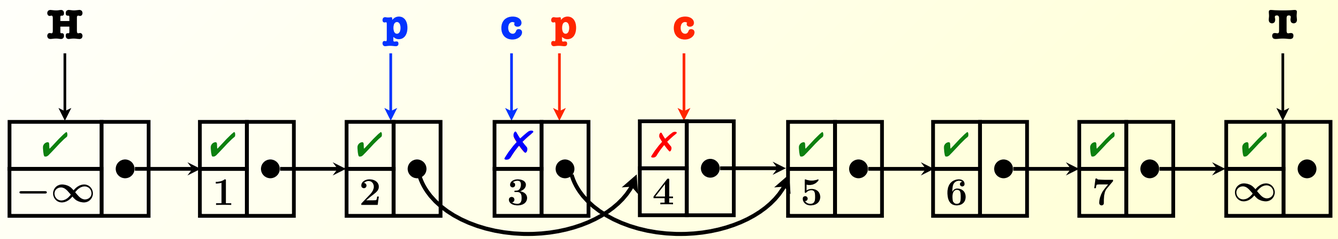
# Sets Fine-Grained Synchronization

rmv(4)

~~Lock Coupling~~

Lock only one cell

rmv(3)



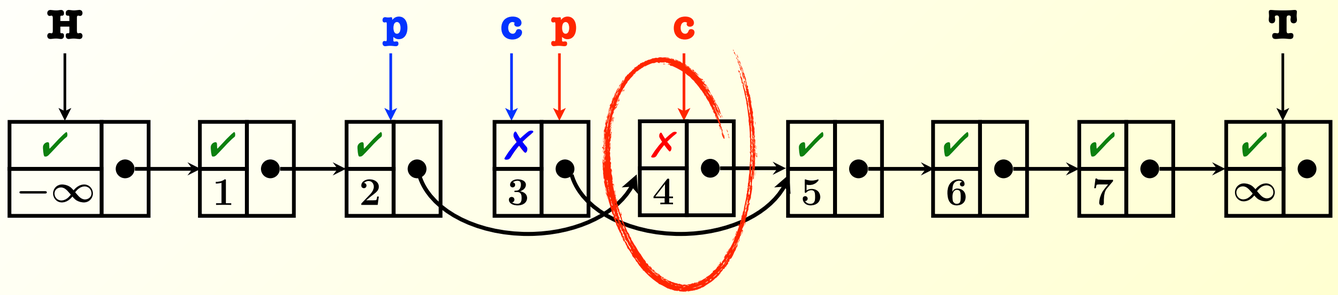
# Sets Fine-Grained Synchronization

rmv(4)

~~Lock Coupling~~

Lock only one cell

rmv(3)



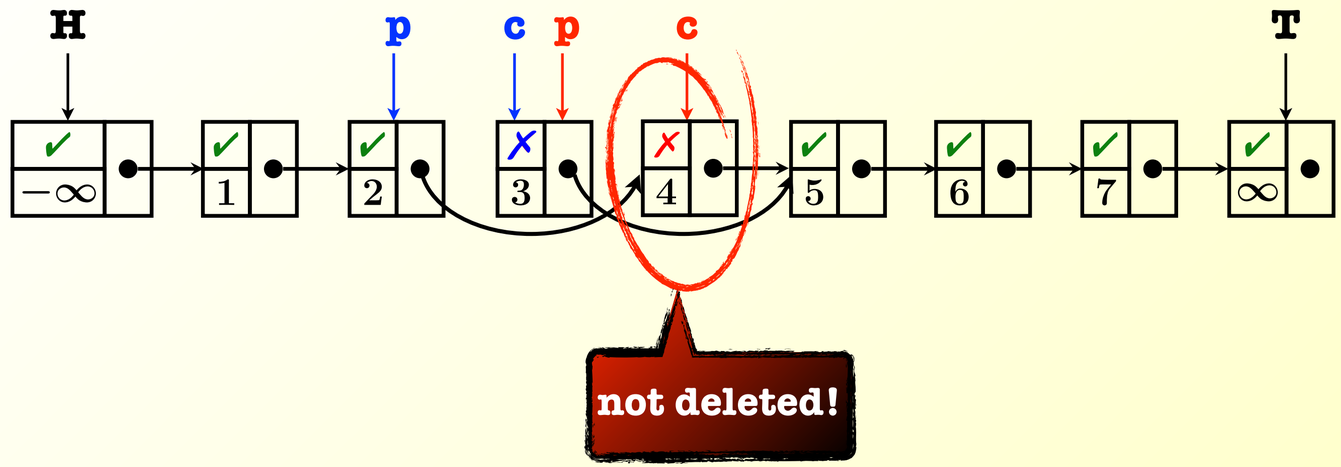
# Sets | Fine-Grained Synchronization

rmv(4)

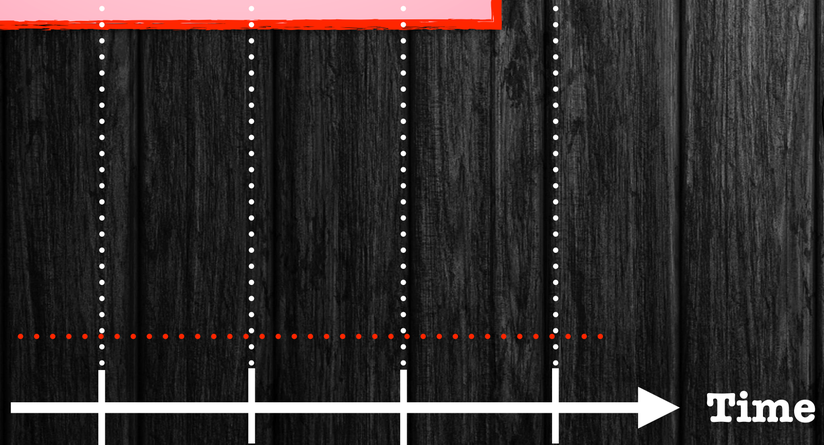
~~Lock Coupling~~

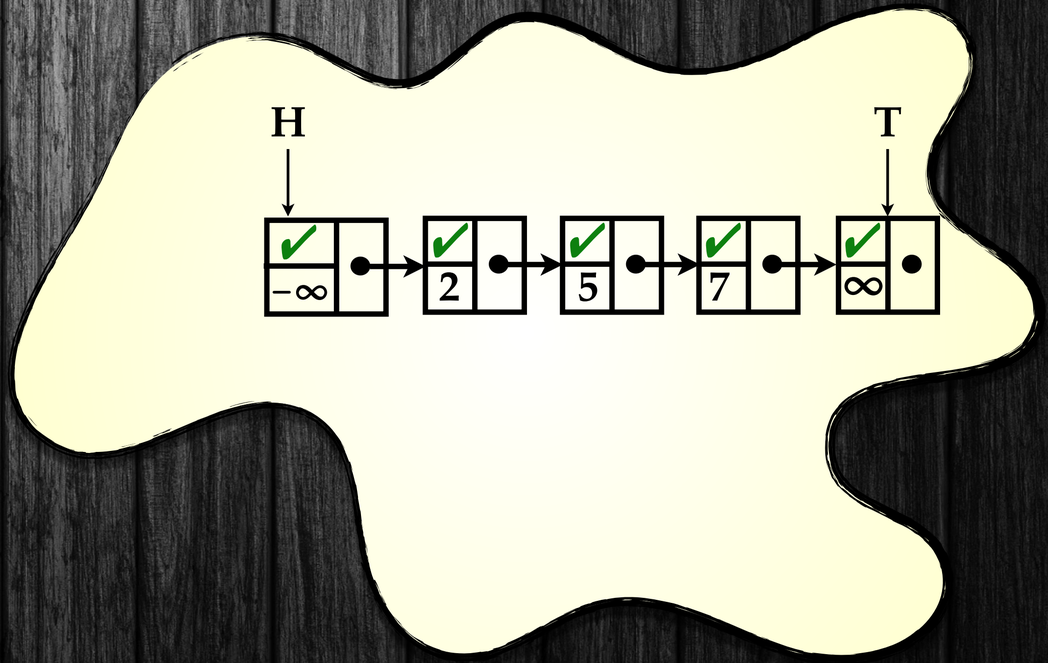
Lock only one cell

rmv(3)



```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17   unlock(c)
18   unlock(p)
```



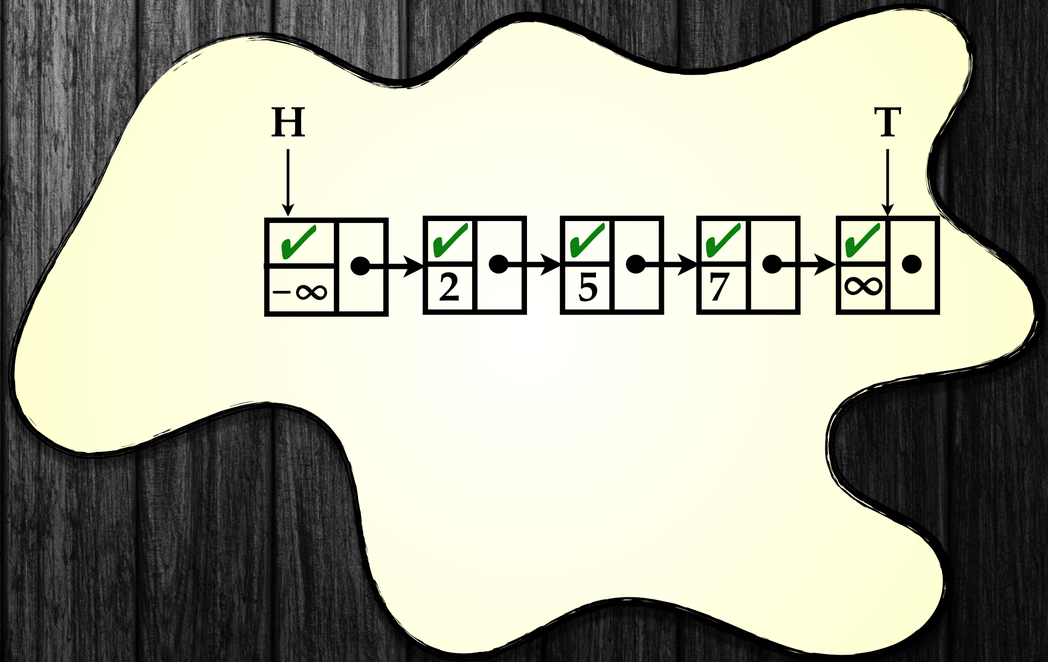



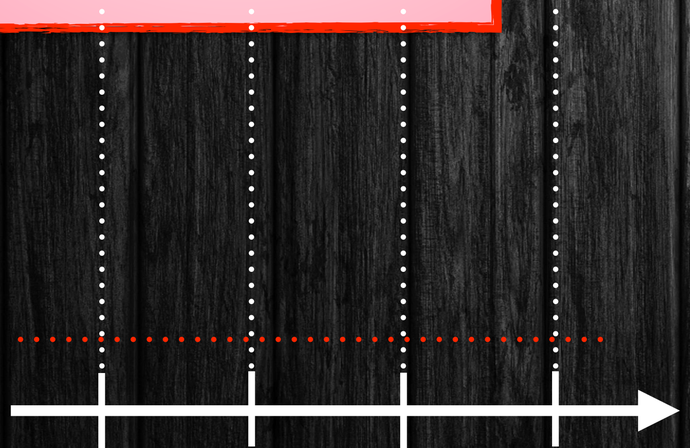
# Sets | Fine-Grained Synchronization | Linearization Policy

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17   unlock(c)
18   unlock(p)
```

LP

"time point at which I find a key $\geq$ input key"



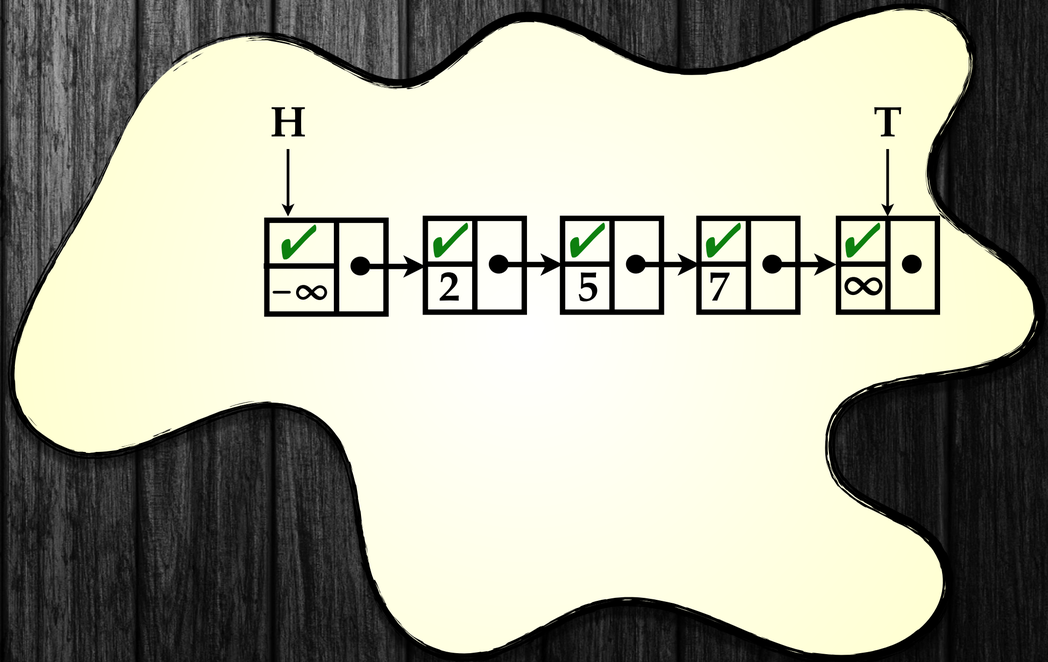
# Sets | Fine-Grained Synchronization | Linearization Policy

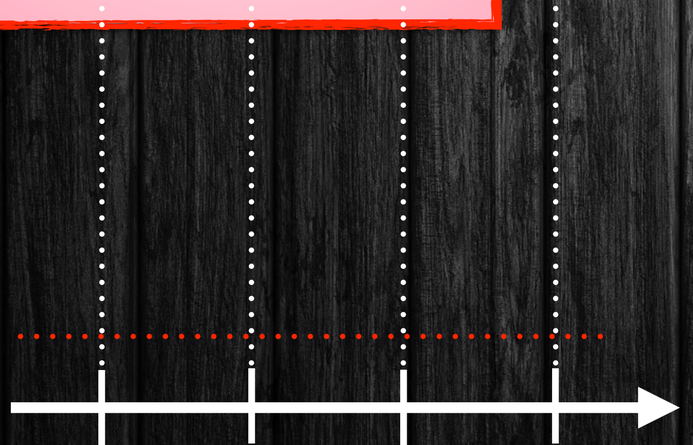
```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17   unlock(c)
18   unlock(p)
```

LP

"time point at which I find a key ≥ input key"

"time point at which I lock the cell with key ≥ input key"



# Sets | Fine-Grained Synchronization | Linearization Policy

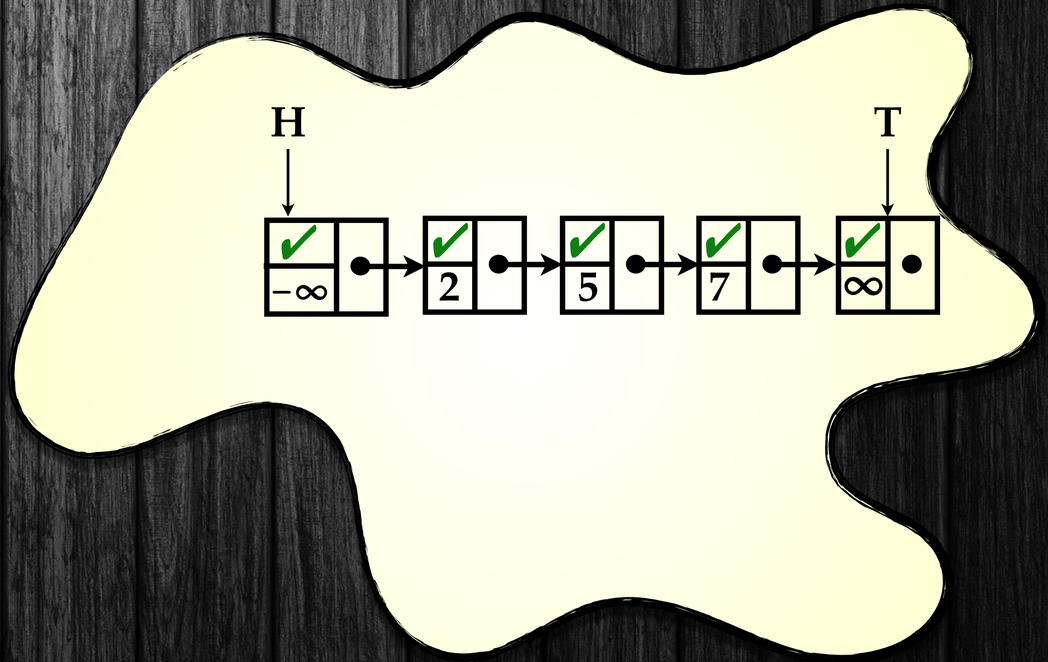
```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17   unlock(c)
18   unlock(p)
```

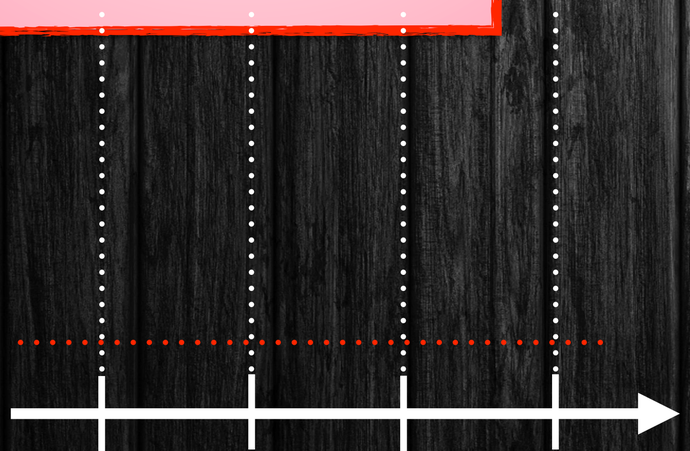
LP

"time point at which I find a key ≥ input key"

"time point at which I lock the cell with key ≥ input key"

add(4)



# Sets | Fine-Grained Synchronization | Linearization Policy

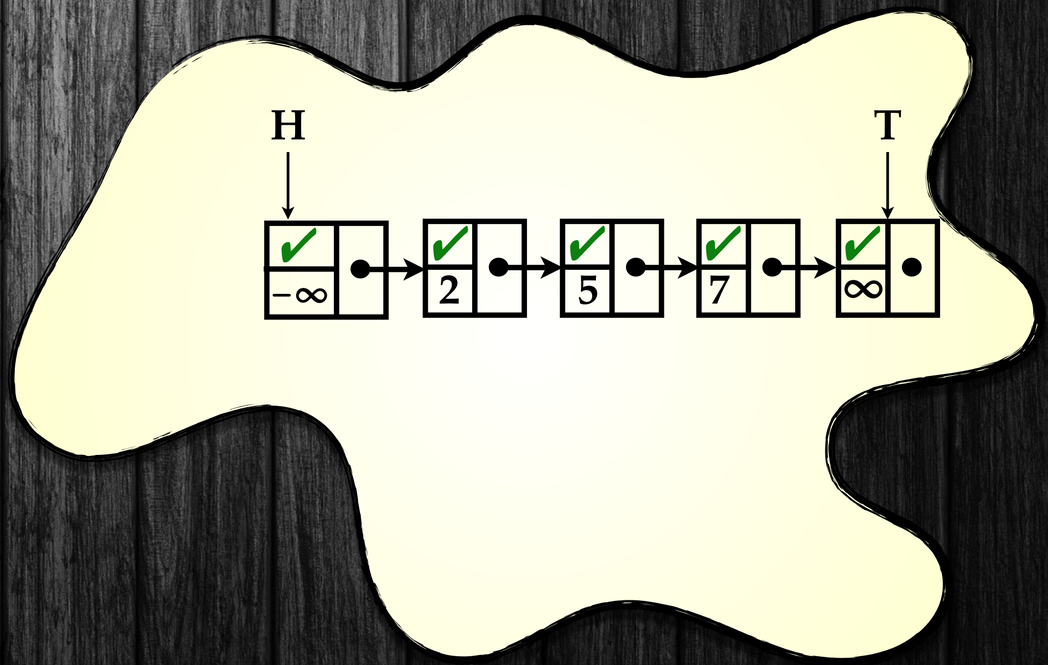
```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17   unlock(c)
18   unlock(p)
```

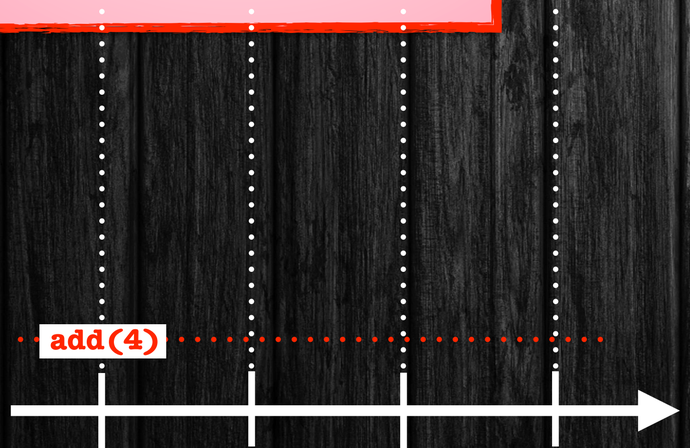
LP

"time point at which I find a key ≥ input key"

"time point at which I lock the cell with key ≥ input key"

add(4)



# Sets | Fine-Grained Synchronization | Linearization Policy

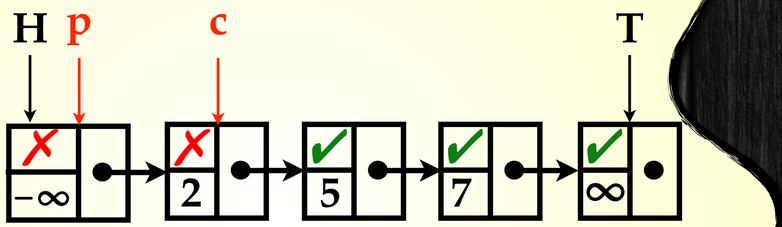
```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17   unlock(c)
18   unlock(p)
```



add(4)

LP

"time point at which I find a key ≥ input key"






# Sets | Fine-Grained Synchronization | Linearization Policy

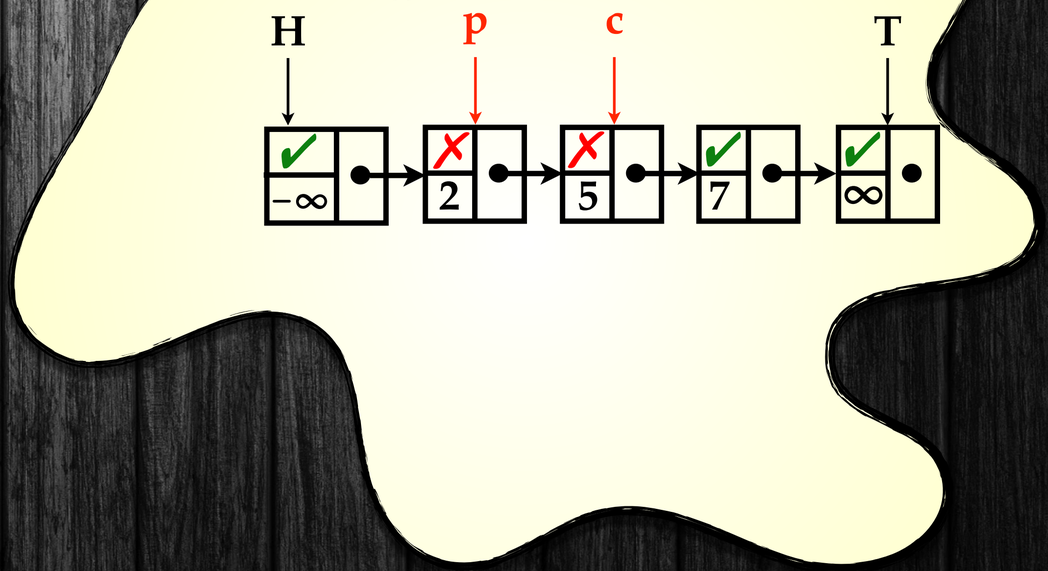
```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17   unlock(c)
18   unlock(p)
```



LP

"time point at which I find a key ≥ input key"

add(4)





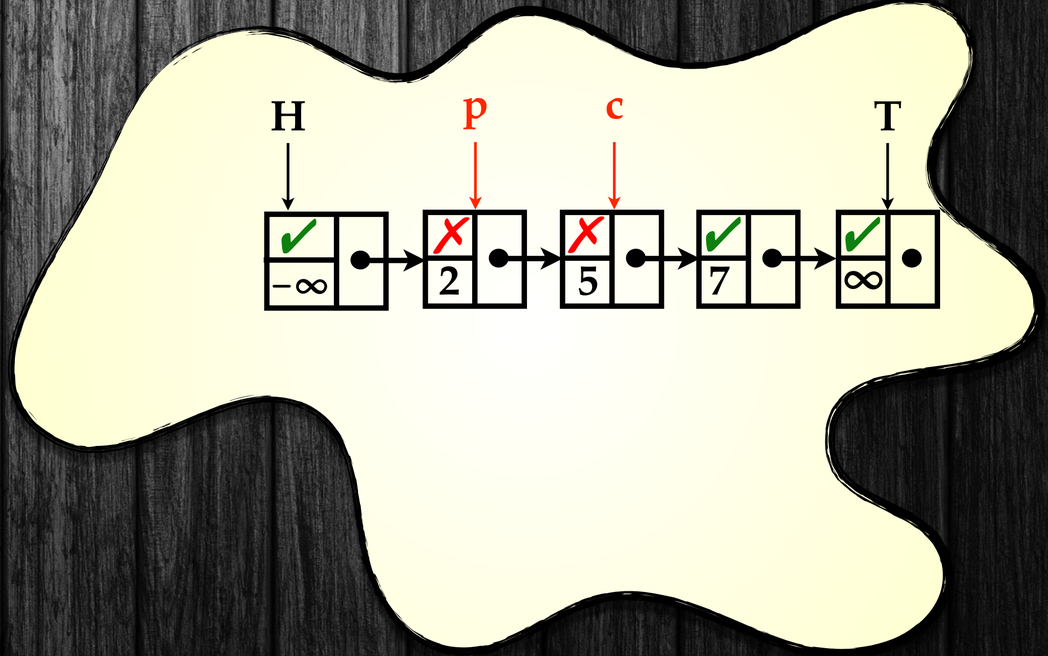

# Sets | Fine-Grained Synchronization | Linearization Policy

```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17   unlock(c)
18   unlock(p)
```

LP

"time point at which I find a key ≥ input key"

add(4)






# Sets | Fine-Grained Synchronization | Linearization Policy

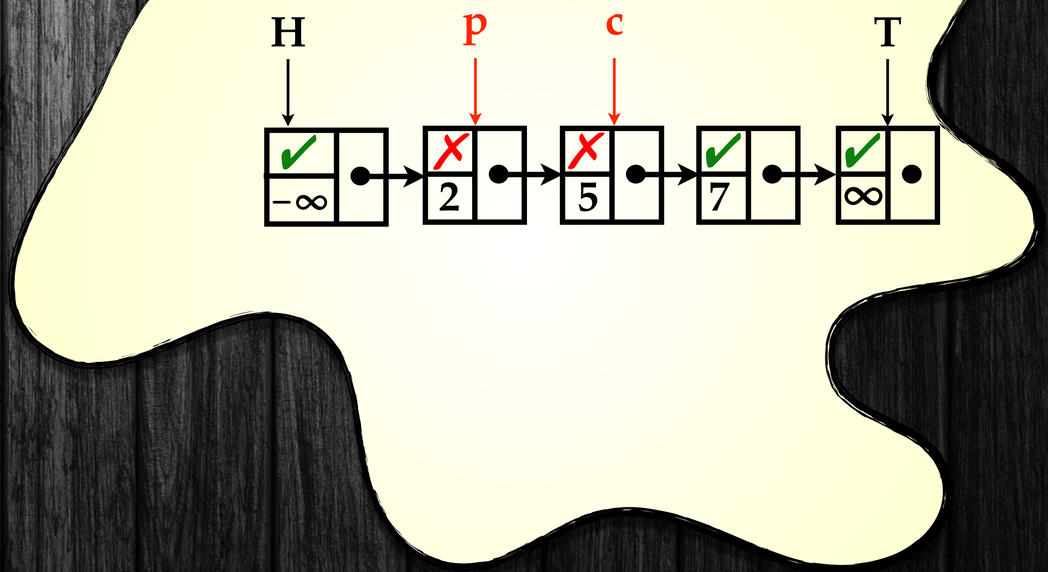
```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17    unlock(c)
18    unlock(p)
```

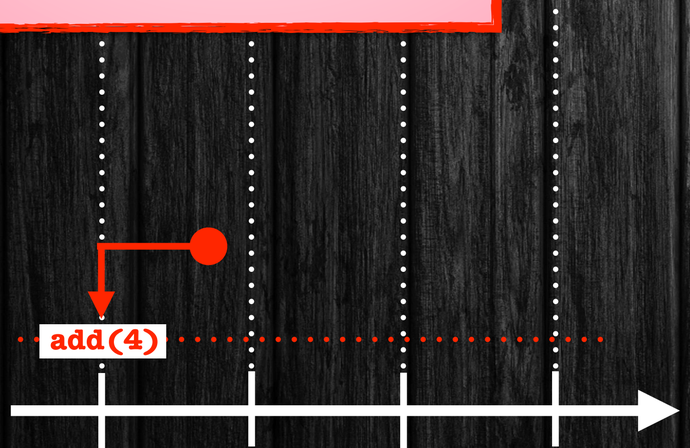


LP

"time point at which I find a key ≥ input key"

add(4)



# Sets | Fine-Grained Synchronization | Linearization Policy

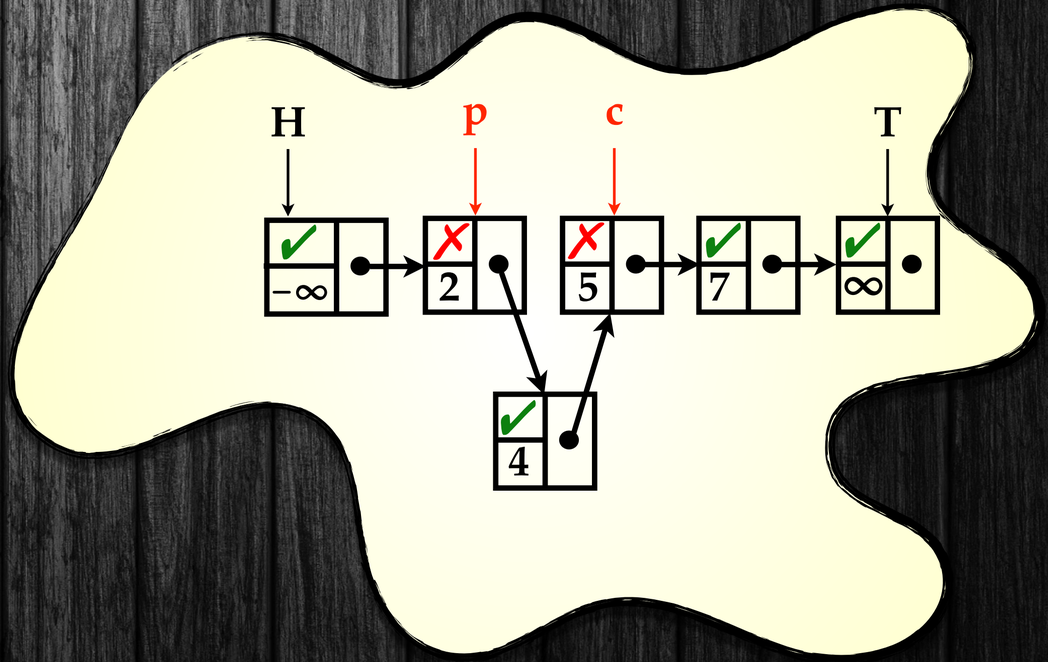
```
add(k):
Node p, c
 1  lock(H)
 2  p = H
 3  c = p.next;
 4  lock(c);
 5  while (c.key < k)
 6    unlock(p);
 7    p = c;
 8    c = c.next;
 9    lock(c);
10  if (c.key = k)
11    return false
12  else
13    n = new Node(k,-)
14    n.next = c
15    p.next = n
16    return true
17    unlock(c)
18    unlock(p)
```

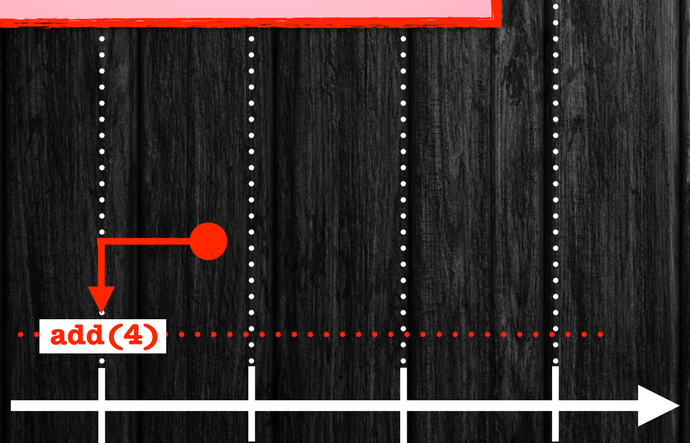


add(4)

LP

"time point at which I find a key ≥ input key"



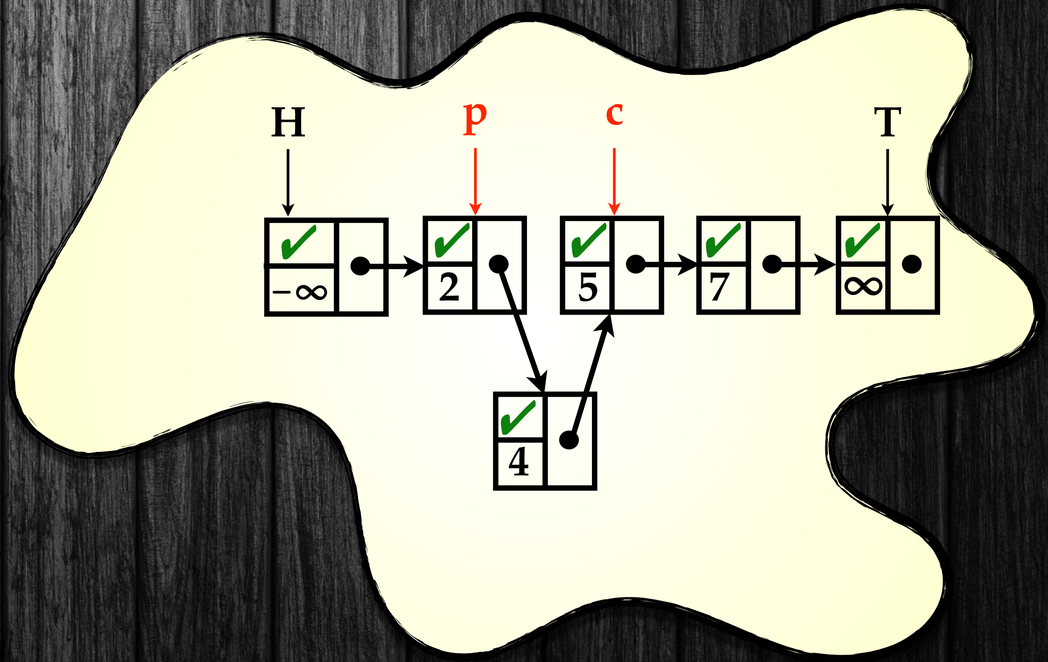
# Sets | Fine-Grained Synchronization | Linearization Policy

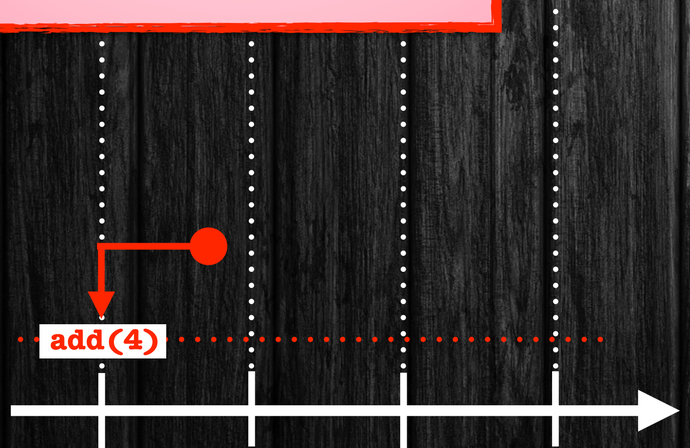
```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17   unlock(c)
18   unlock(p)
```

LP

"time point at which I find a key ≥ input key"

add(4)



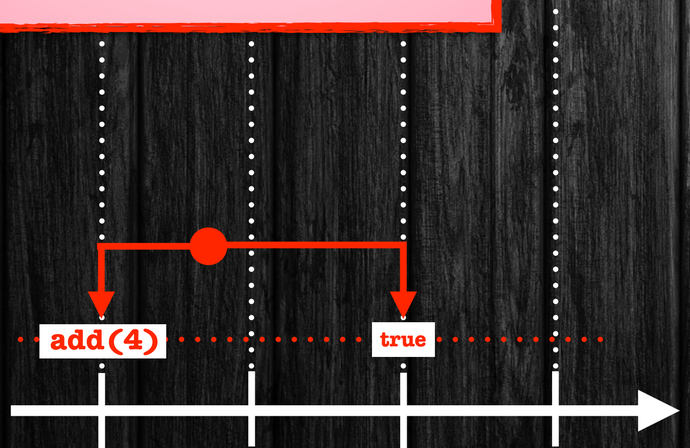
# Sets | Fine-Grained Synchronization | Linearization Policy

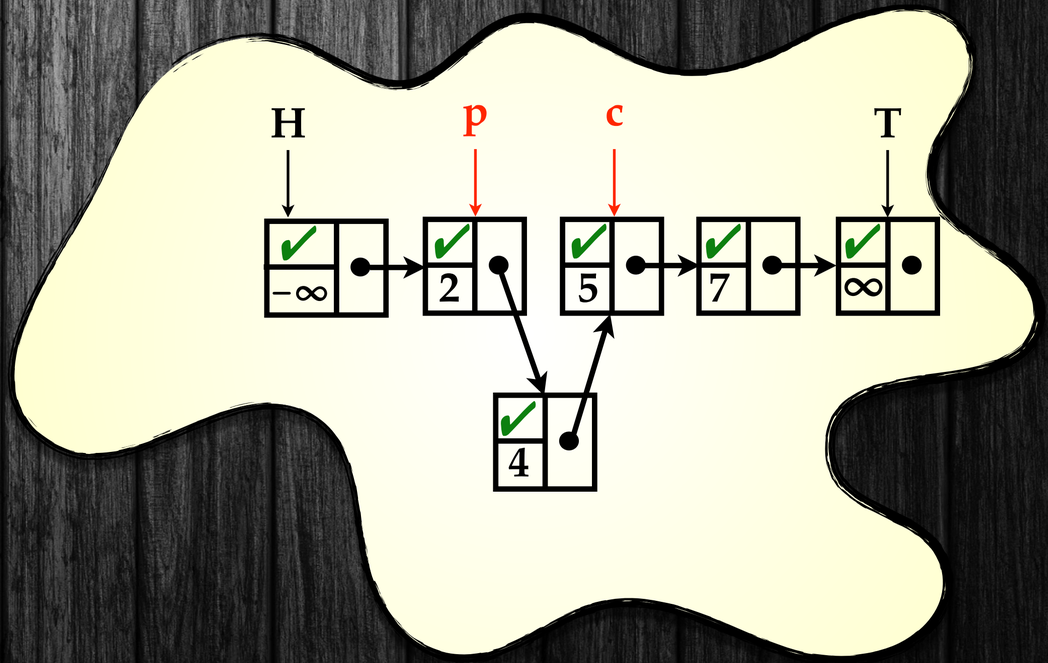
```
add(k):
Node p, c
1  lock(H)
2  p = H
3  c = p.next;
4  lock(c);
5  while (c.key < k)
6    unlock(p);
7    p = c;
8    c = c.next;
9    lock(c);
10 if (c.key = k)
11   return false
12 else
13   n = new Node(k,-)
14   n.next = c
15   p.next = n
16   return true
17   unlock(c)
18   unlock(p)
```

LP

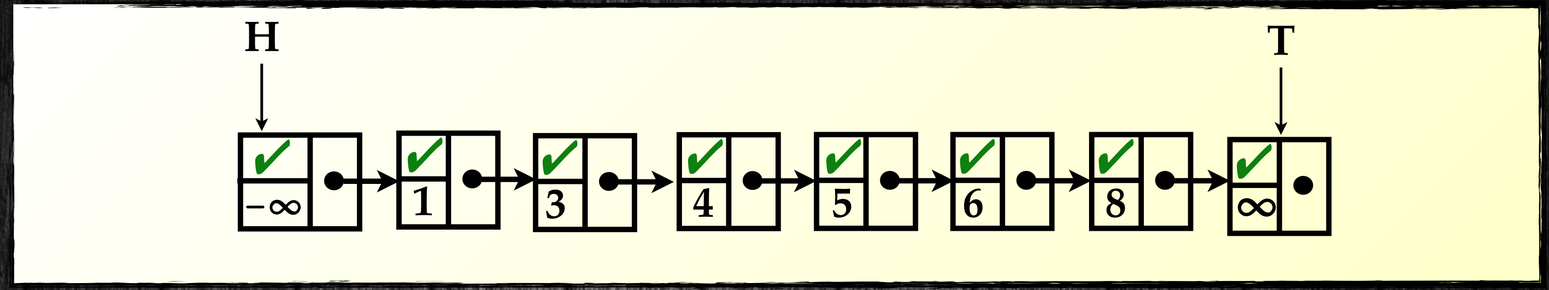
"time point at which I find a key ≥ input key"

add(4)

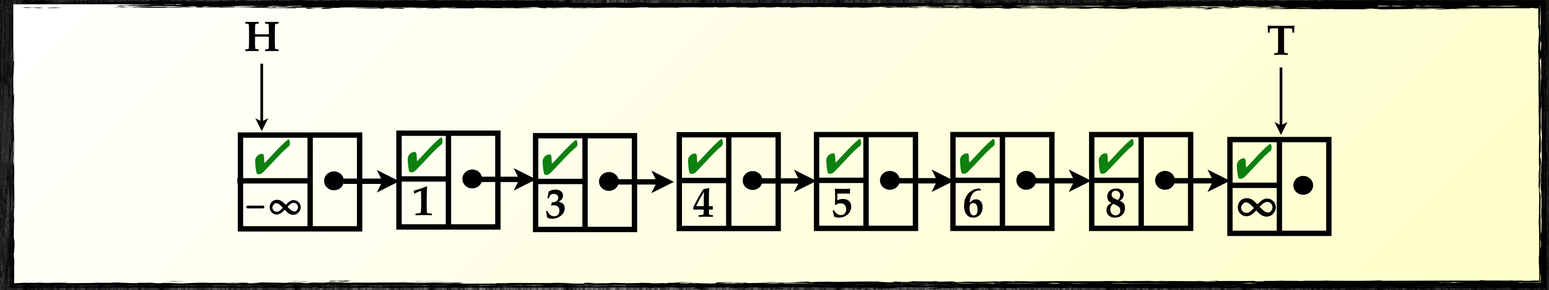


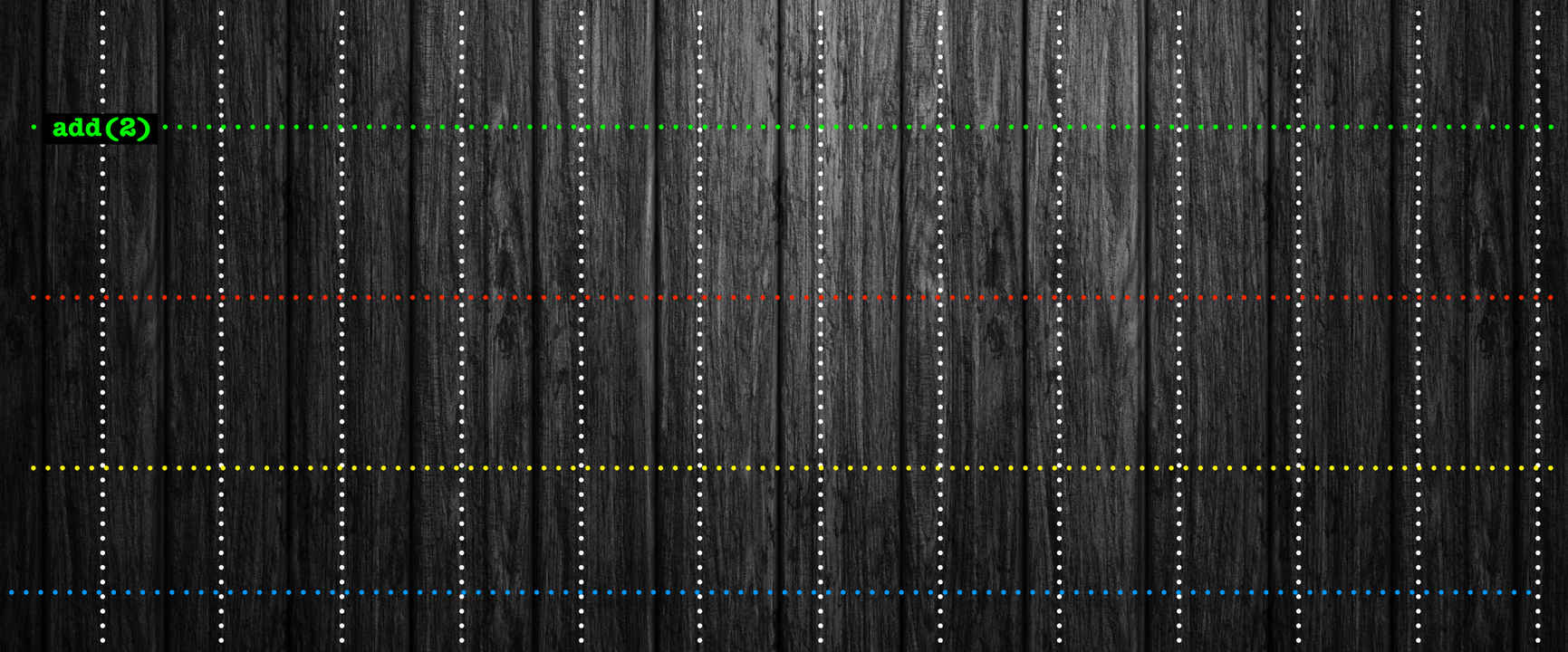
# Sets | Fine-Grained Synchronization | Linearization Policy



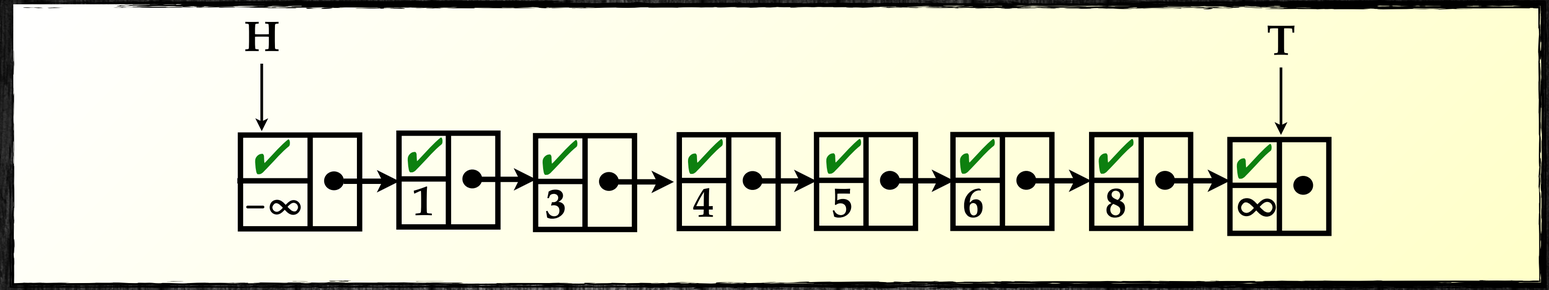
# Sets | Fine-Grained Synchronization | Linearization Policy

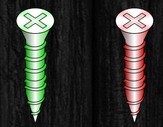


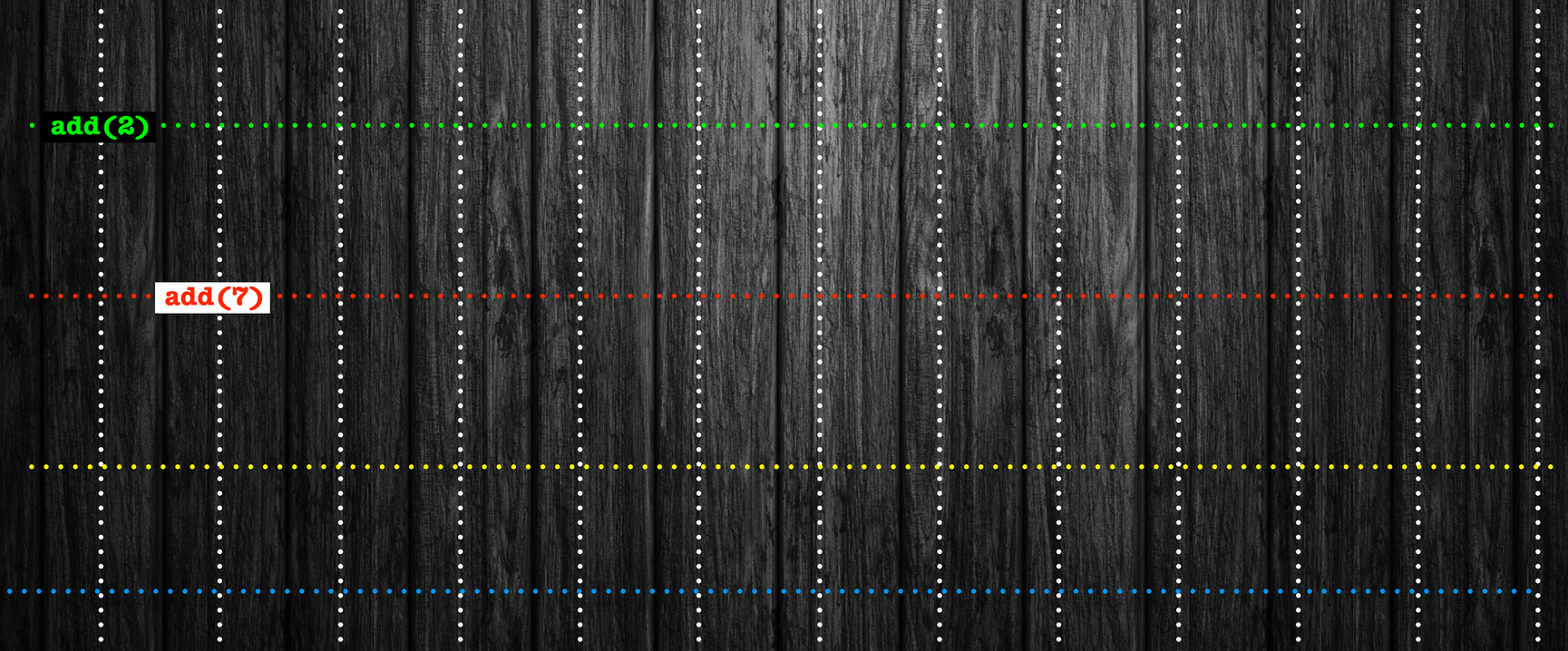


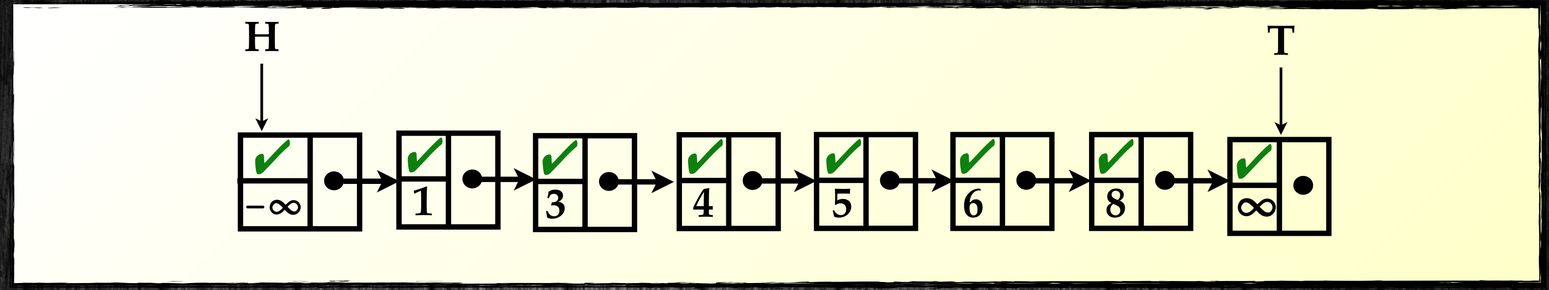
# Sets | Fine-Grained Synchronization | Linearization Policy



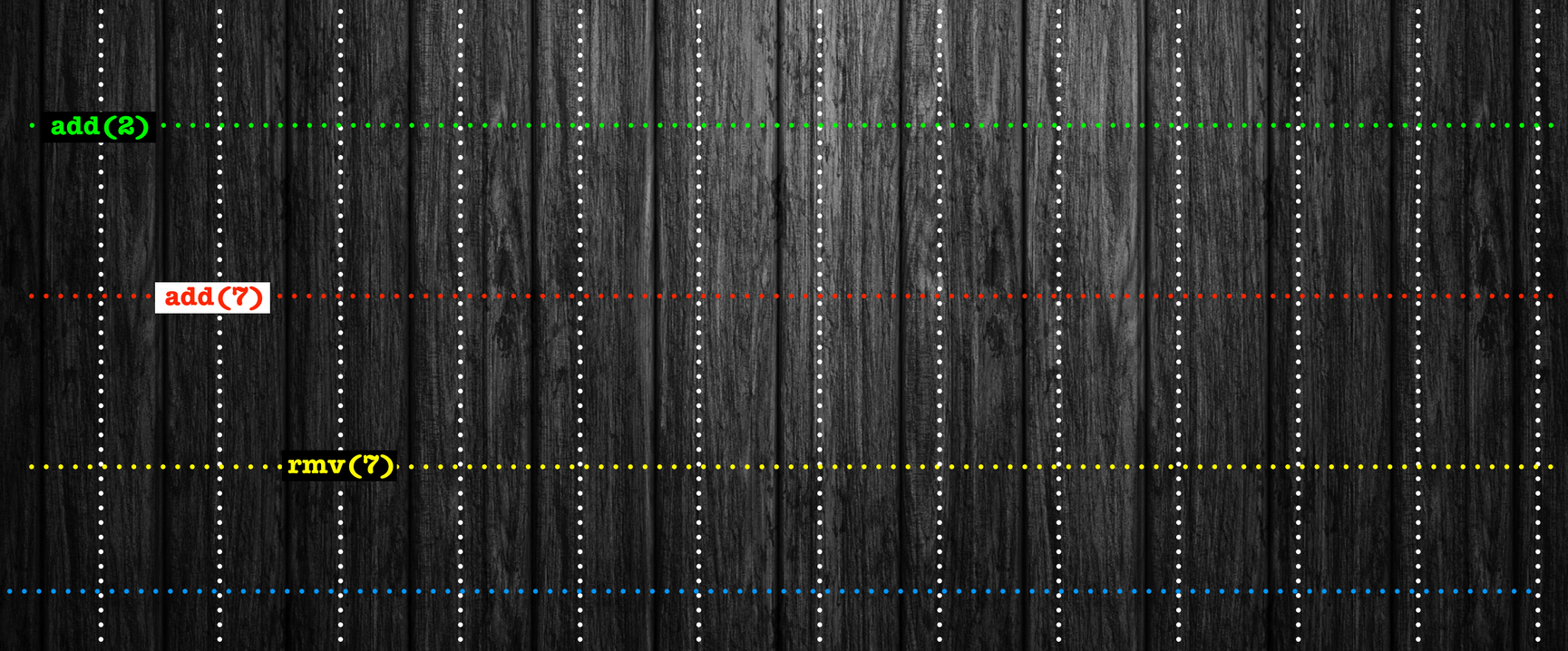





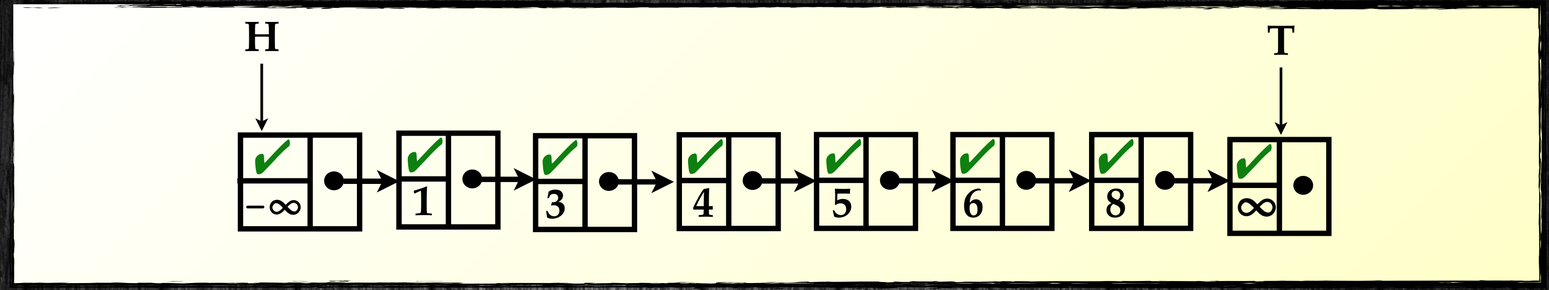
# Sets | Fine-Grained Synchronization | Linearization Policy





# Sets | Fine-Grained Synchronization | Linearization Policy

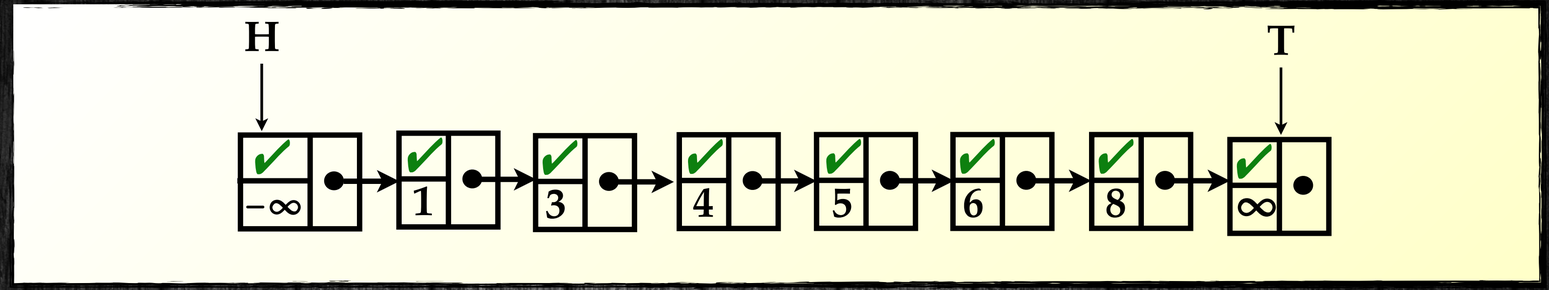



add(2)

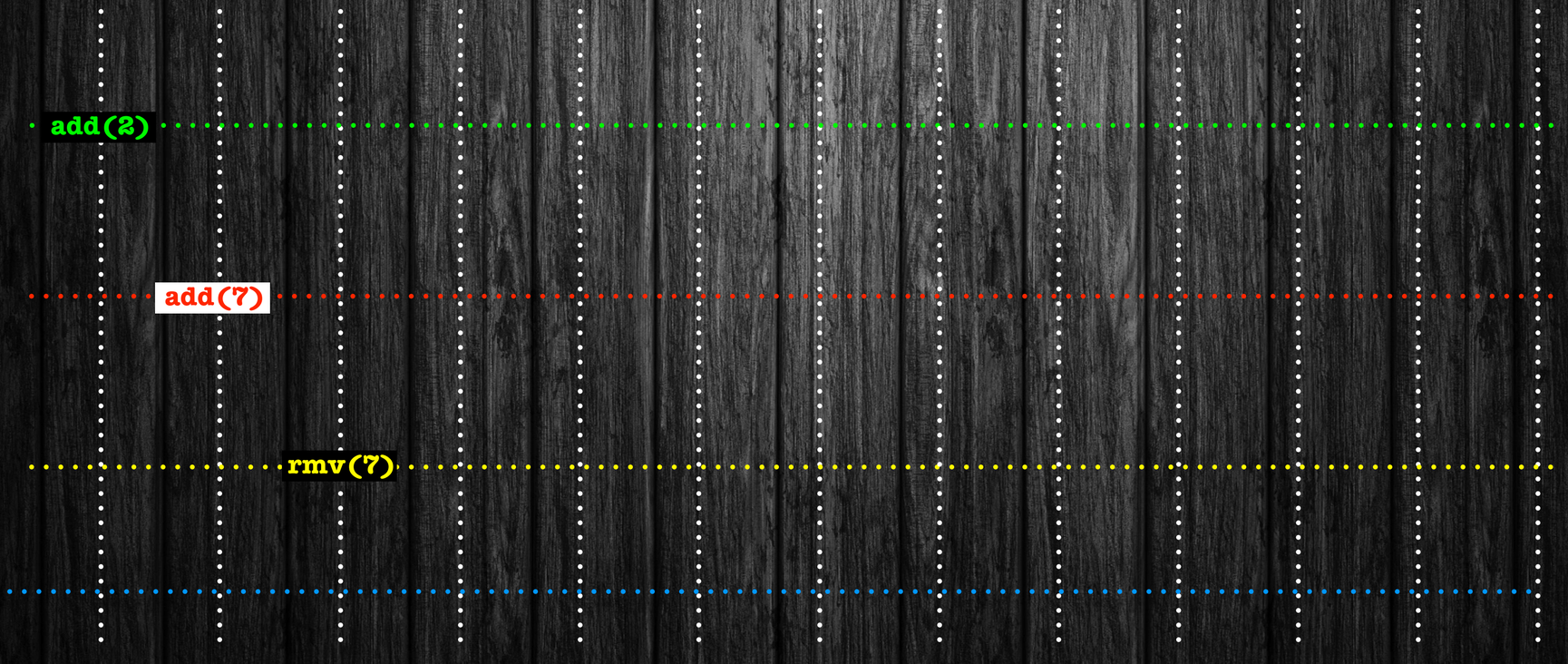
add(7)

rmv(7)

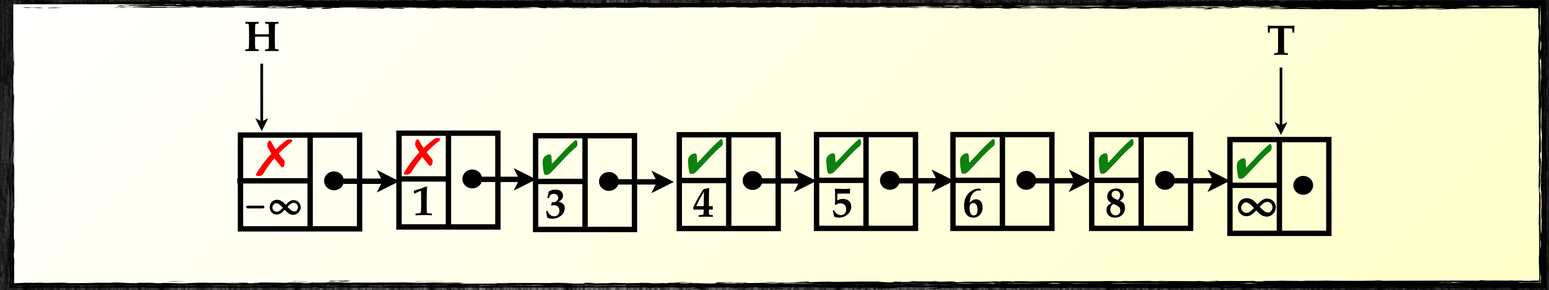
# Sets | Fine-Grained Synchronization | Linearization Policy



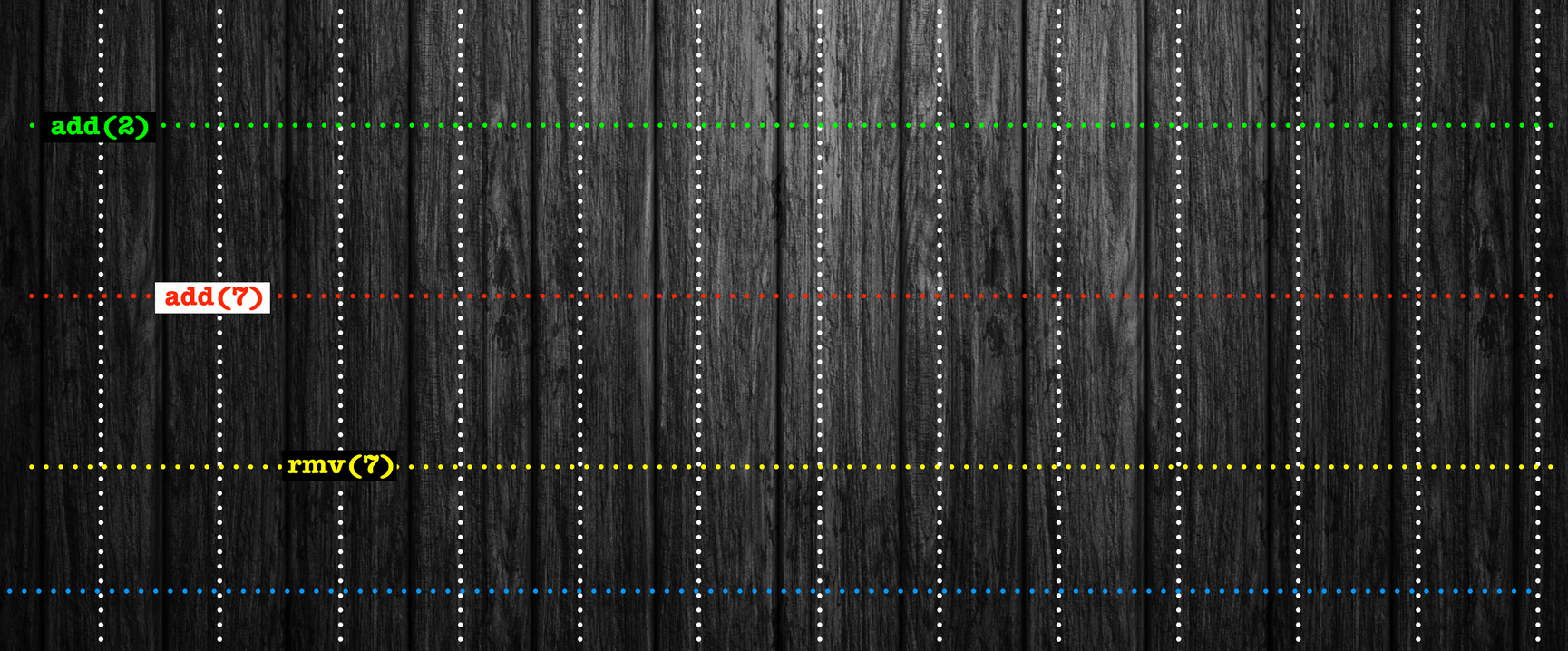


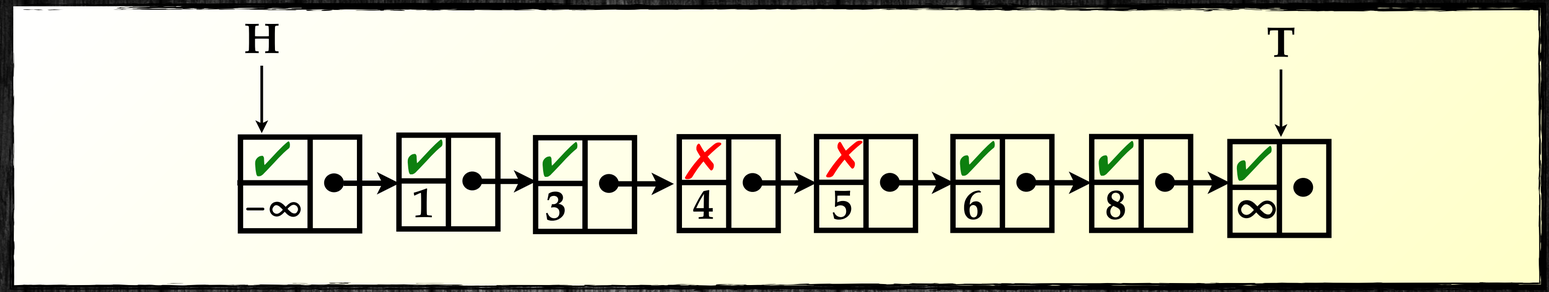
# Sets | Fine-Grained Synchronization | Linearization Policy



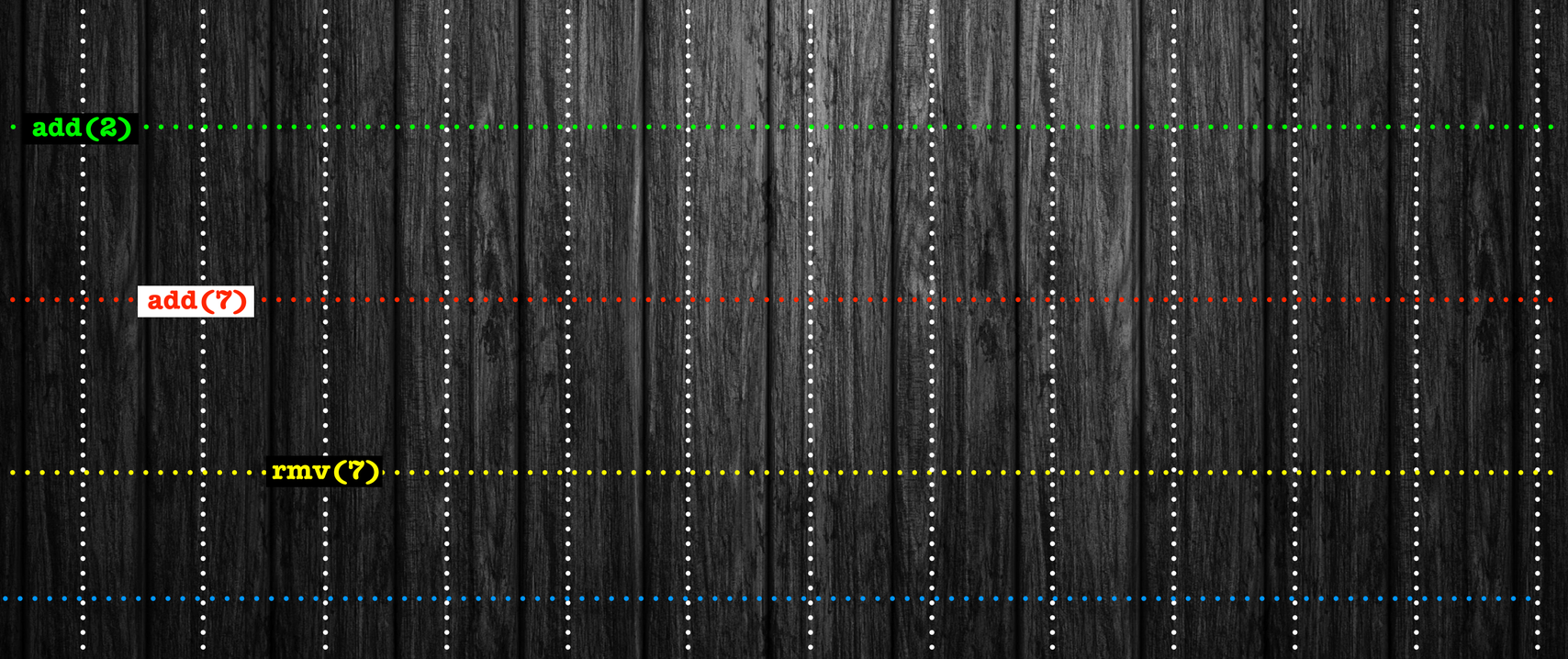


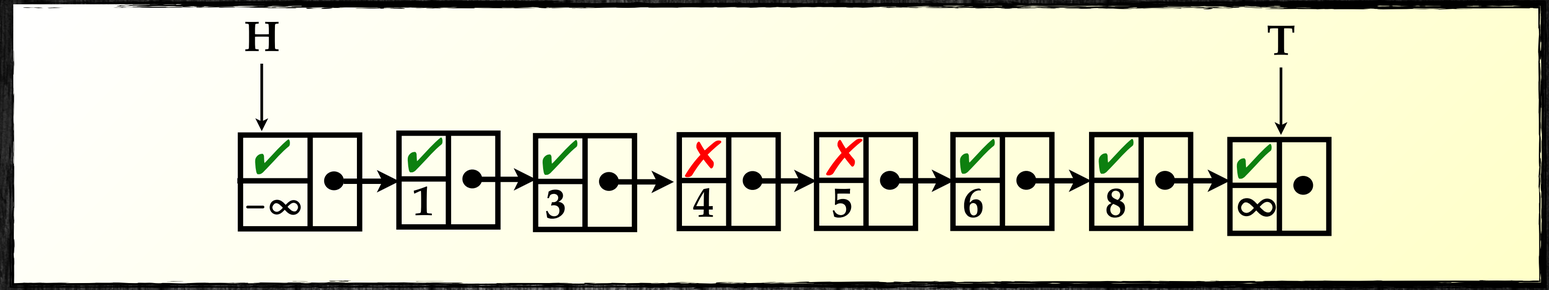
# Sets | Fine-Grained Synchronization | Linearization Policy





# Sets | Fine-Grained Synchronization | Linearization Policy

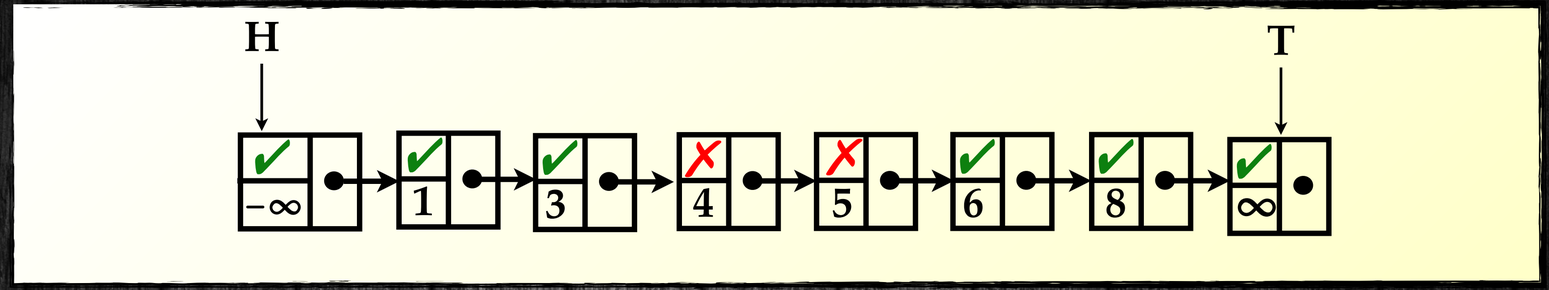



add(2)

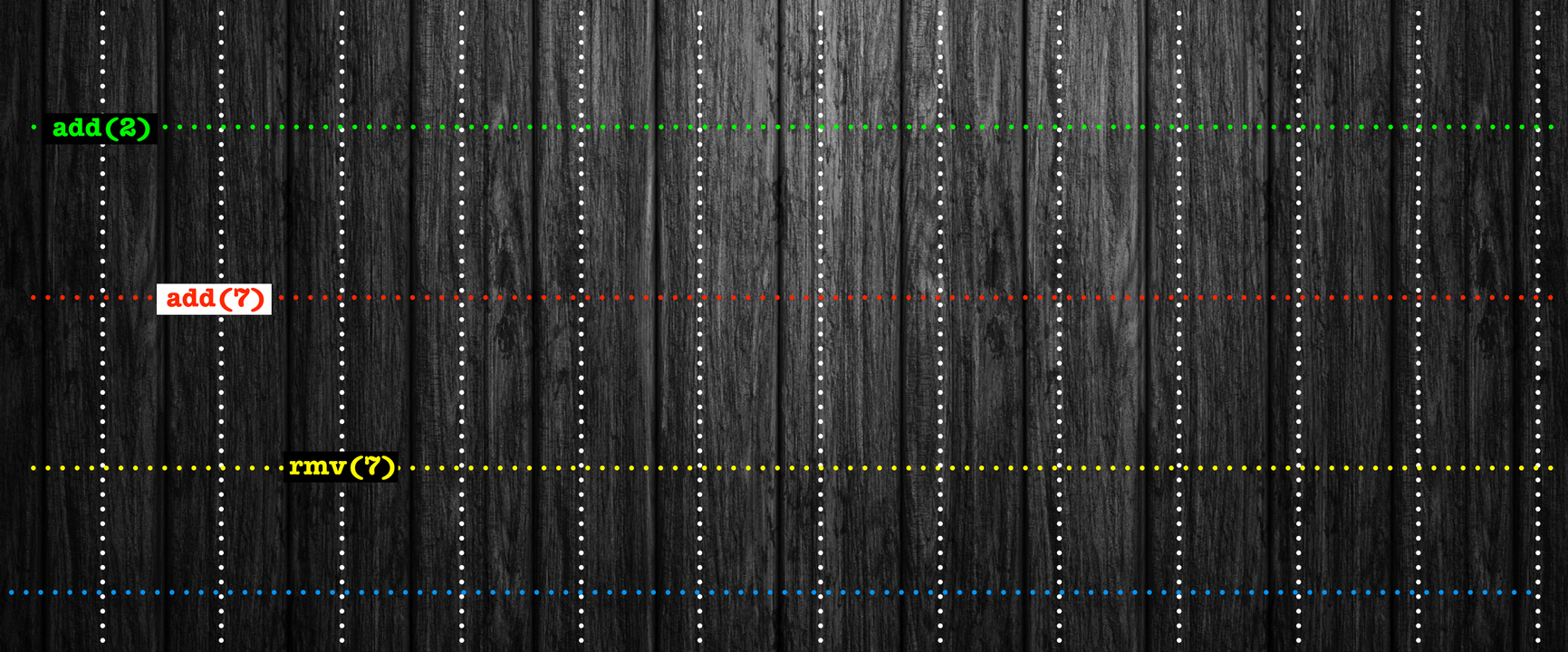
add(7)

rmv(7)

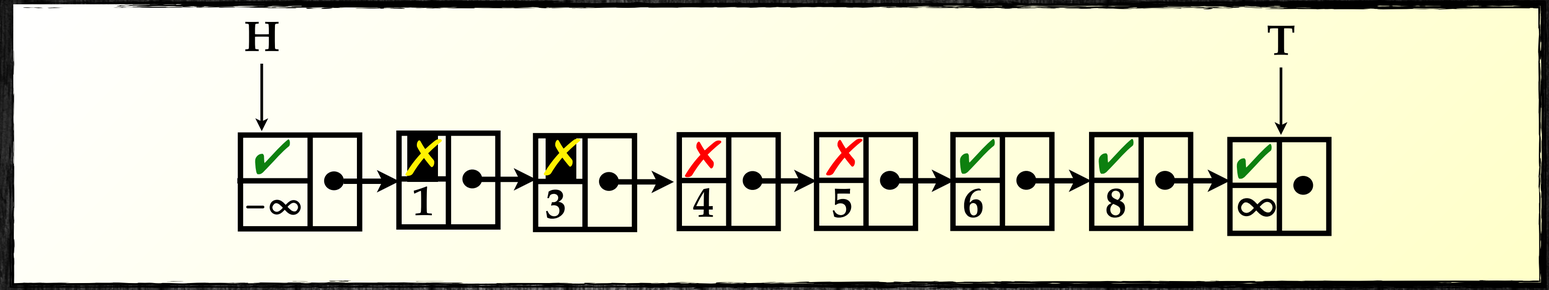
# Sets | Fine-Grained Synchronization | Linearization Policy





# Sets | Fine-Grained Synchronization | Linearization Policy

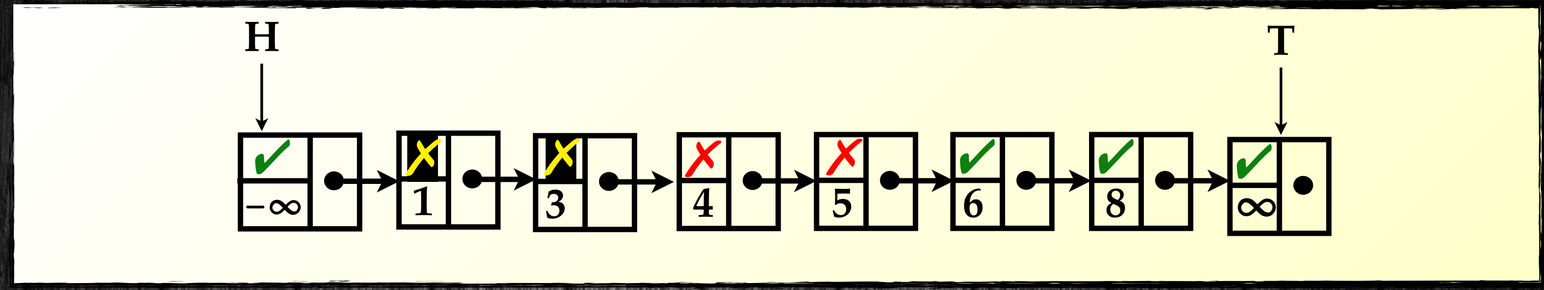





add(2)

add(7)

rmv(7)

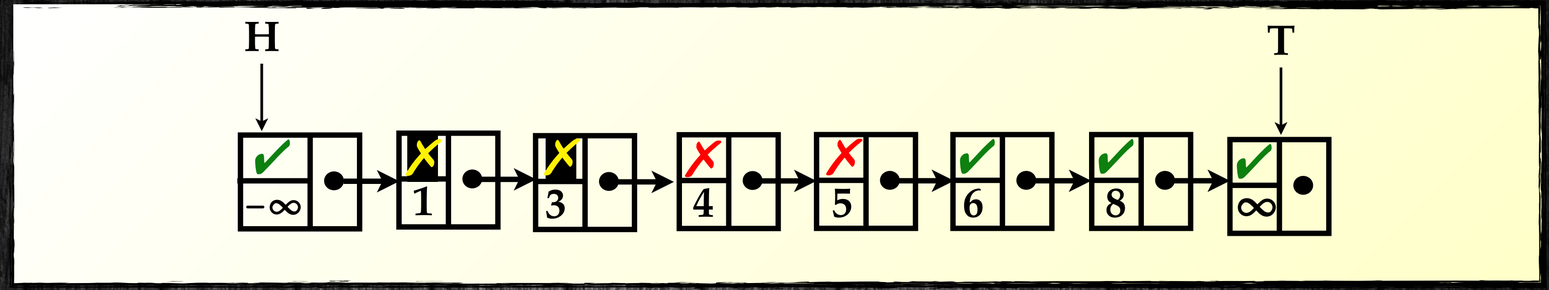



add(2)

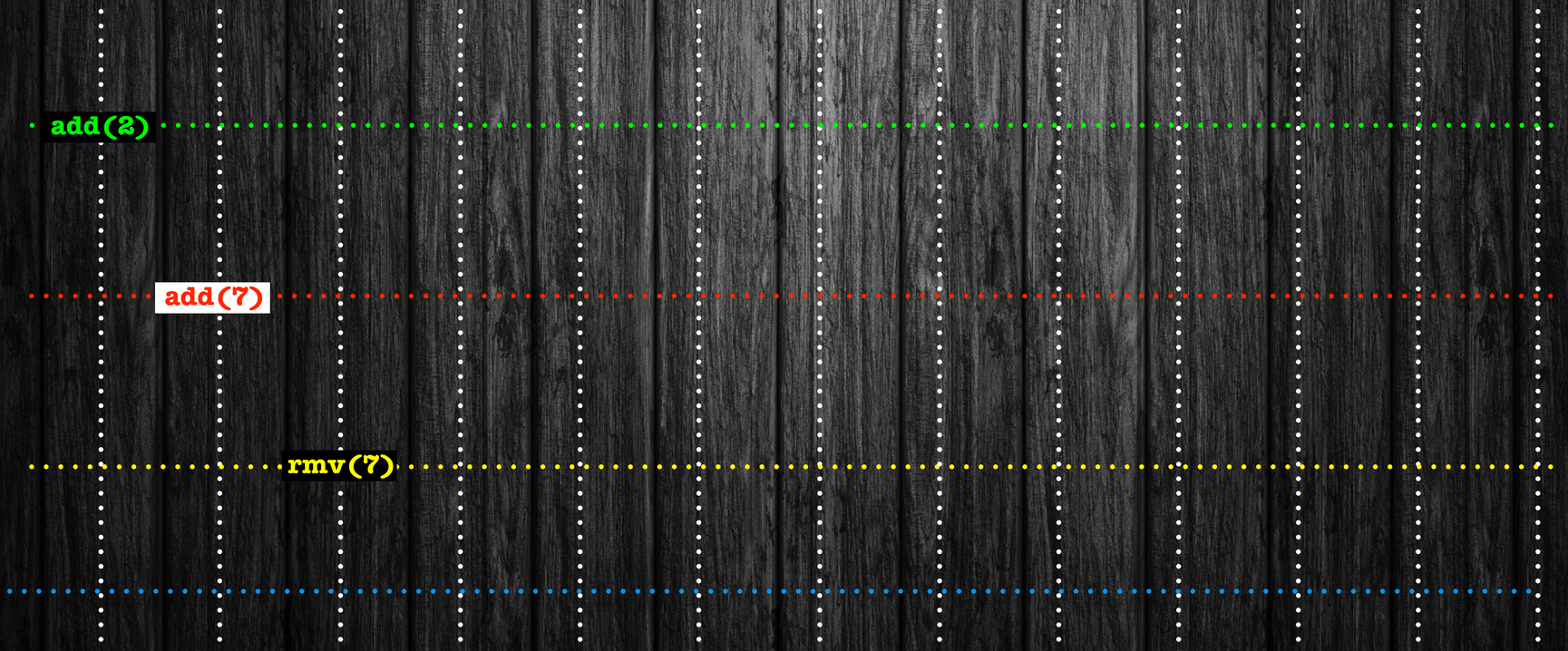
add(7)

rmv(7)

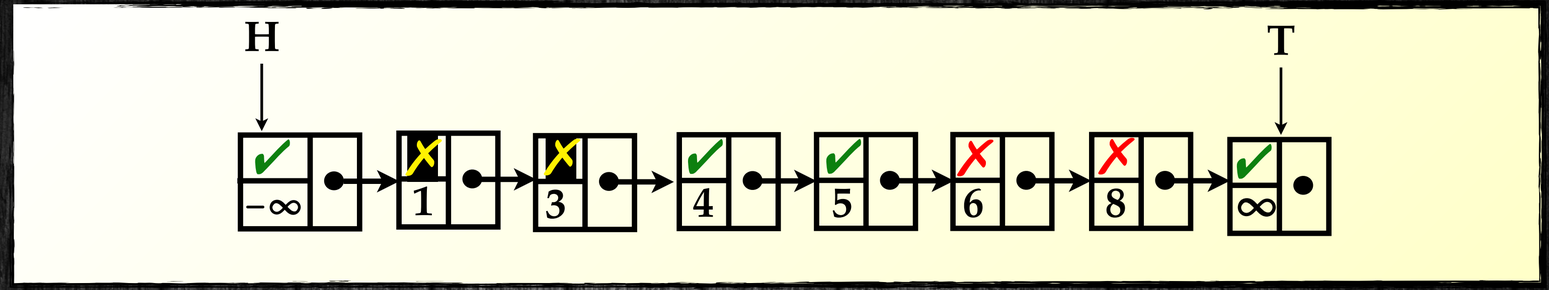
# Sets | Fine-Grained Synchronization | Linearization Policy





# Sets | Fine-Grained Synchronization | Linearization Policy



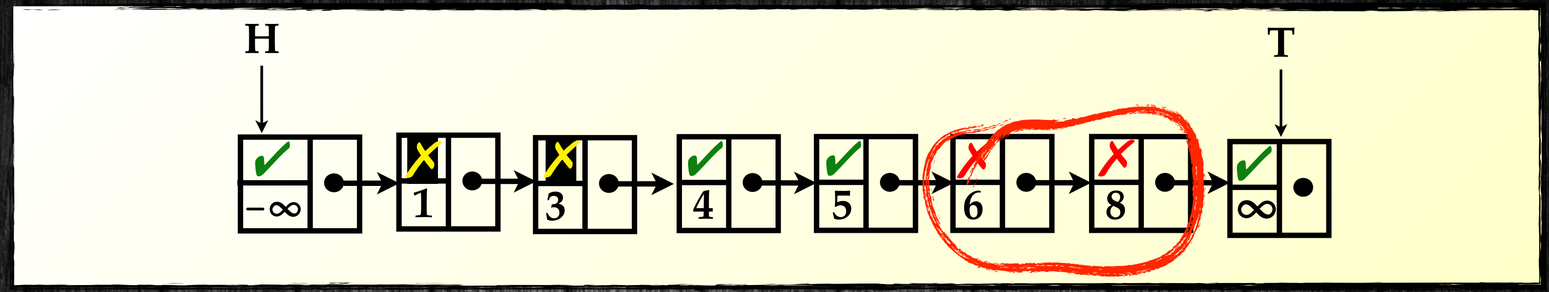



add(2)

add(7)

rmv(7)

# Sets | Fine-Grained Synchronization | Linearization Policy



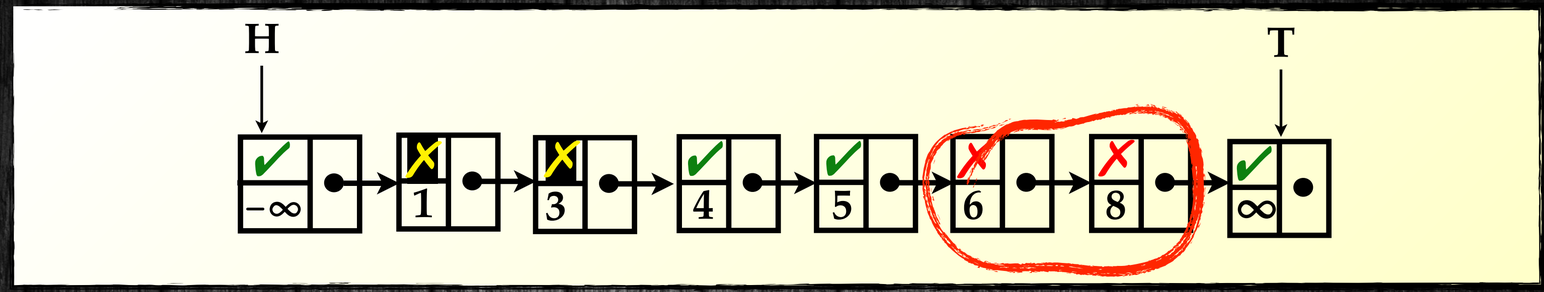



add(2)

add(7)

rmv(7)

# Sets | Fine-Grained Synchronization | Linearization Policy




LP

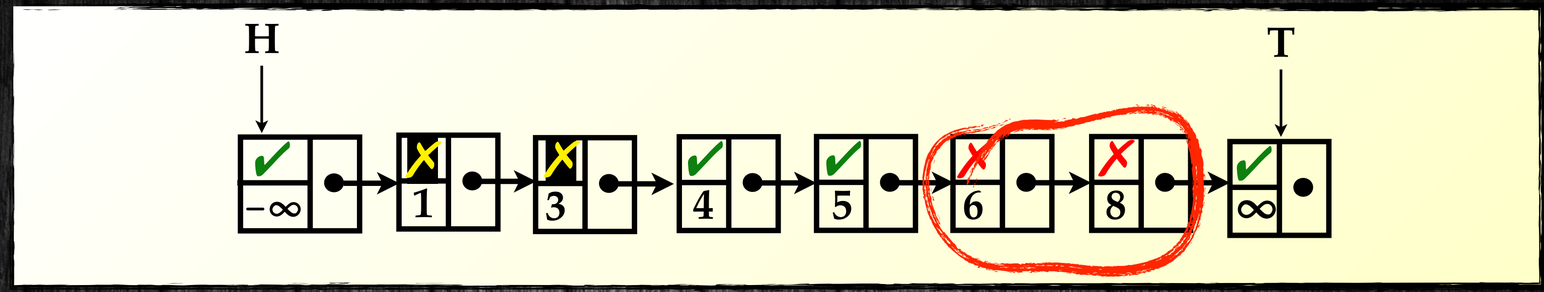
"time point at which I find a key ≥ input key"

add(2)

add(7)

rmv(7)

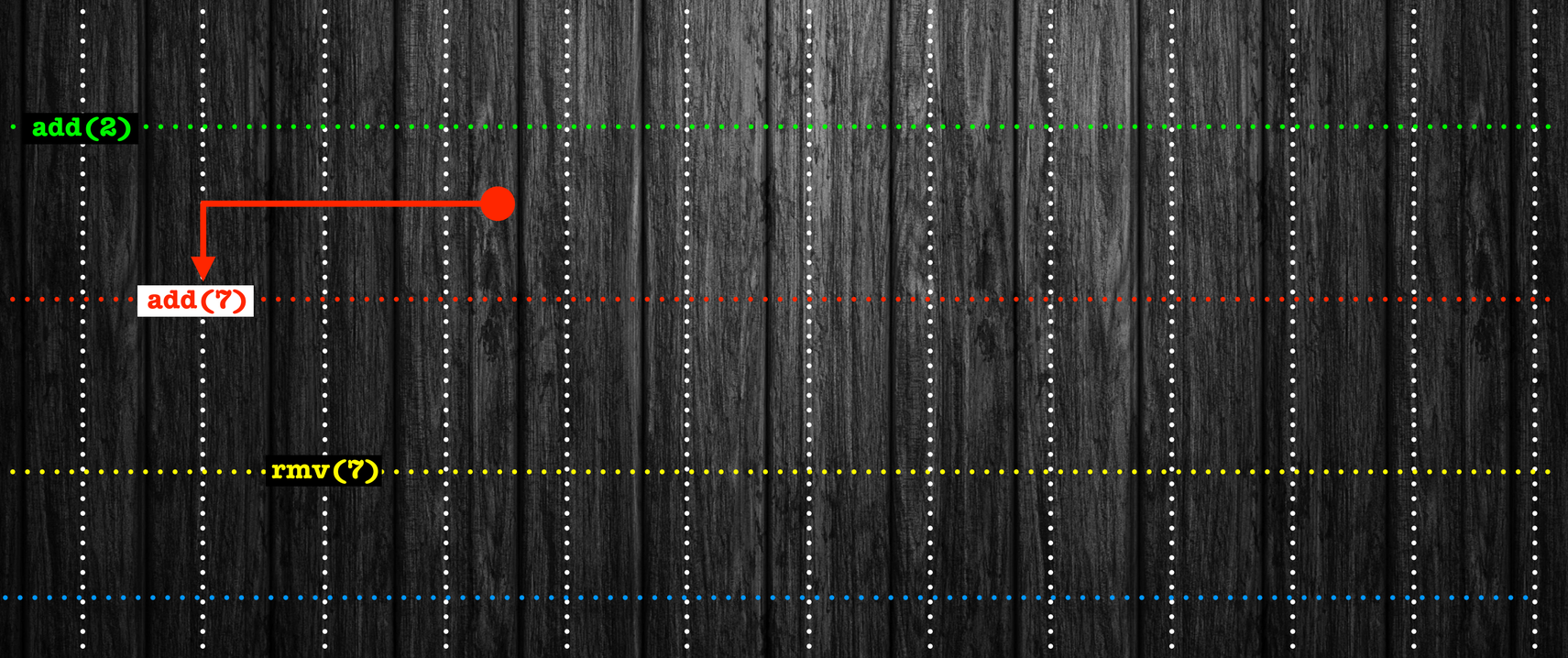
# Sets | Fine-Grained Synchronization | Linearization Policy



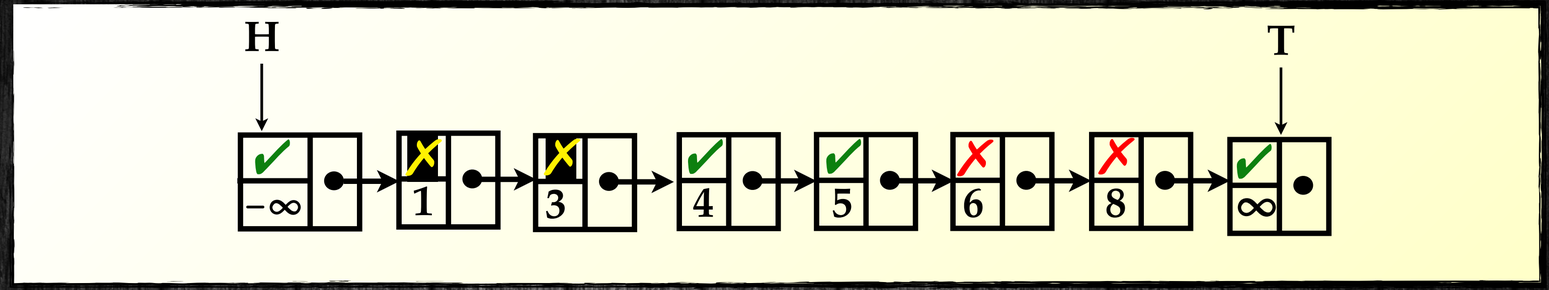


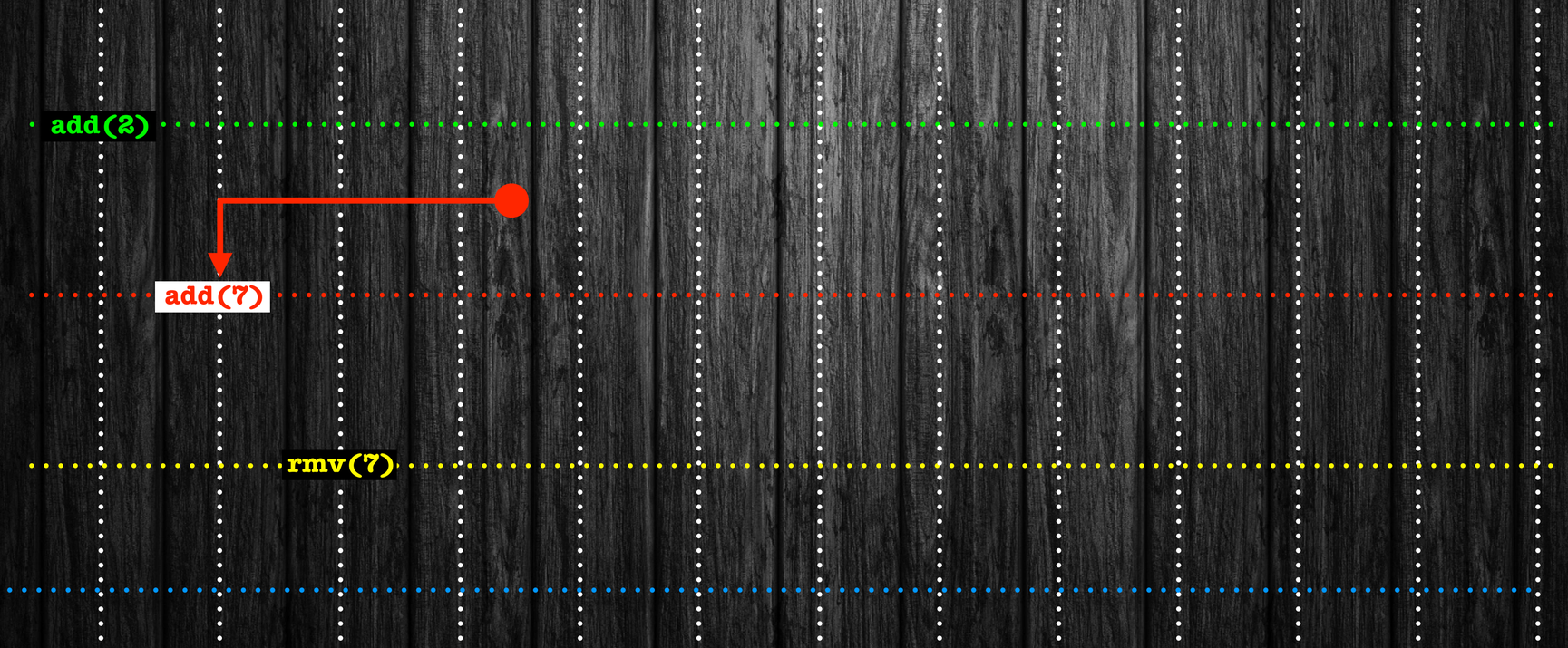

LP

**“time point at which I find a key ≥ input key”**

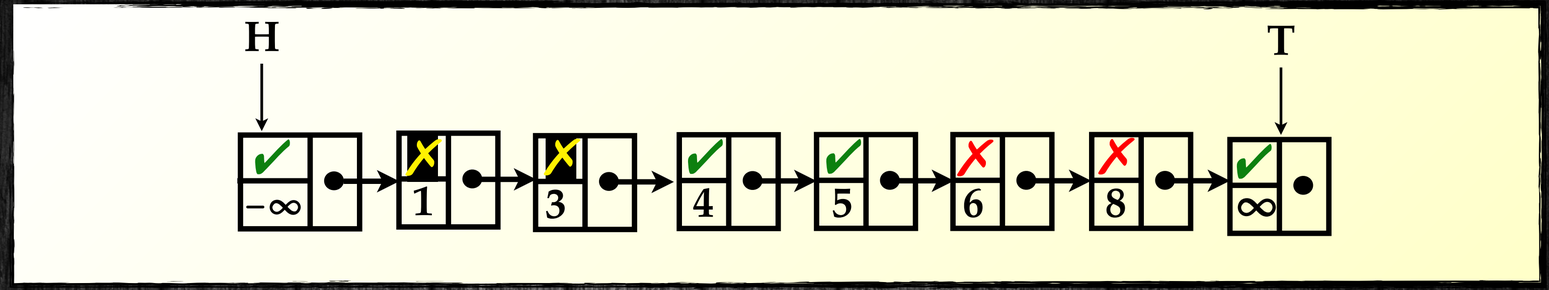


# Sets | Fine-Grained Synchronization | Linearization Policy

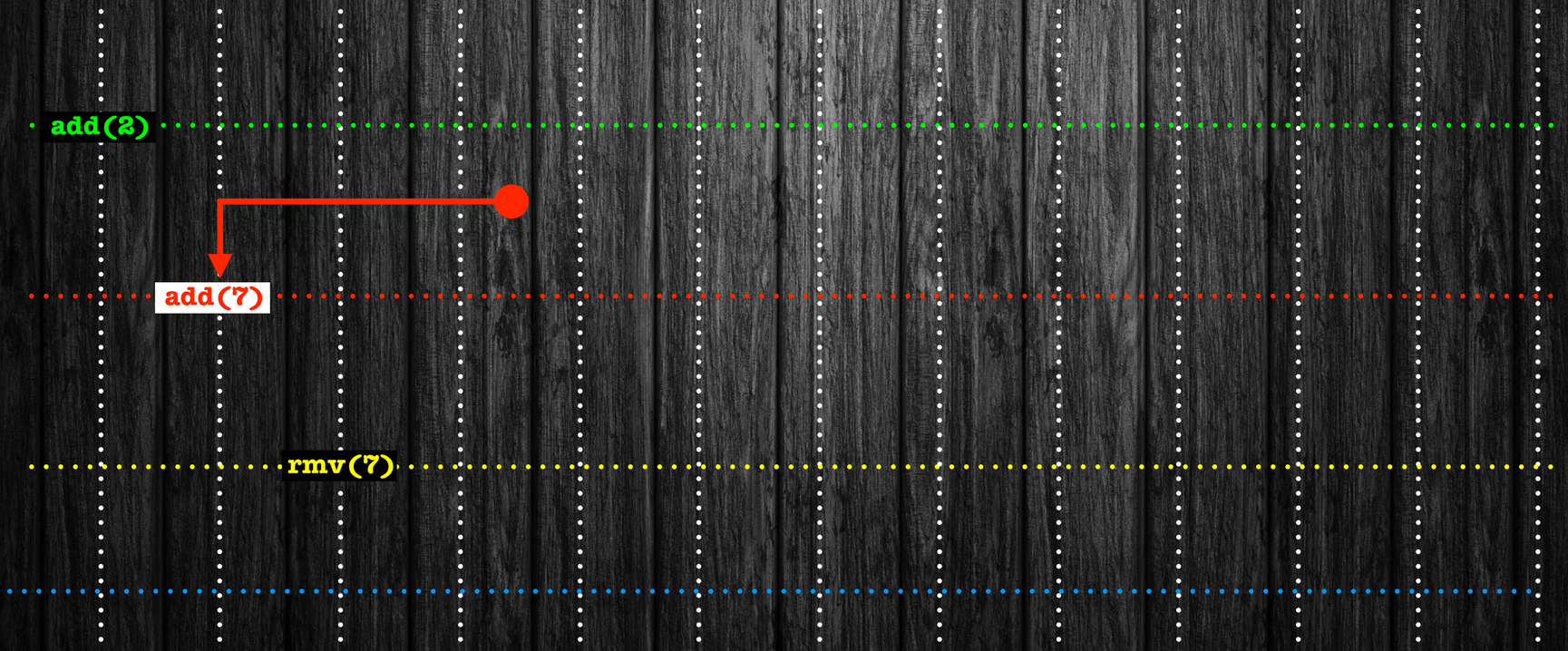


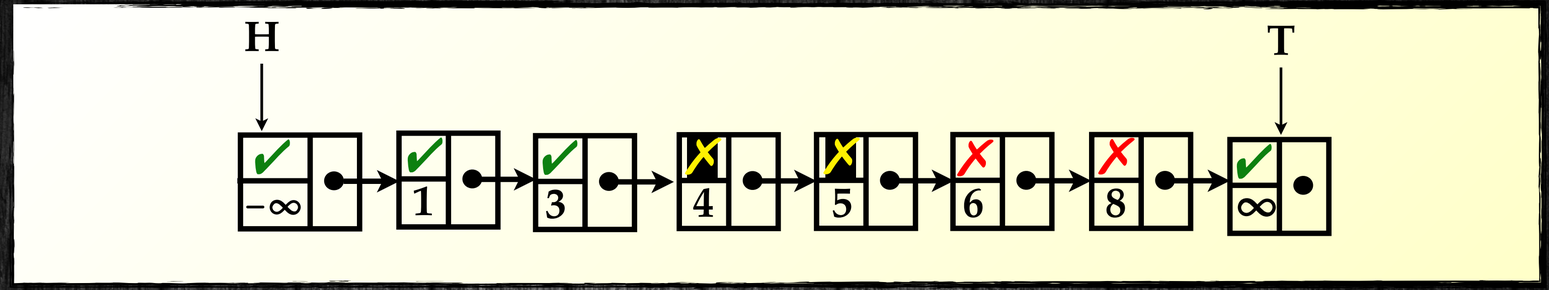
# Sets | Fine-Grained Synchronization | Linearization Policy



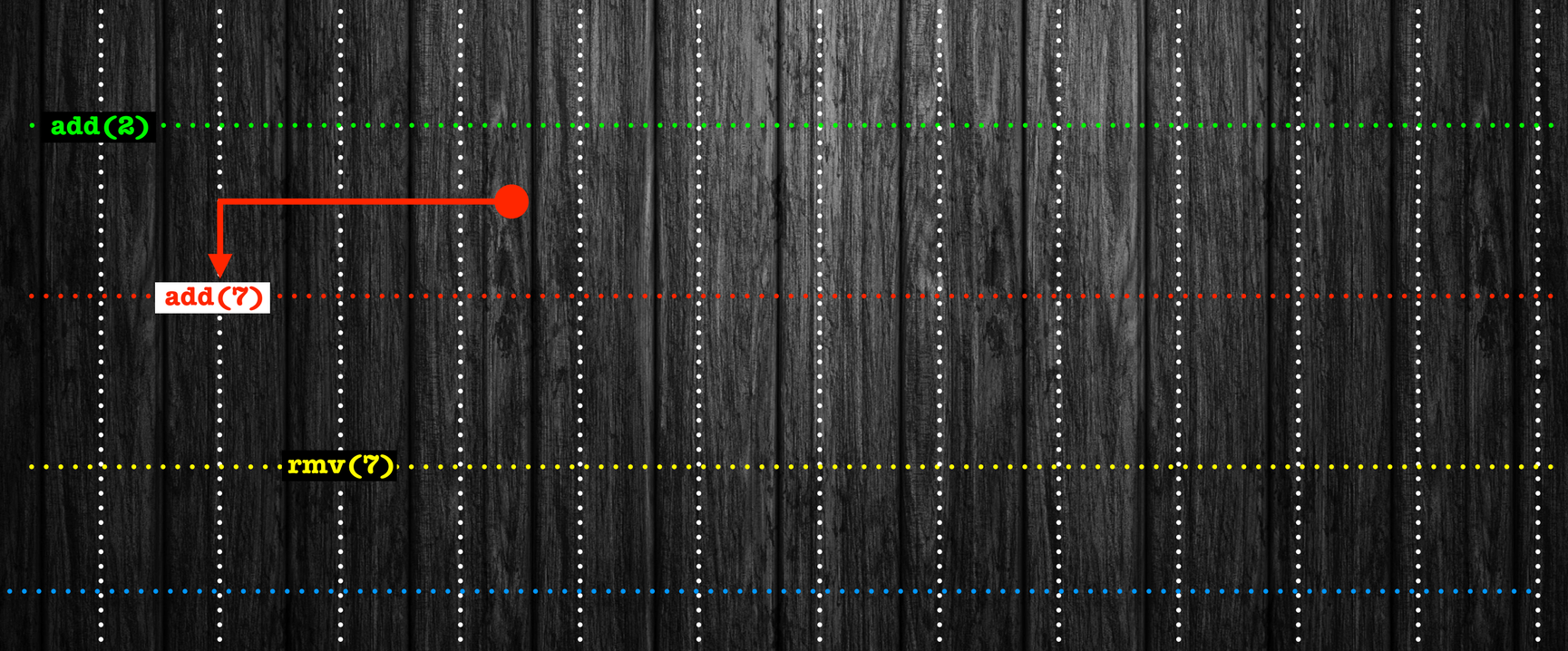


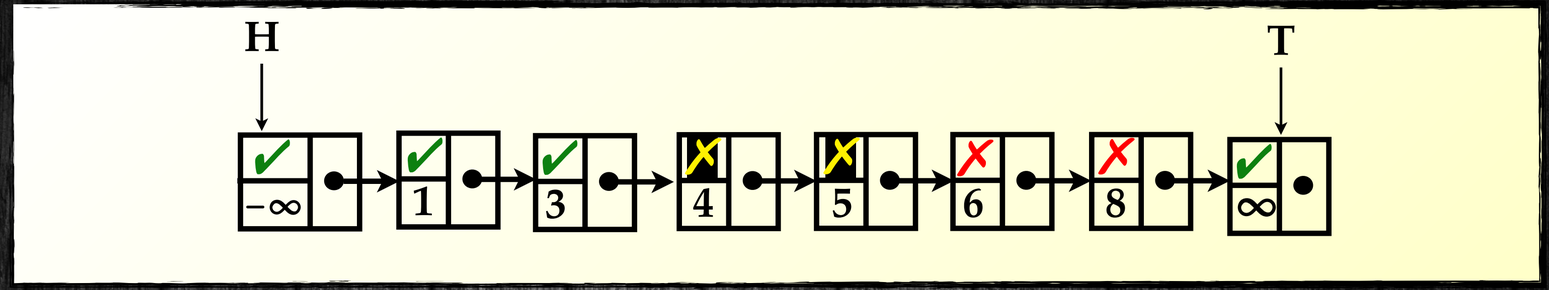



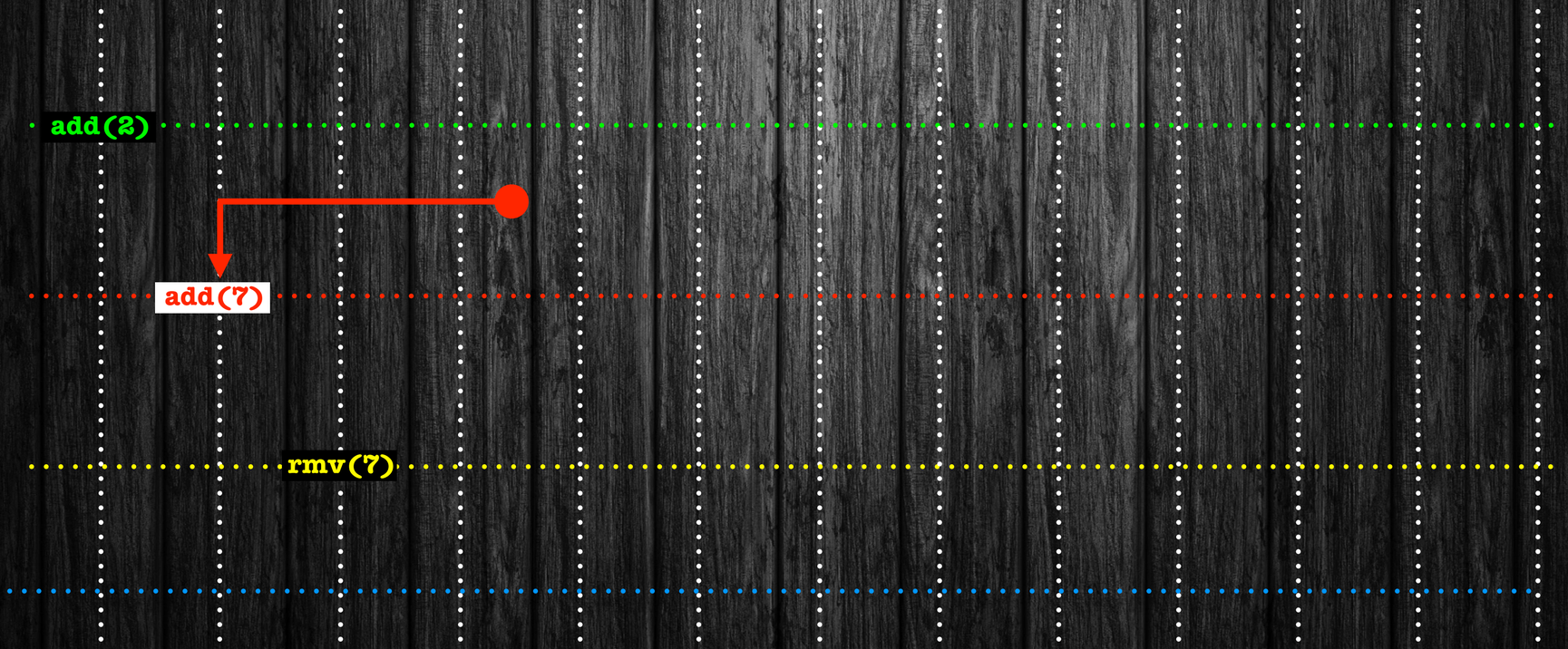
# Sets | Fine-Grained Synchronization | Linearization Policy



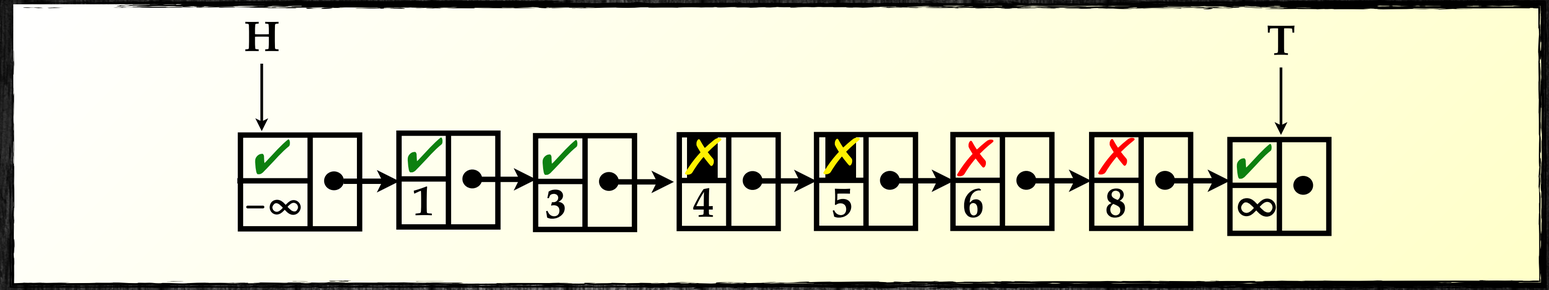


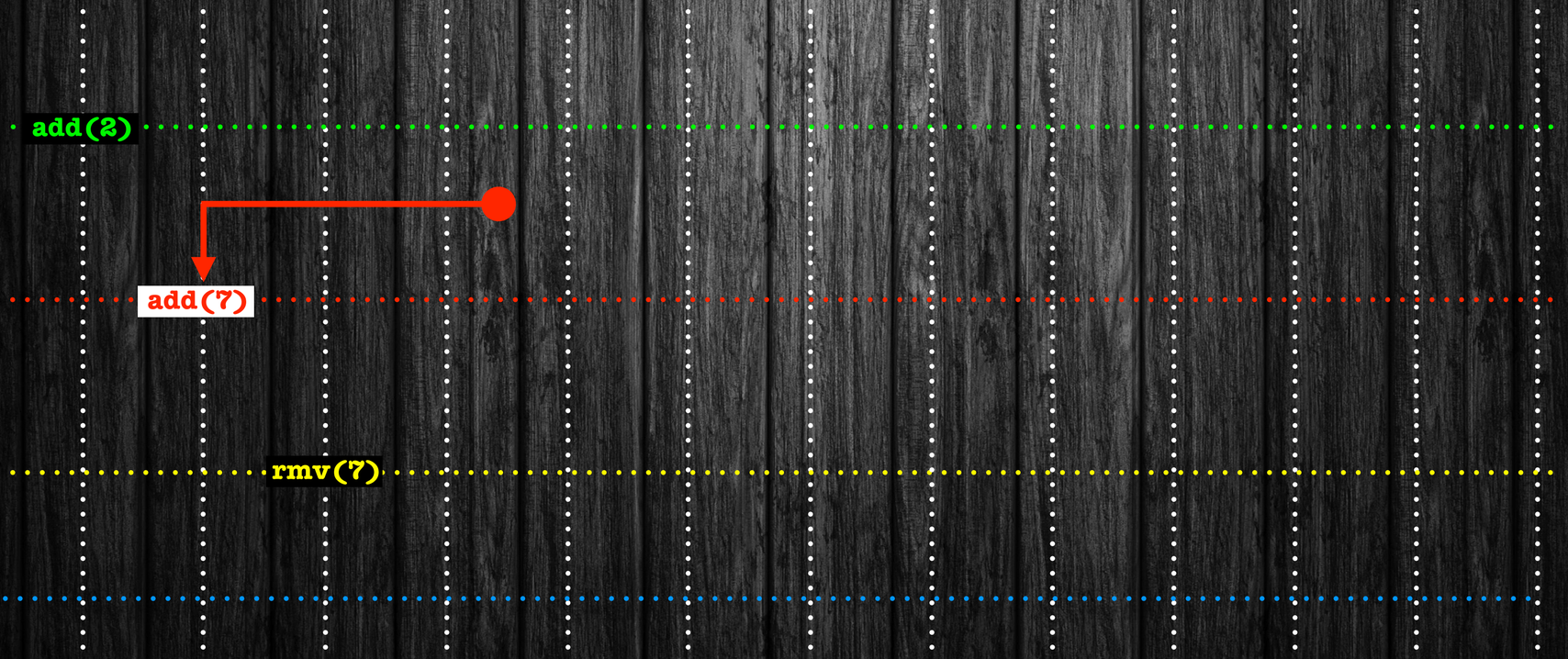

H
T
✓
−∞
✓
1
✓
3
✗
4
✗
5
✗
6
✗
8
✓
∞


add(2)
add(7)
rmv(7)

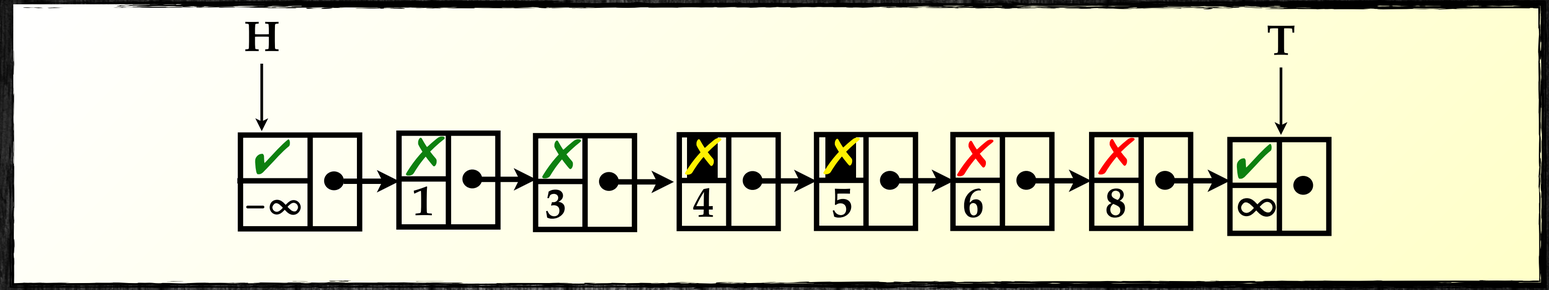
# Sets | Fine-Grained Synchronization | Linearization Policy



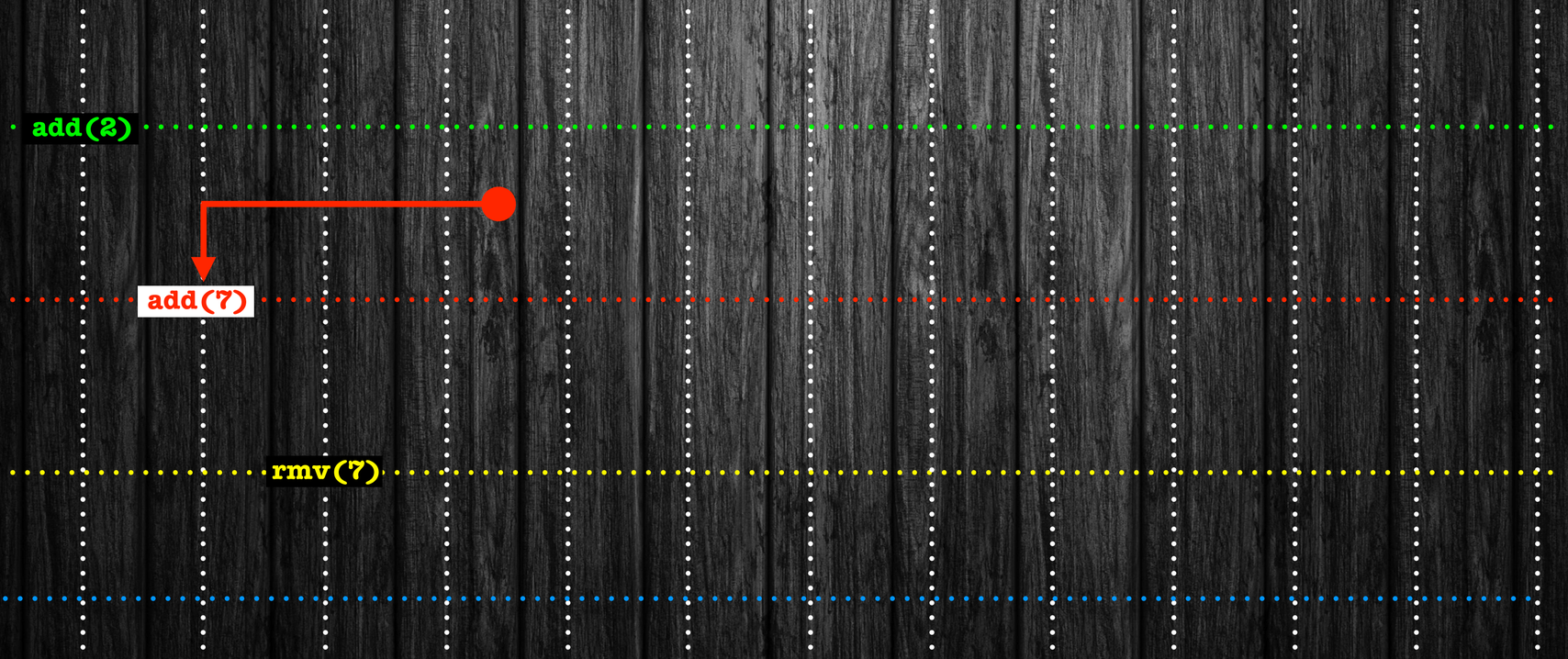


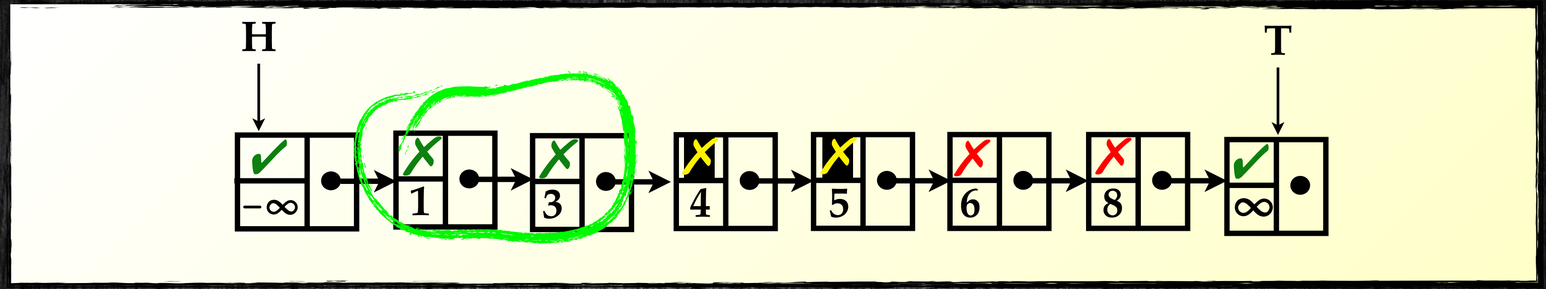
# Sets | Fine-Grained Synchronization | Linearization Policy



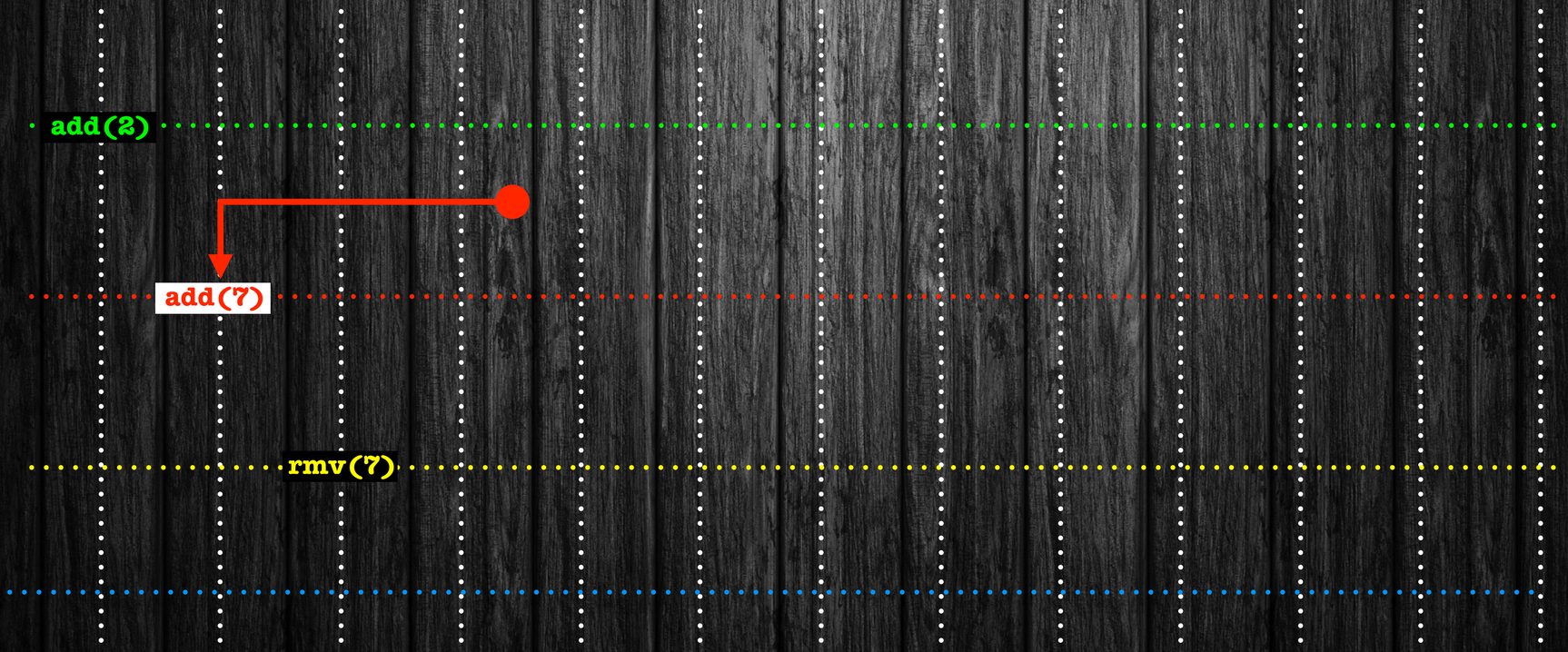


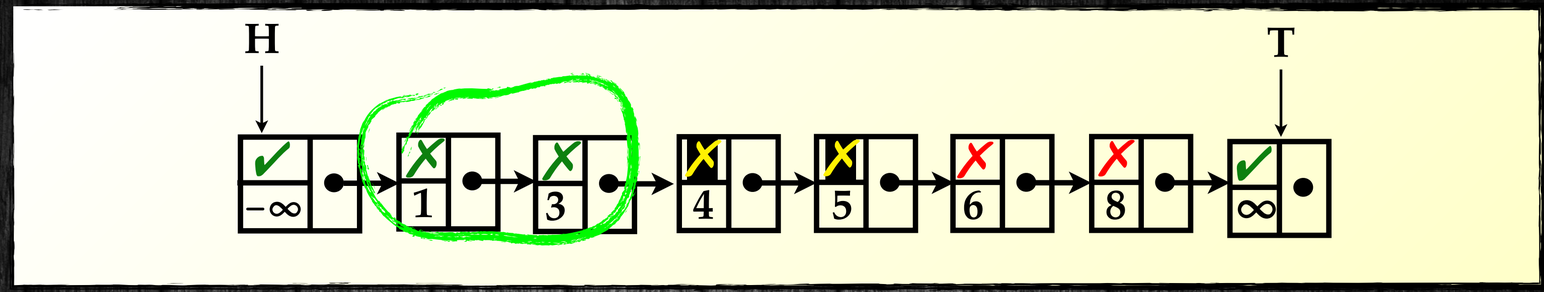
# Sets | Fine-Grained Synchronization | Linearization Policy





# Sets | Fine-Grained Synchronization | Linearization Policy






LP

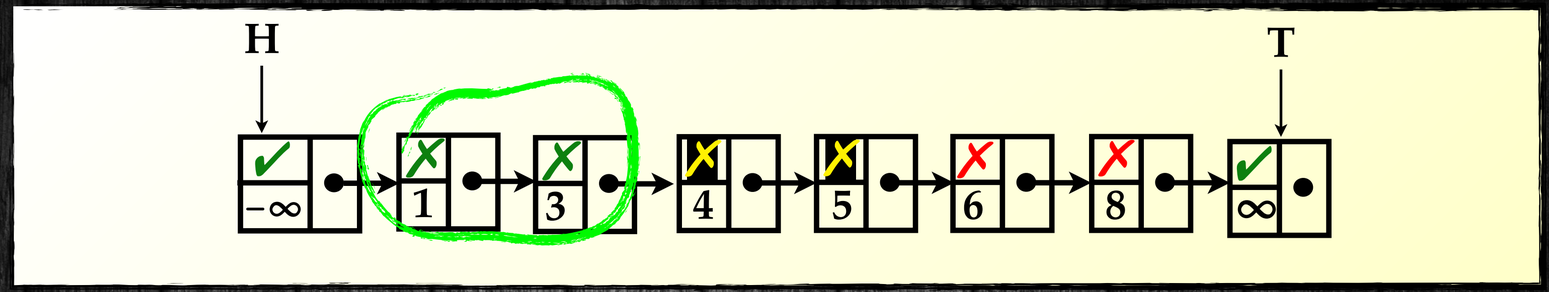
"time point at which I find a key ≥ input key"

add(2)

add(7)

rmv(7)

# Sets | Fine-Grained Synchronization | Linearization Policy






LP

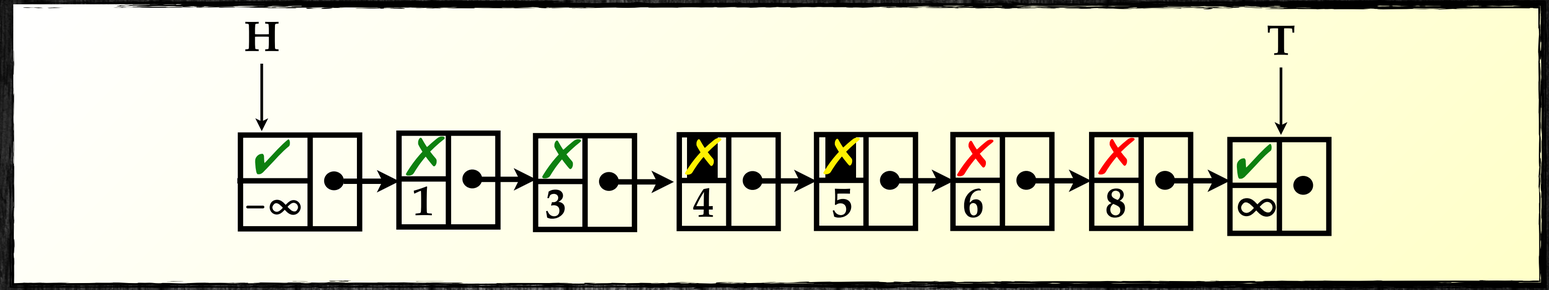
"time point at which I find a key ≥ input key"

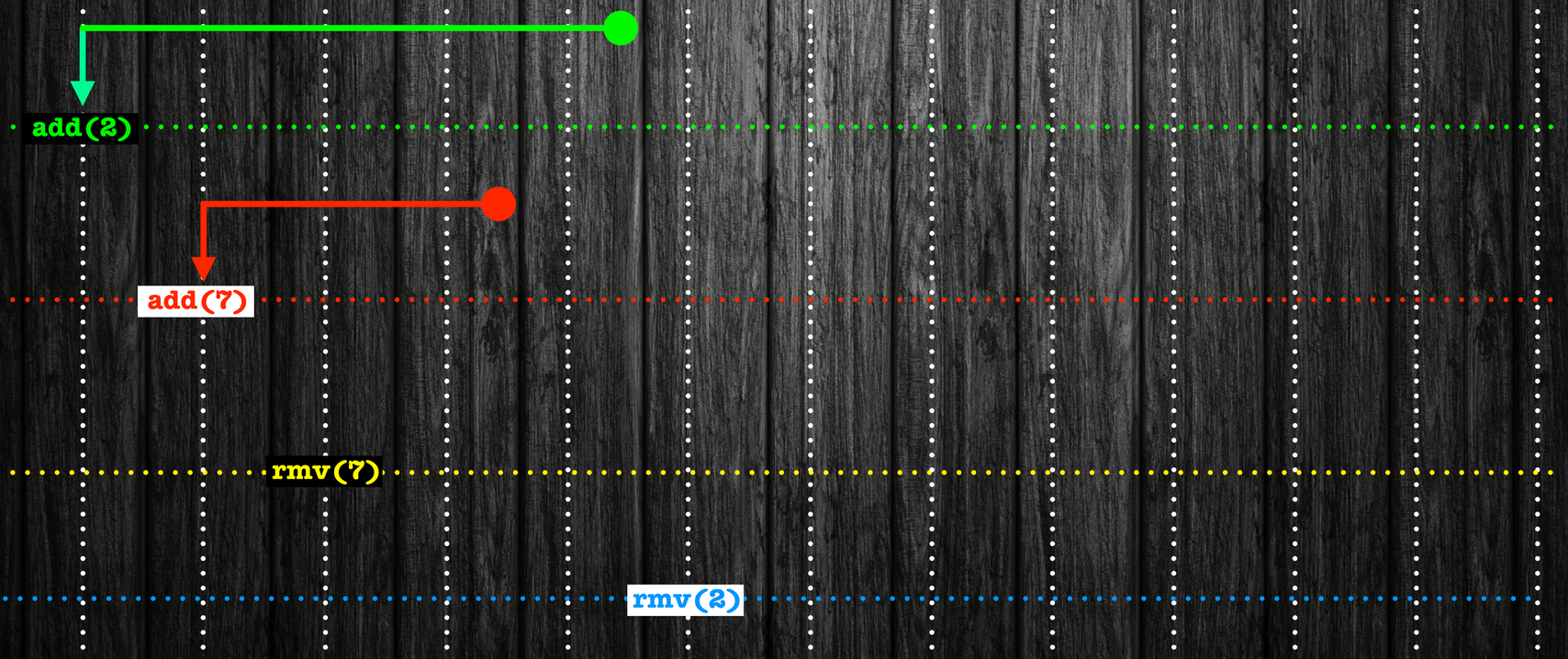
add(2)

add(7)

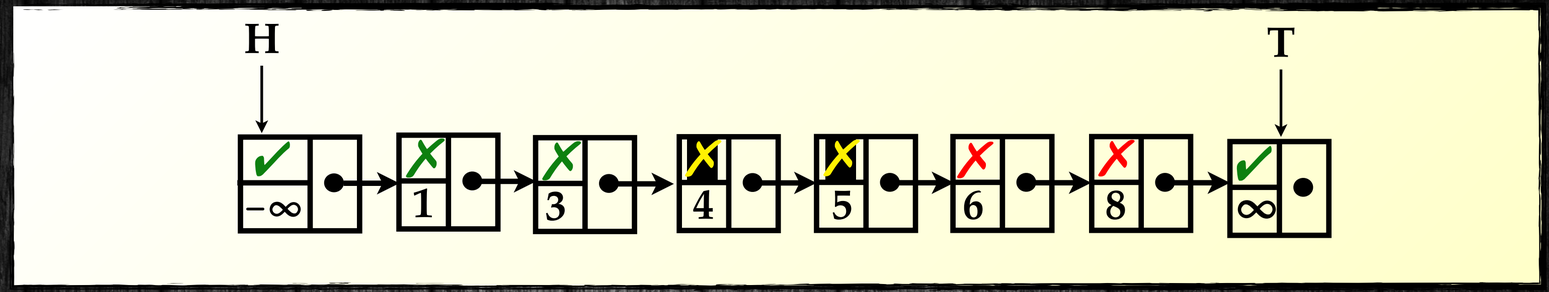
rmv(7)

# Sets | Fine-Grained Synchronization | Linearization Policy

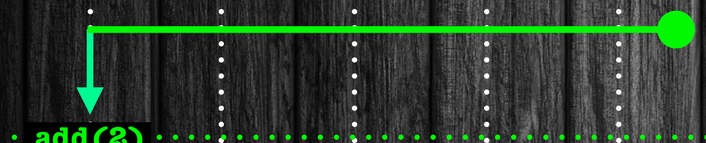


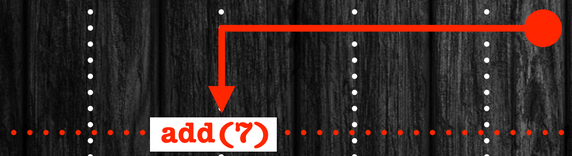



# Sets | Fine-Grained Synchronization | Linearization Policy

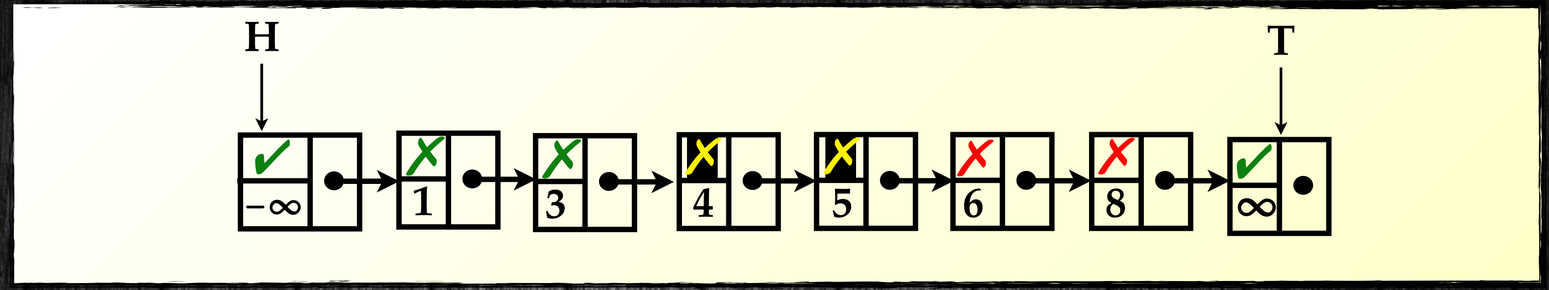


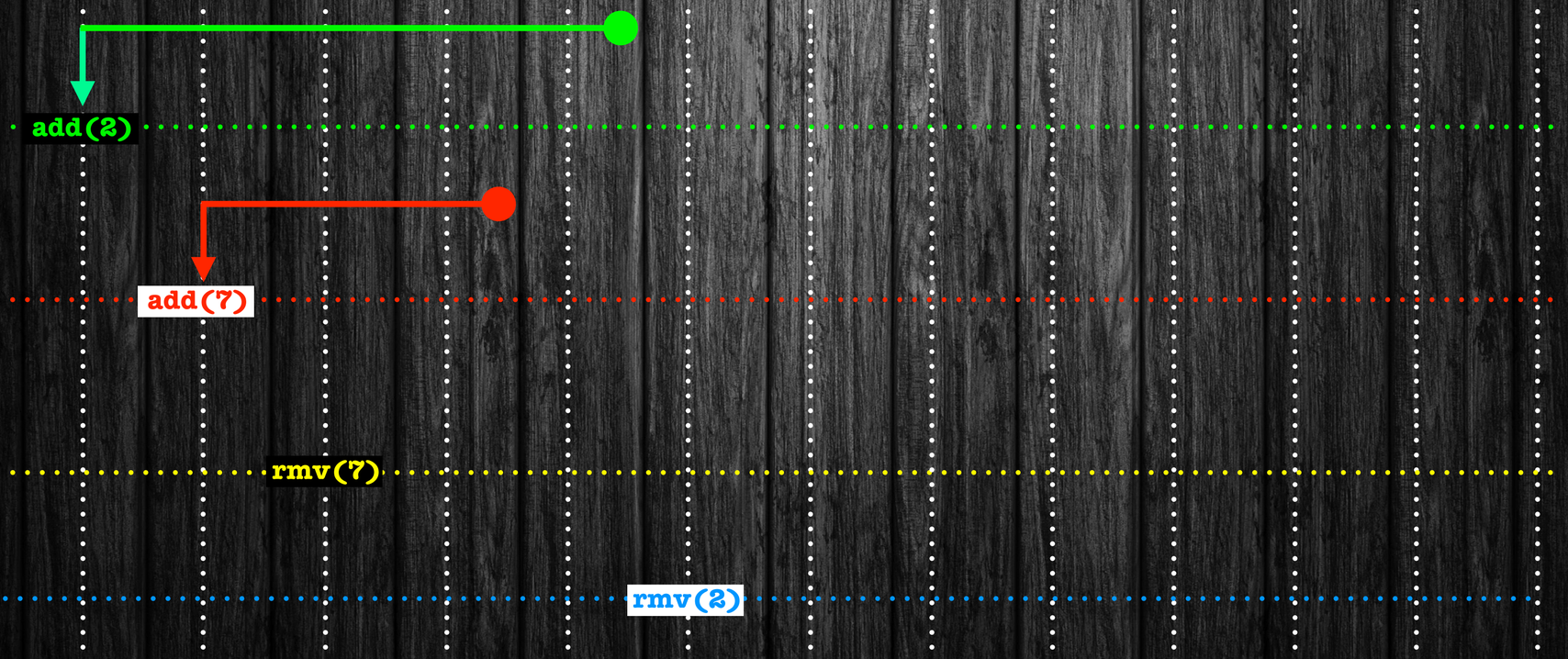


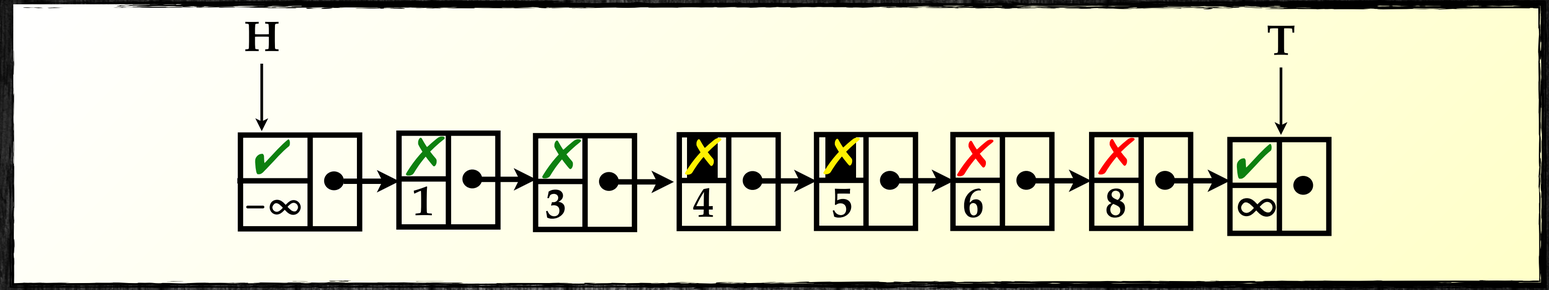


# Sets | Fine-Grained Synchronization | Linearization Policy

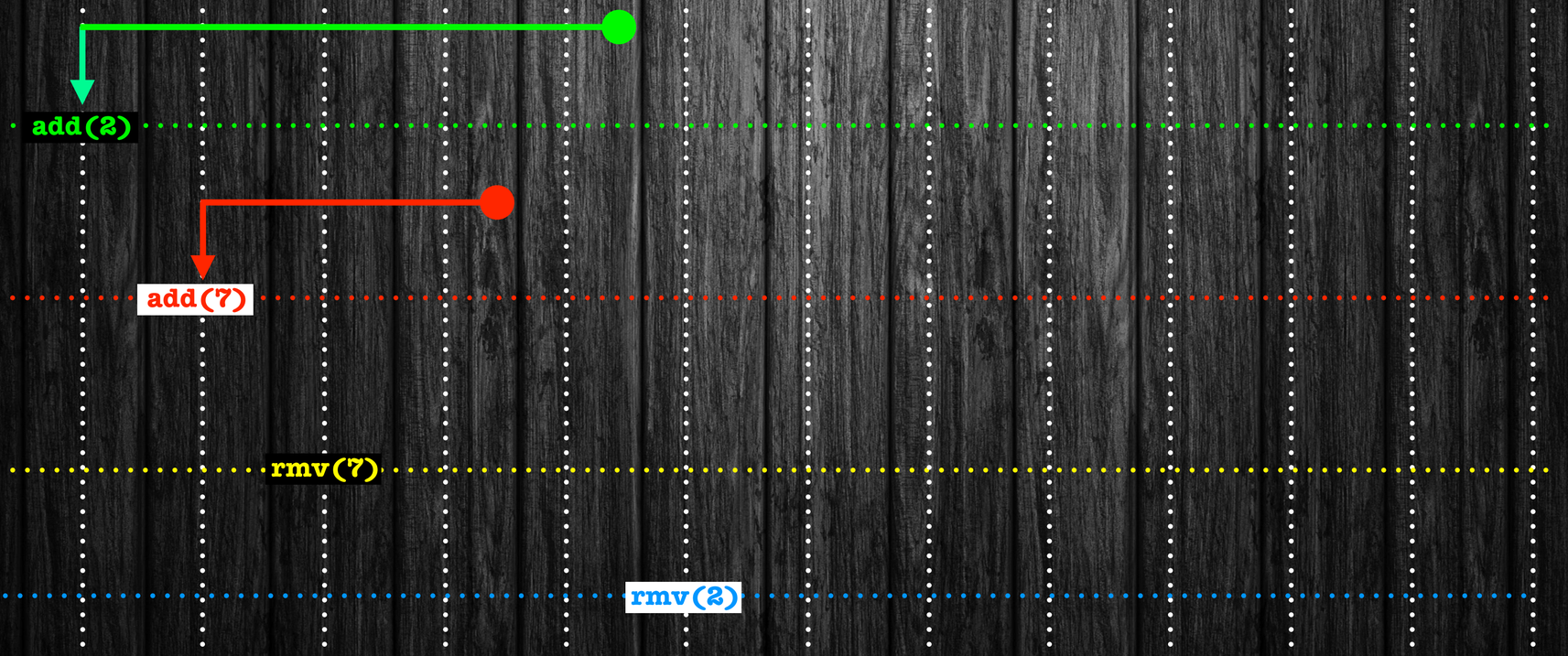


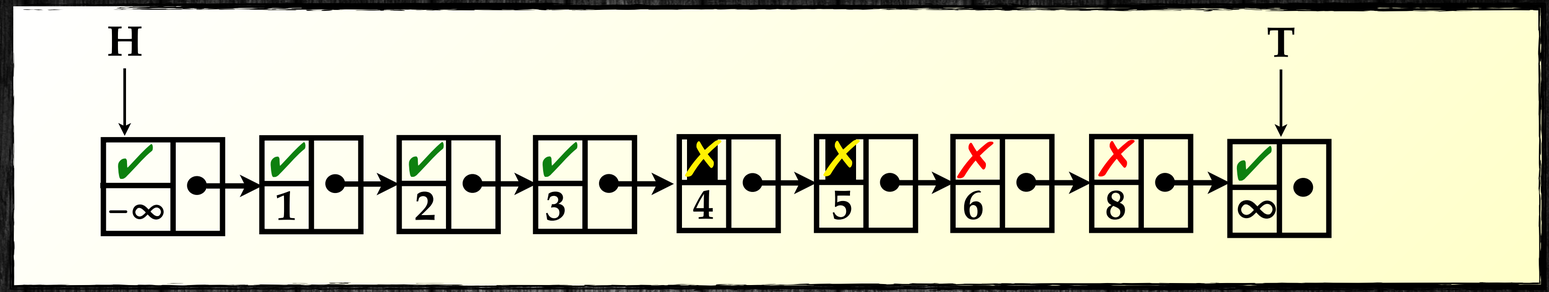
# Sets | Fine-Grained Synchronization | Linearization Policy



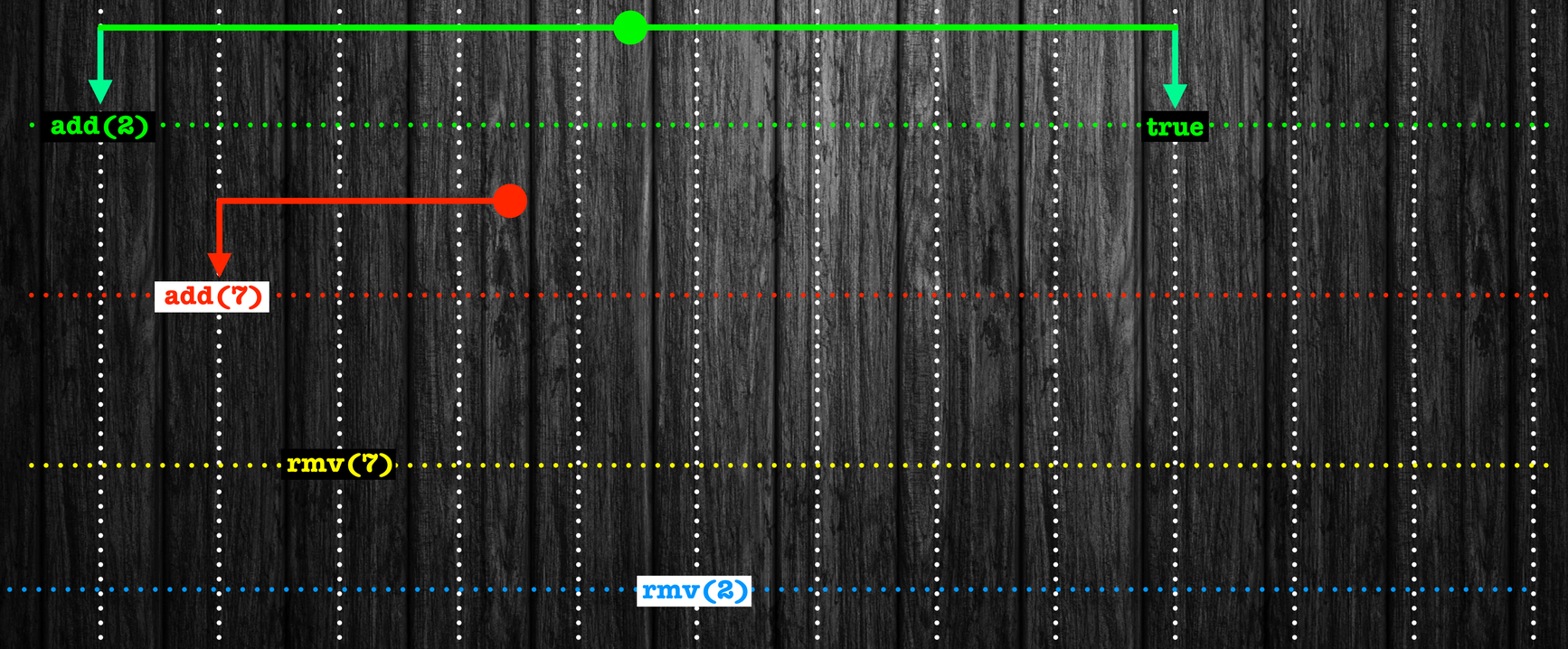


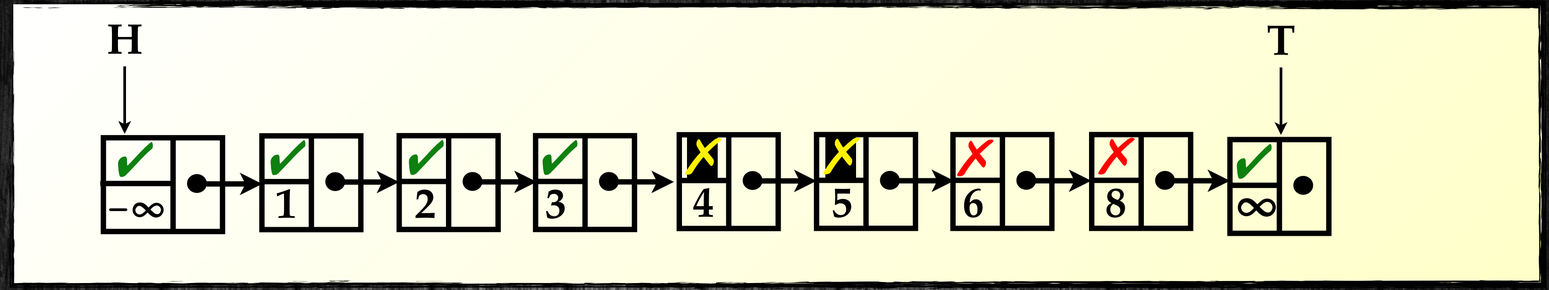



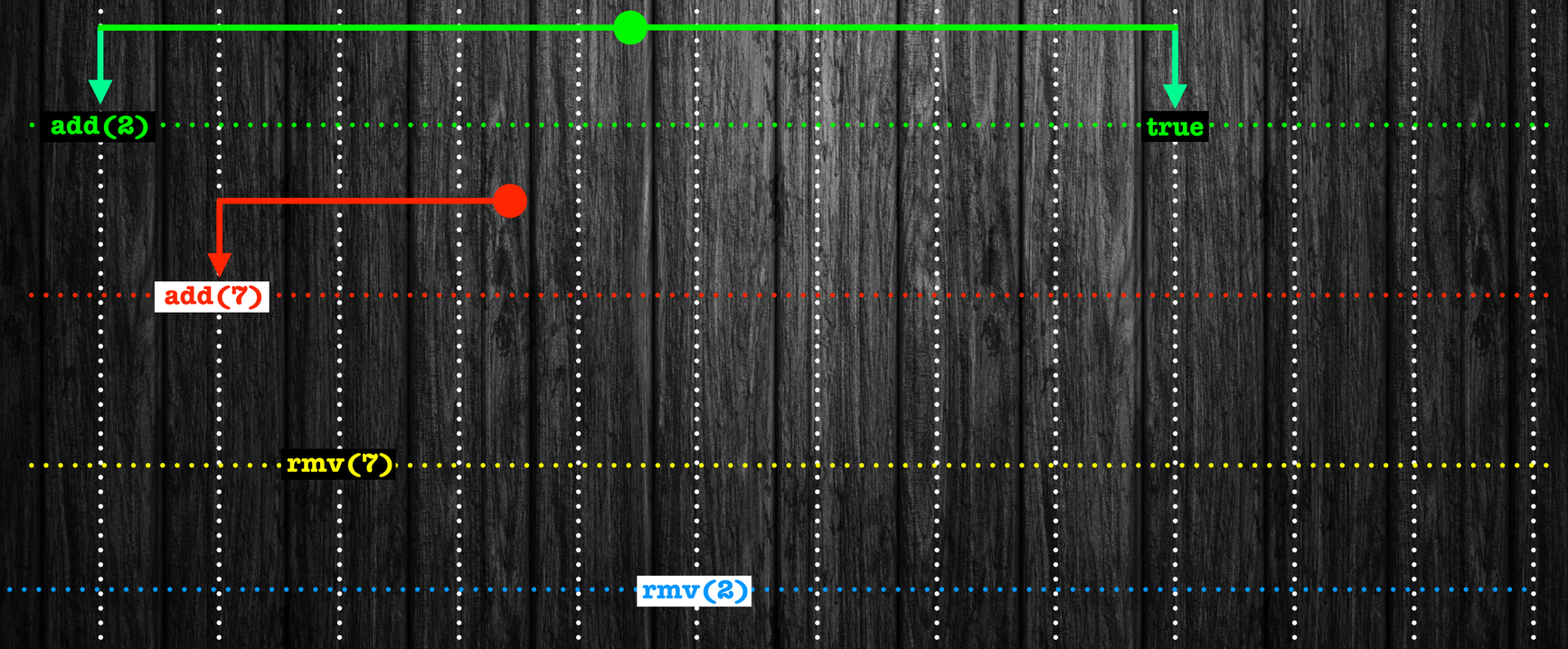
# Sets | Fine-Grained Synchronization | Linearization Policy



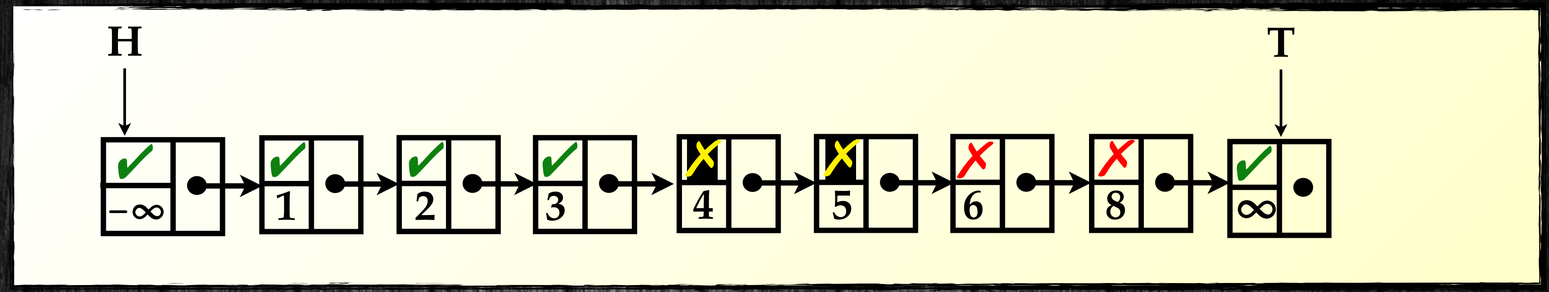


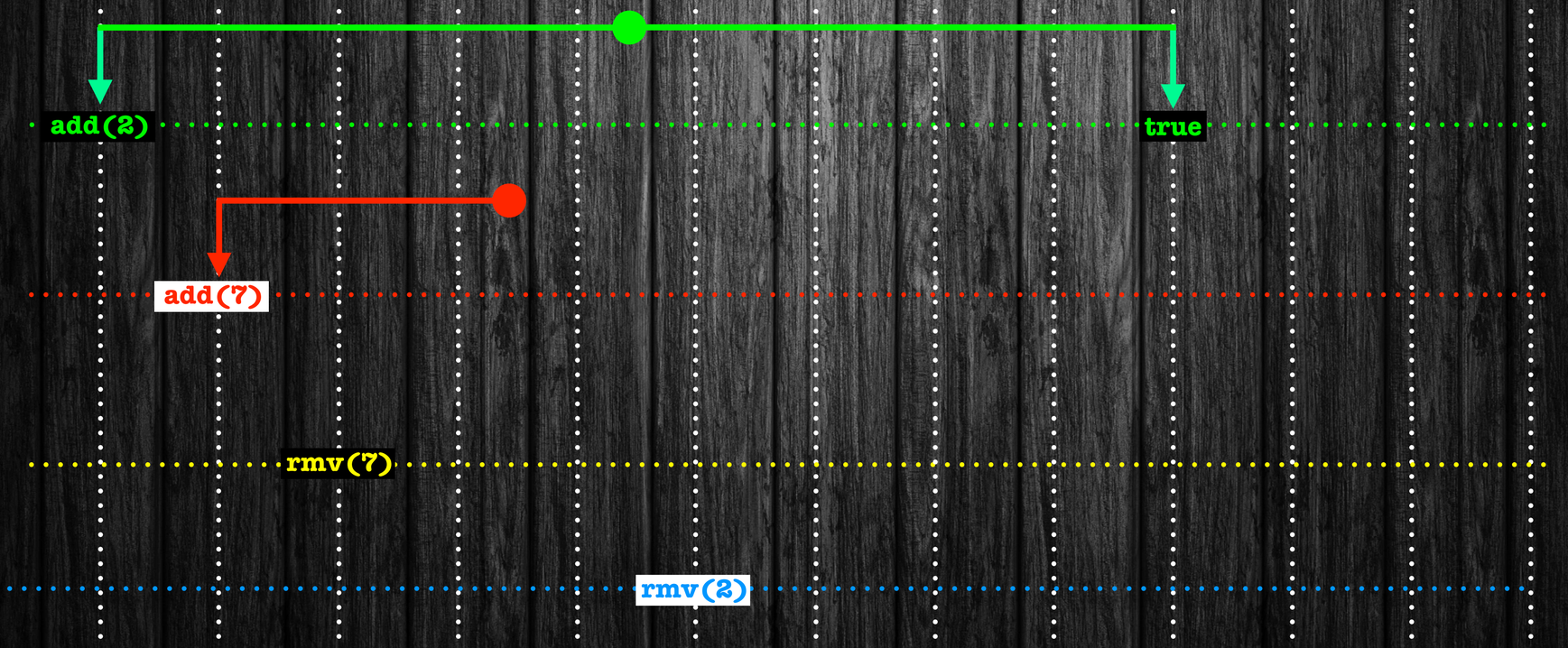

H
T
✓
−∞
✓
1
✓
2
✓
3
✗
4
✗
5
✗
6
✗
8
✓
∞


add(2)
true
add(7)
rmv(7)
rmv(2)

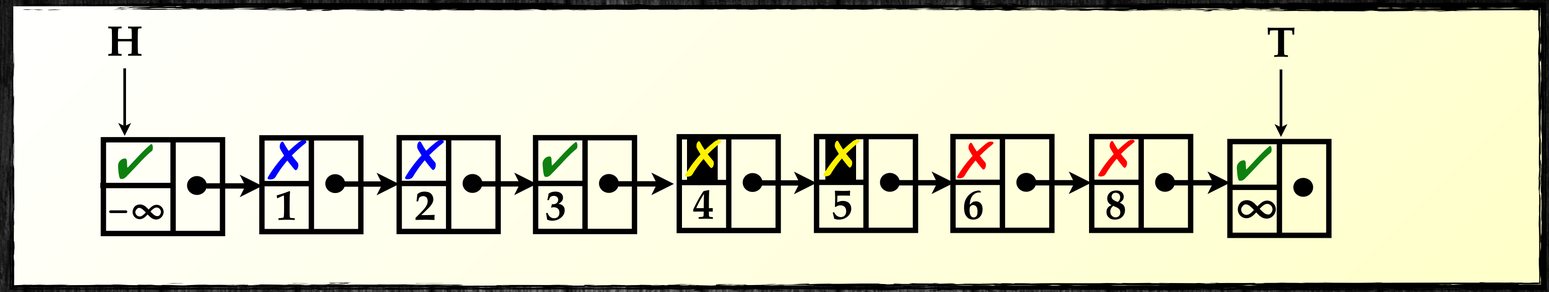
# Sets | Fine-Grained Synchronization | Linearization Policy

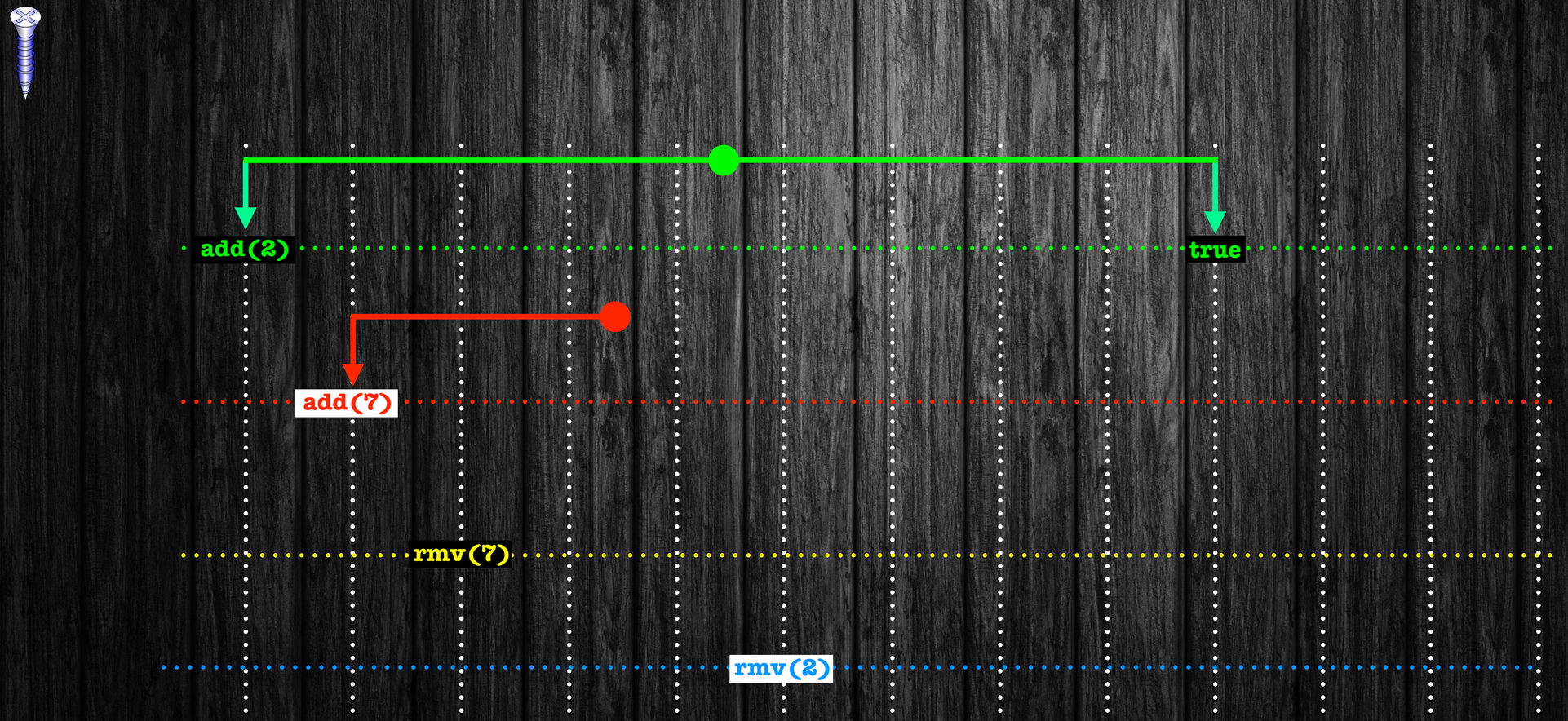




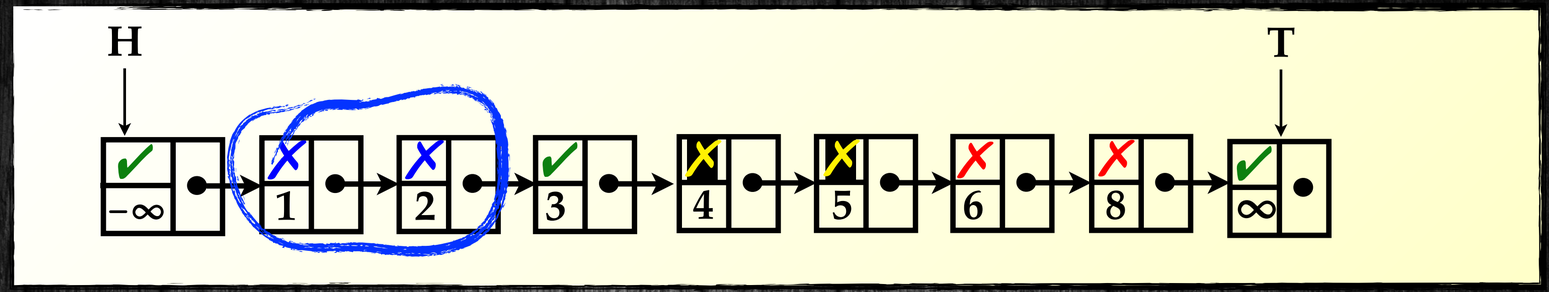



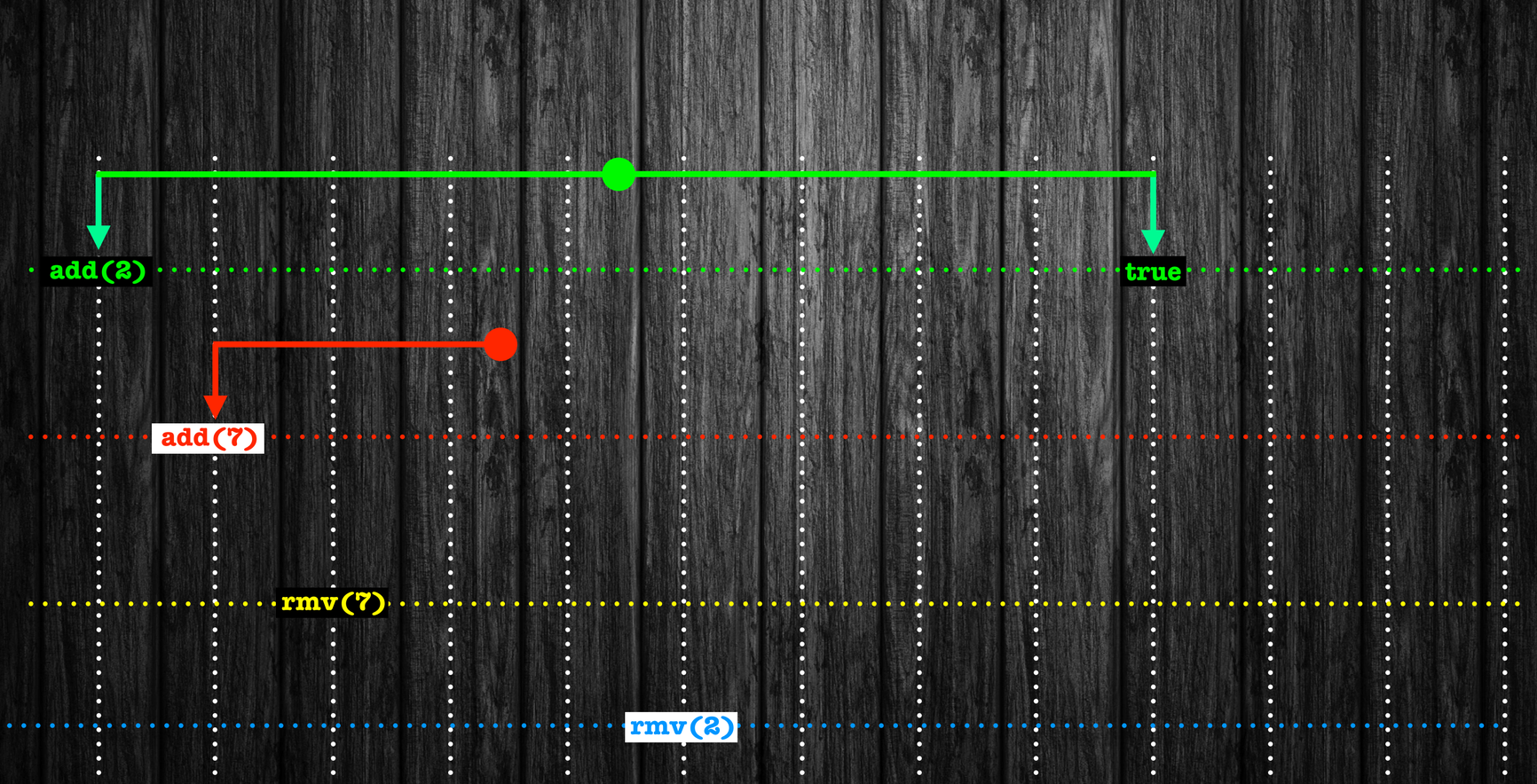
# Sets | Fine-Grained Synchronization | Linearization Policy



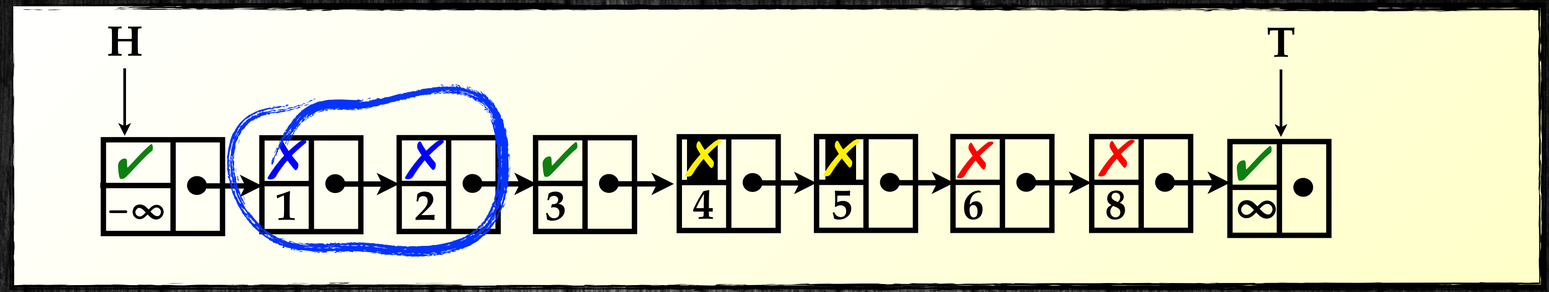



# Sets | Fine-Grained Synchronization | Linearization Policy



# Sets | Fine-Grained Synchronization | Linearization Policy






LP

"time point at which I find a key ≥ input key"

add(2)

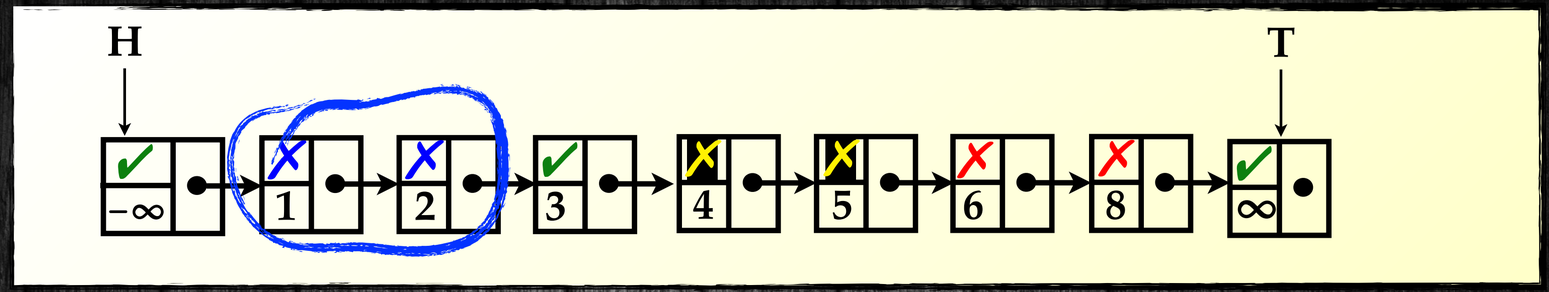
true

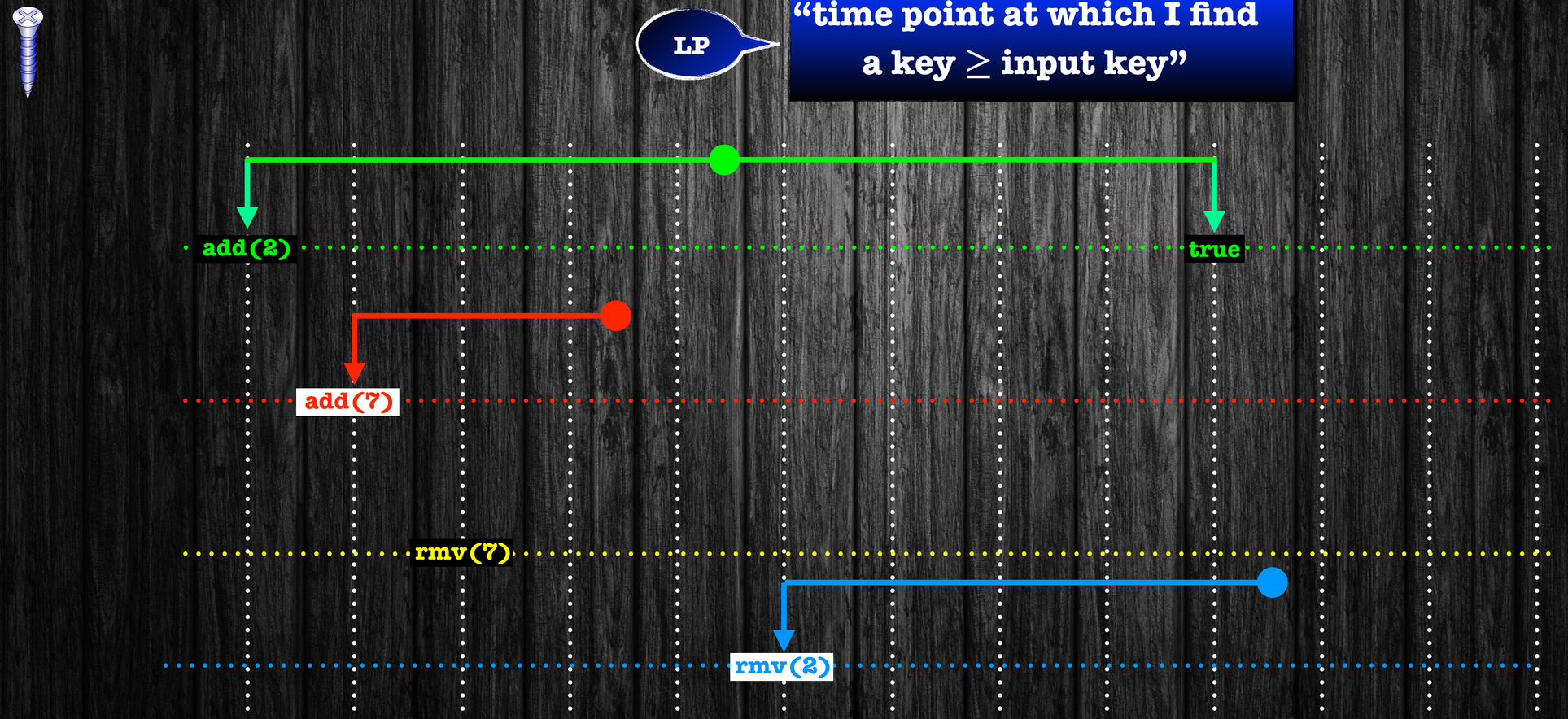
add(7)

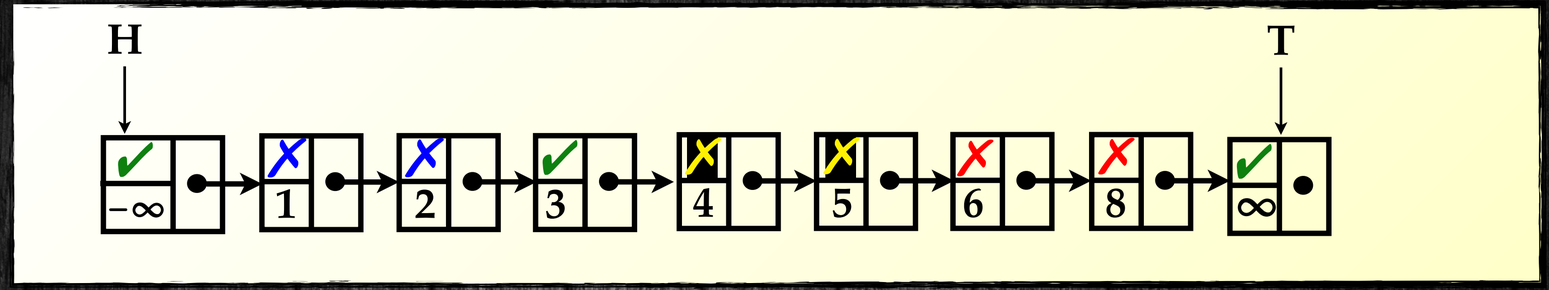
rmv(7)

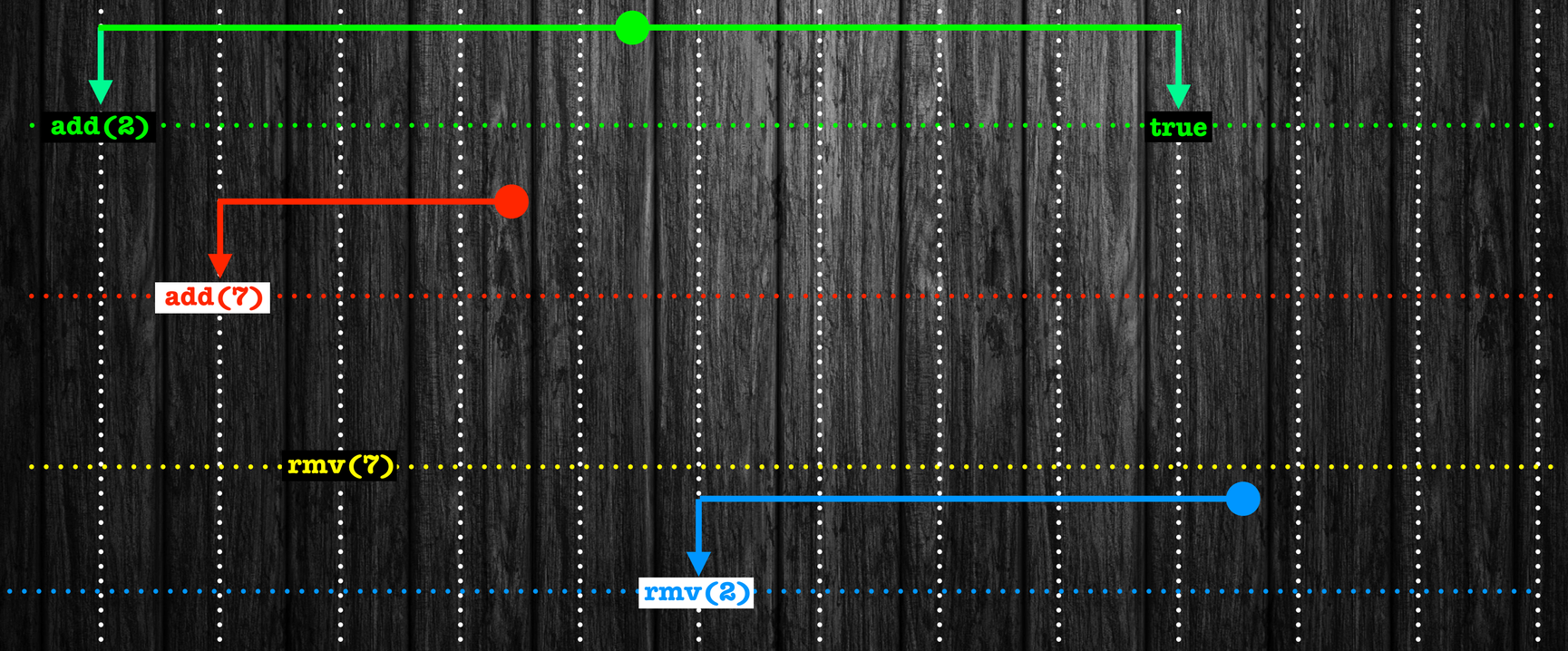
rmv(2)

# Sets | Fine-Grained Synchronization | Linearization Policy

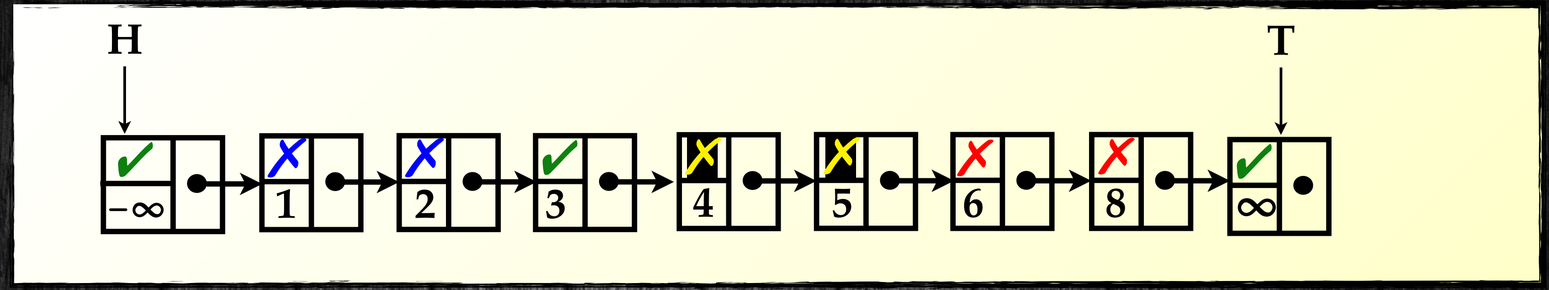


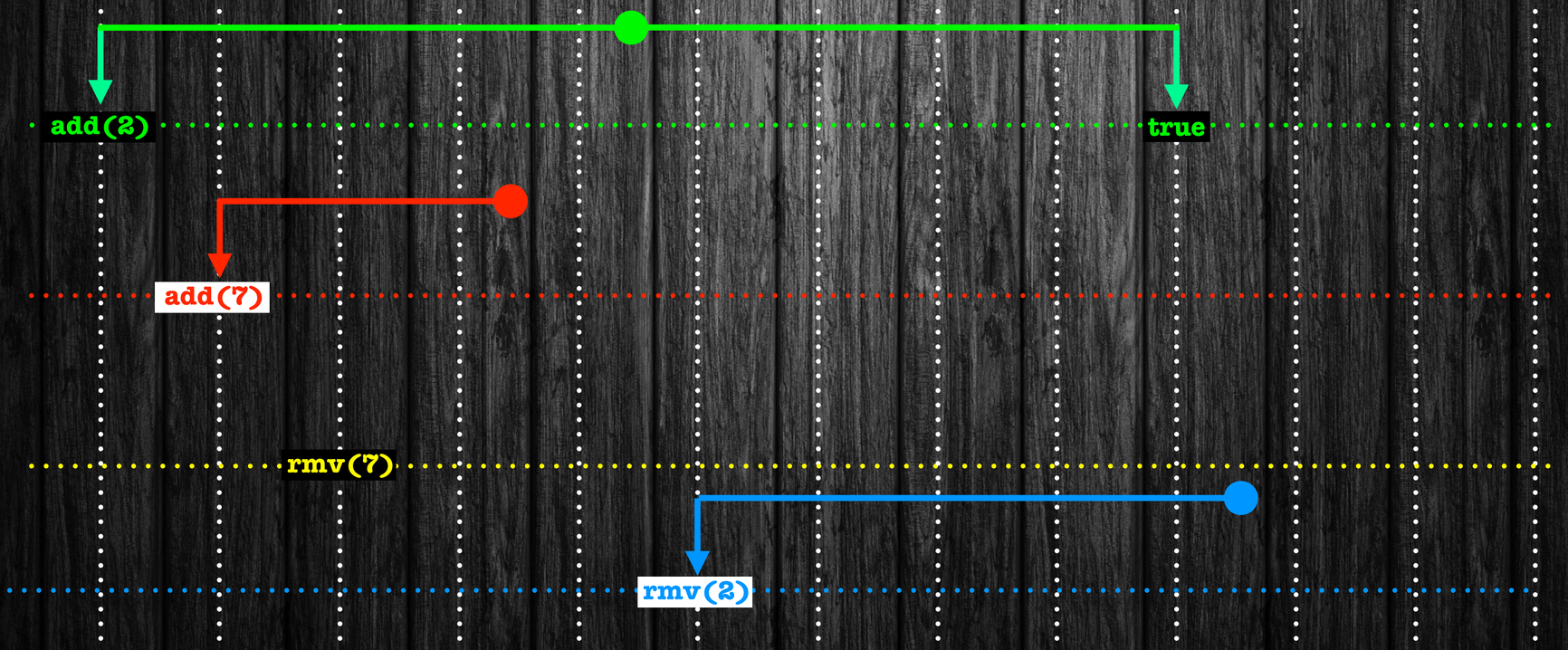

H
T
✓
−∞
✗
1
✗
2
✓
3
✗
4
✗
5
✗
6
✗
8
✓
∞


add(2)
true
add(7)
rmv(7)
rmv(2)

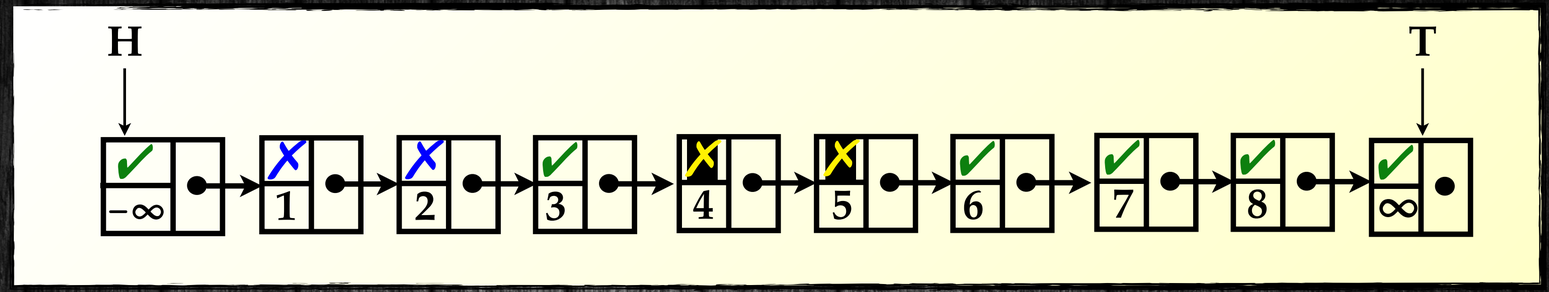
# Sets | Fine-Grained Synchronization | Linearization Policy



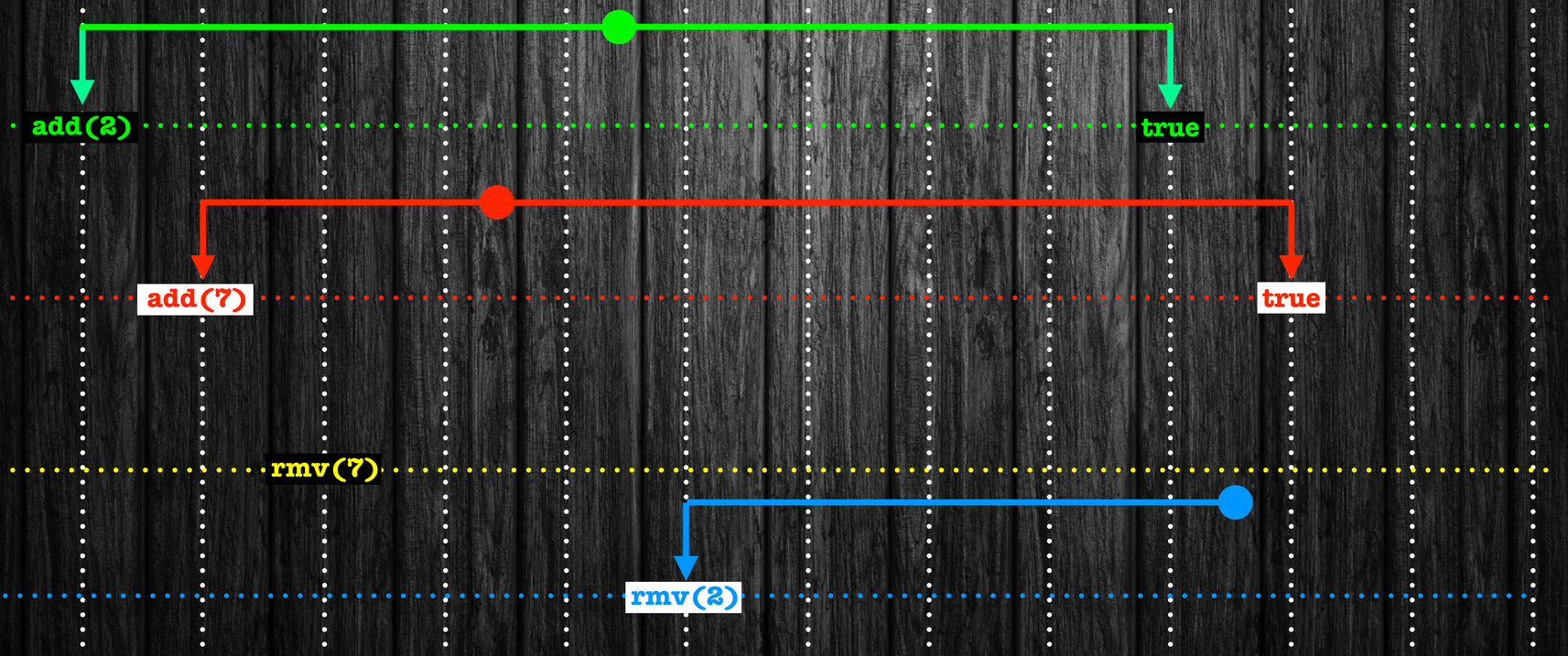


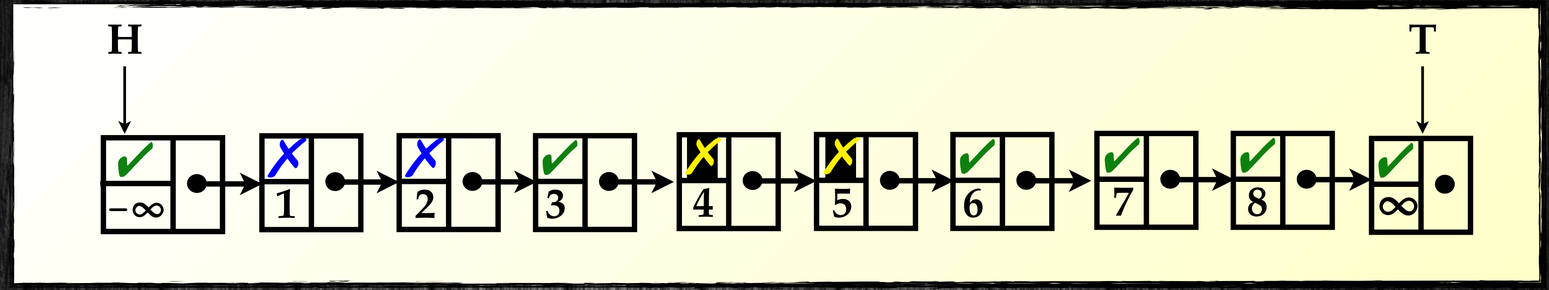
# Sets | Fine-Grained Synchronization | Linearization Policy

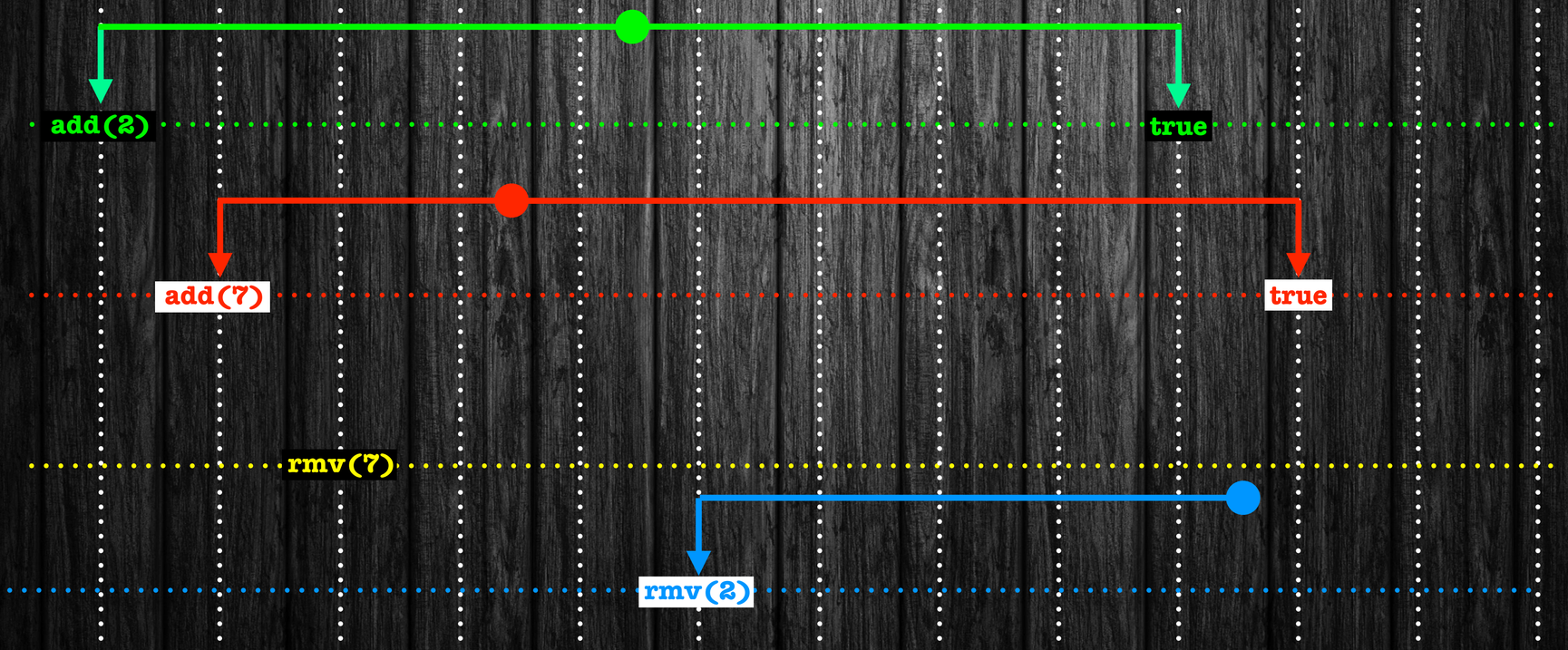




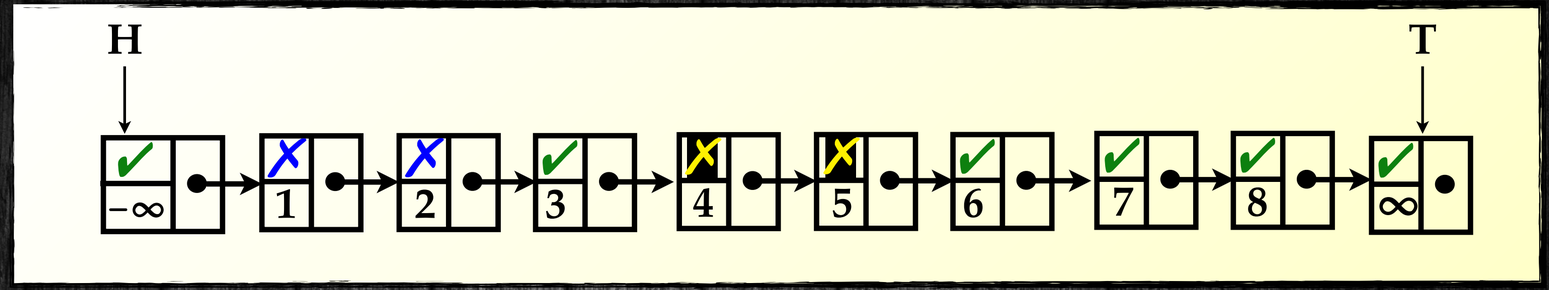



# Sets | Fine-Grained Synchronization | Linearization Policy

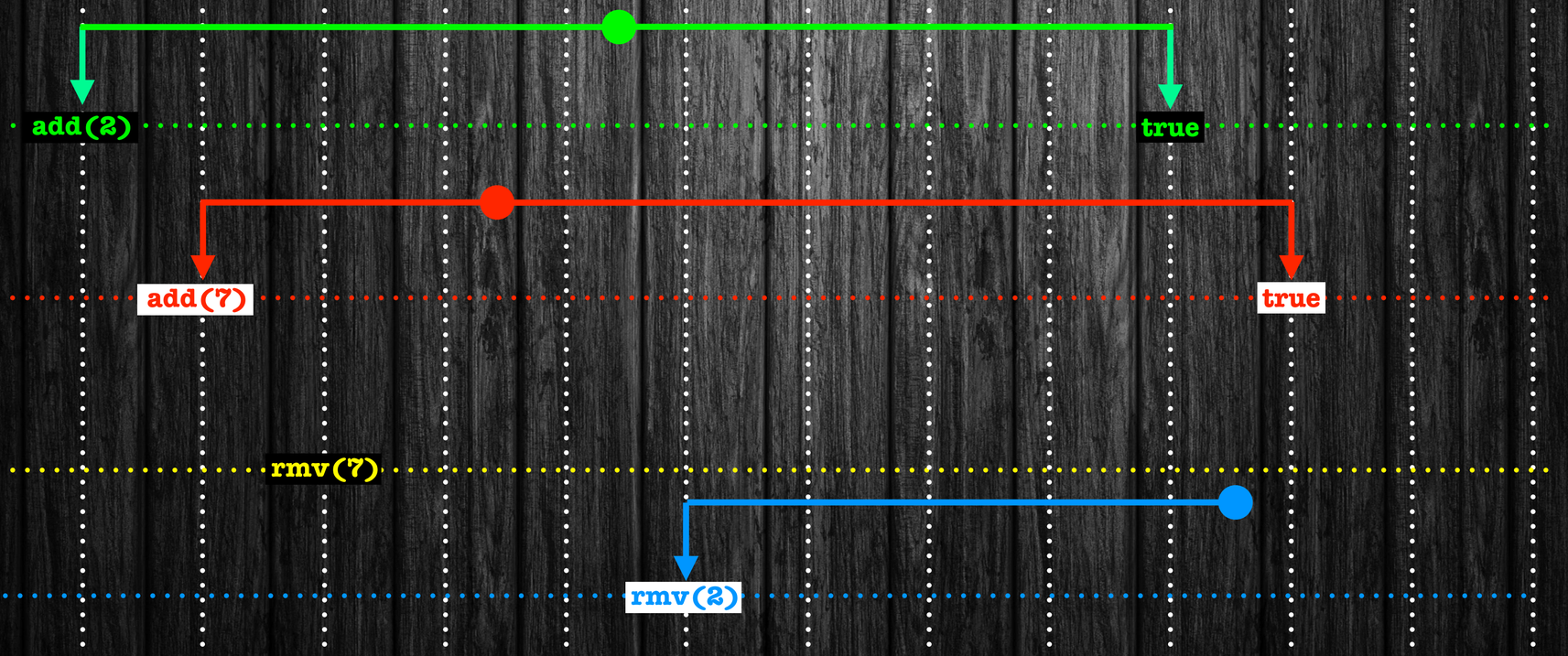


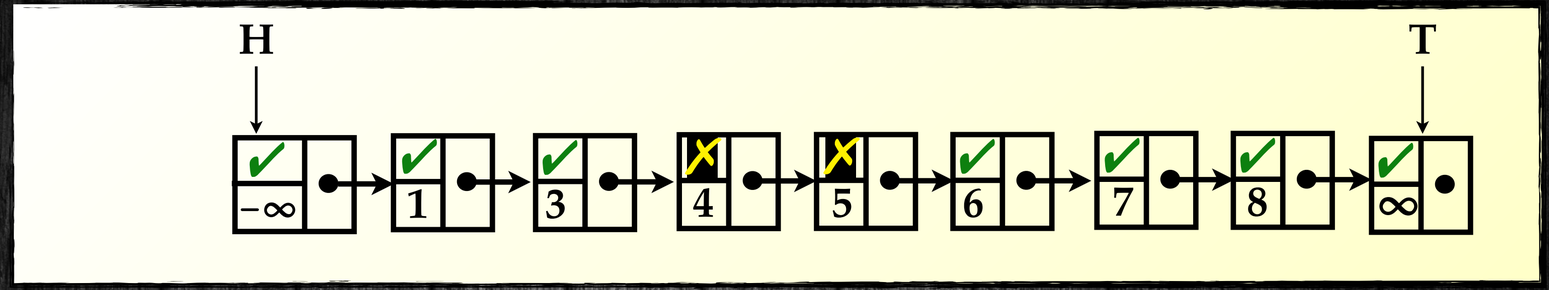
# Sets | Fine-Grained Synchronization | Linearization Policy



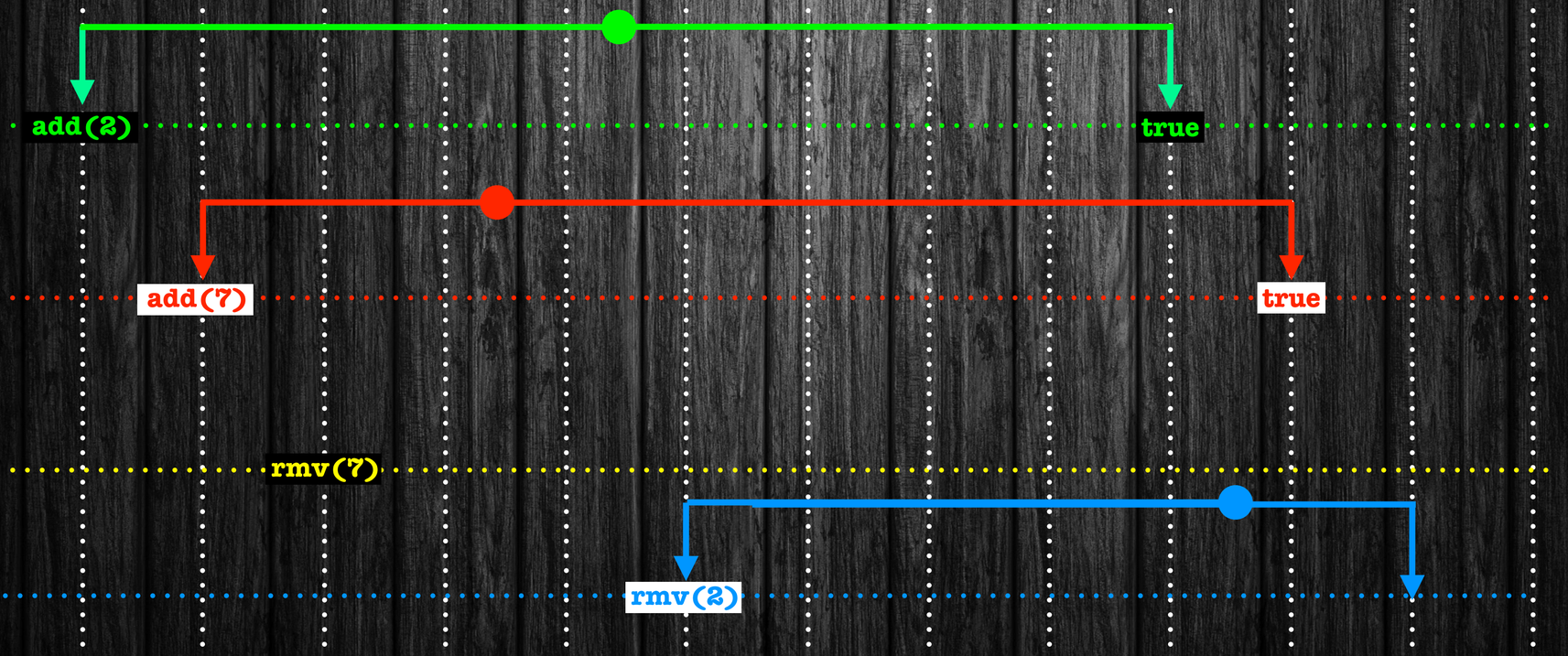


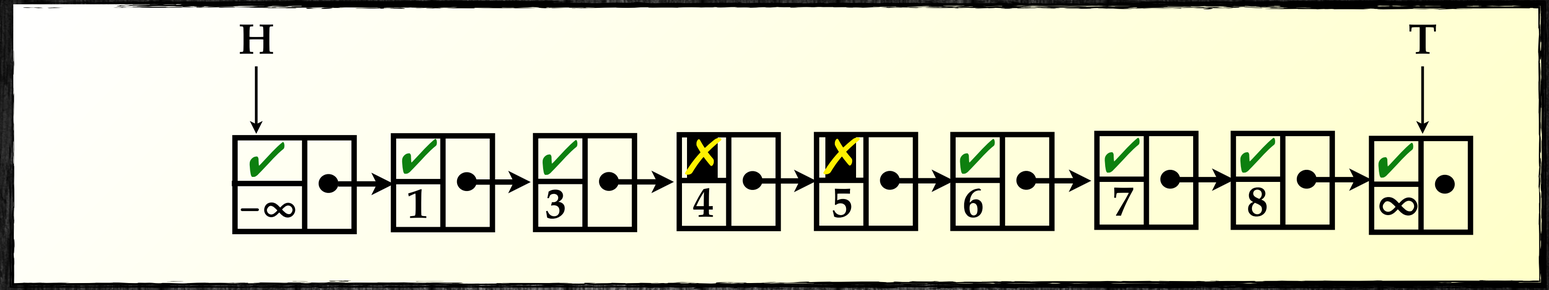
# Sets | Fine-Grained Synchronization | Linearization Policy

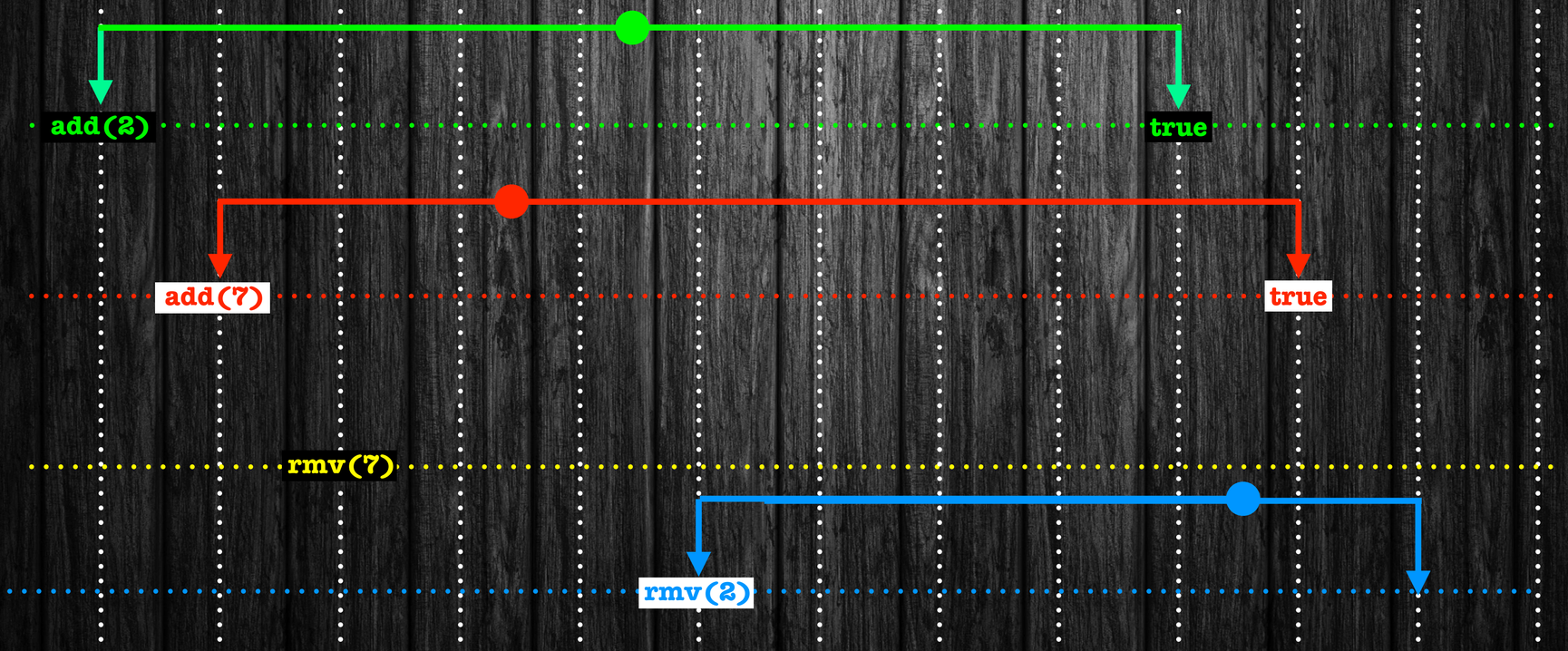




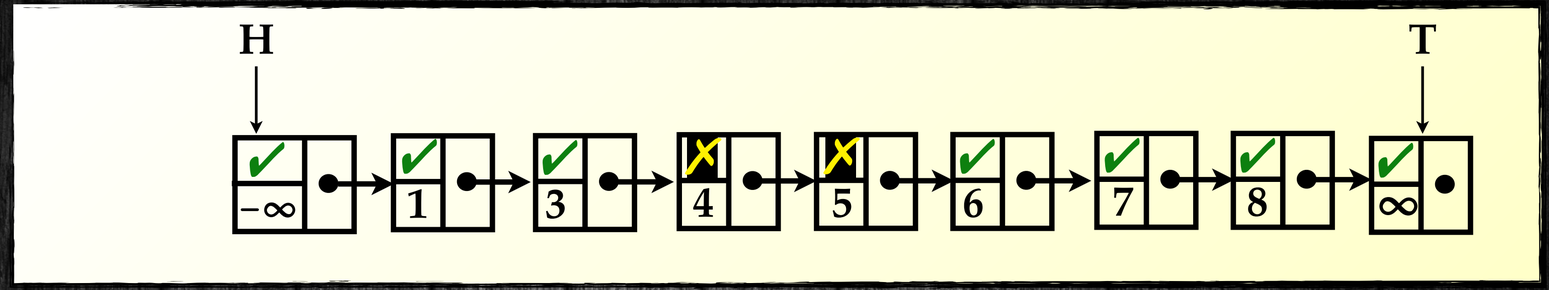



# Sets | Fine-Grained Synchronization | Linearization Policy

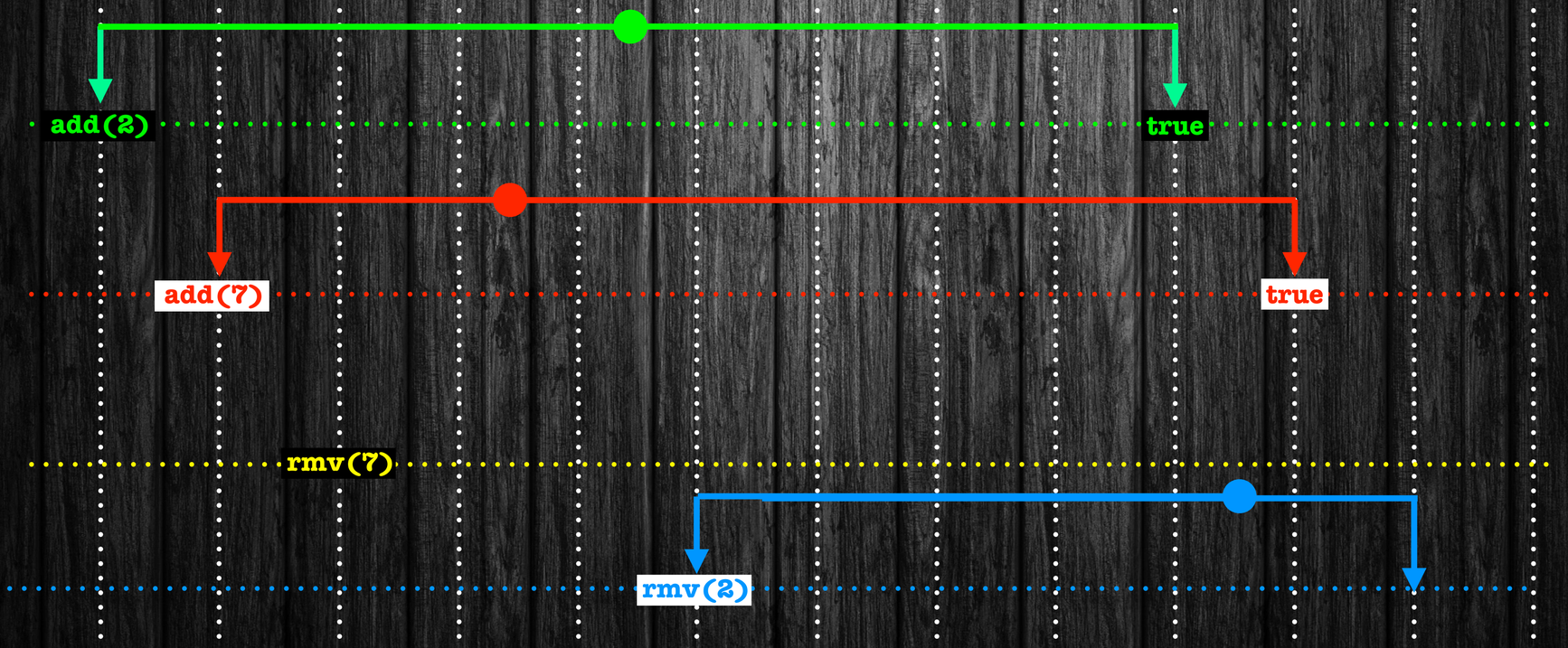


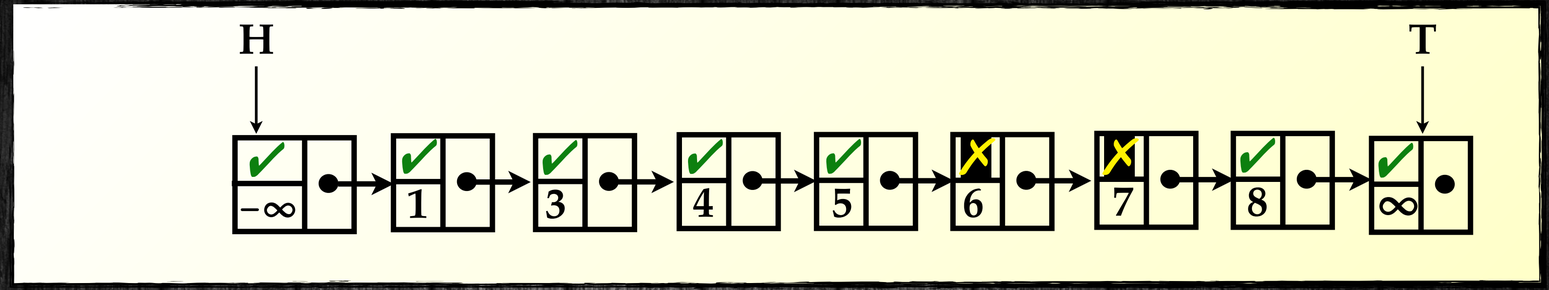
# Sets | Fine-Grained Synchronization | Linearization Policy



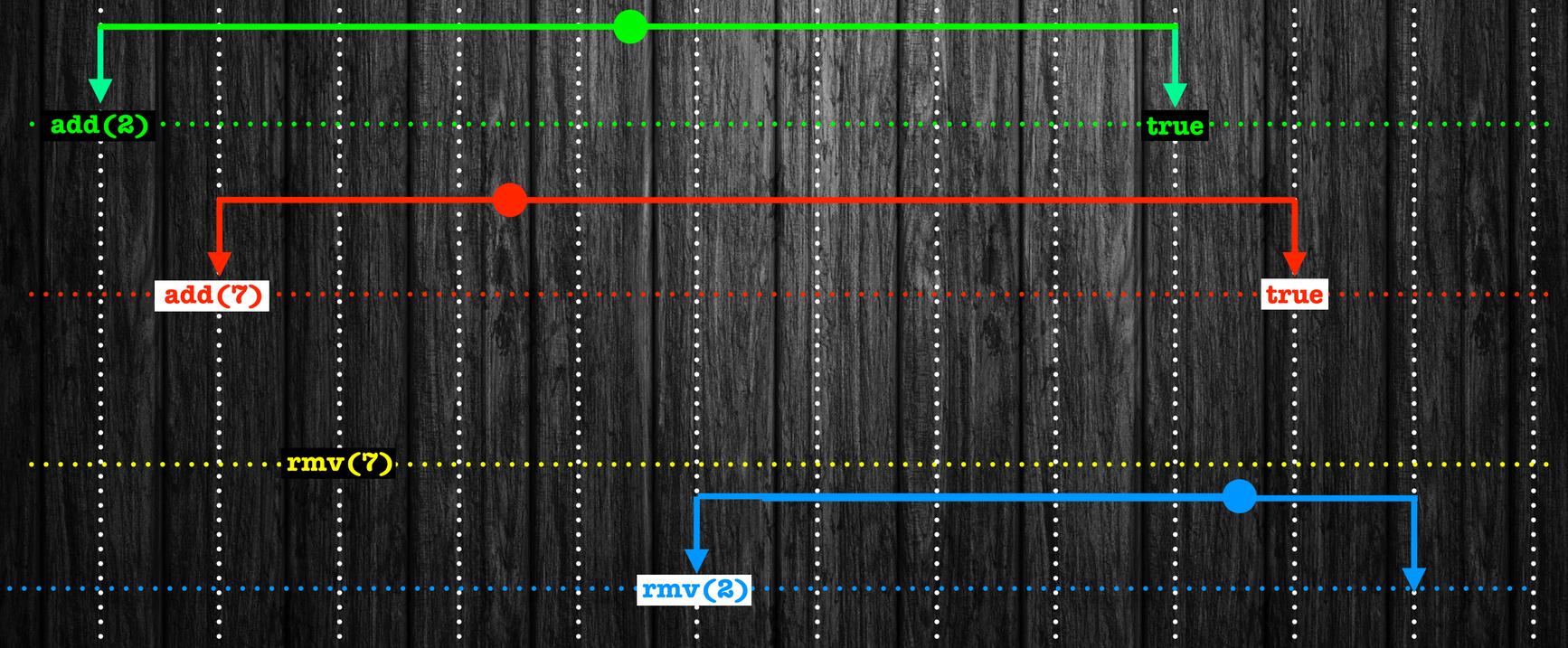


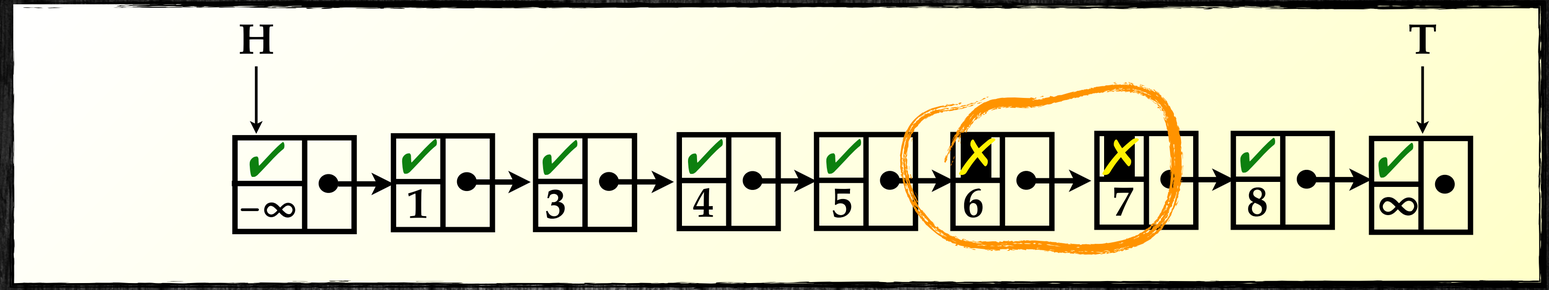
# Sets | Fine-Grained Synchronization | Linearization Policy

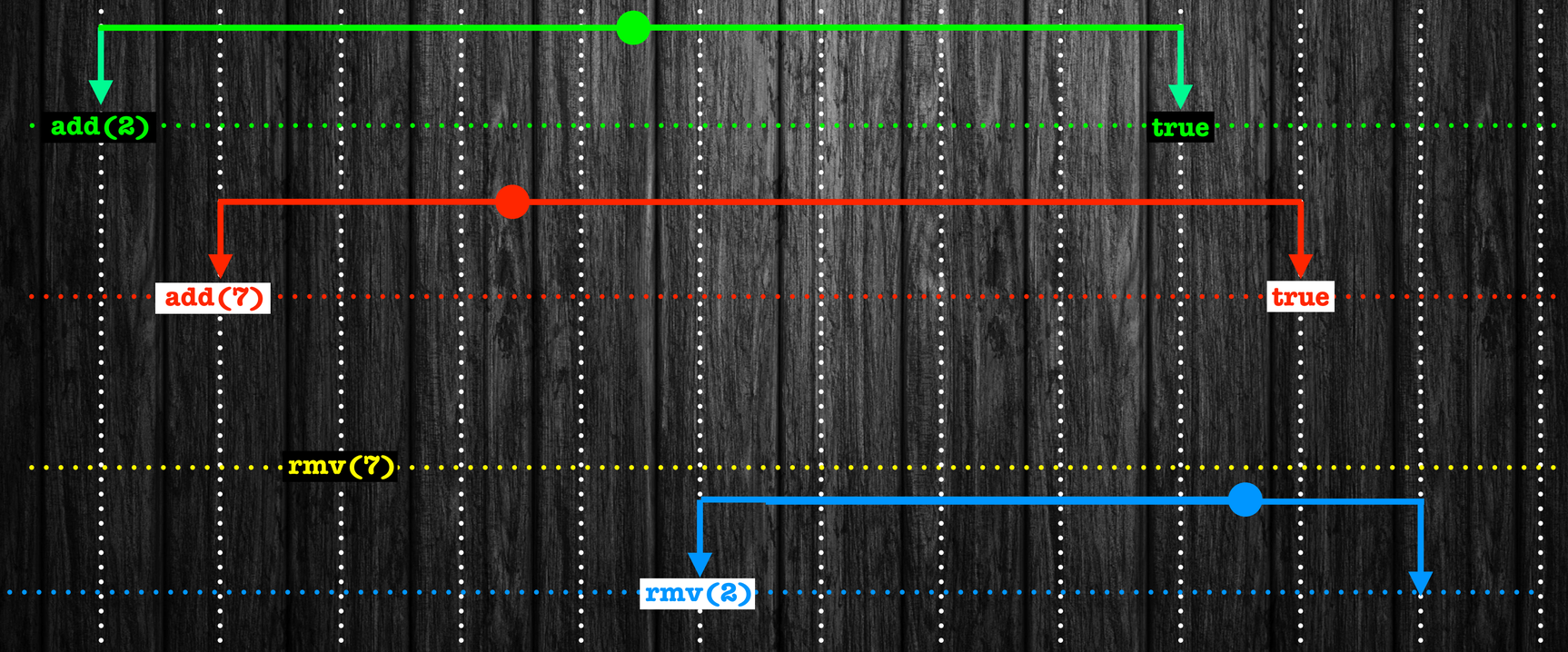




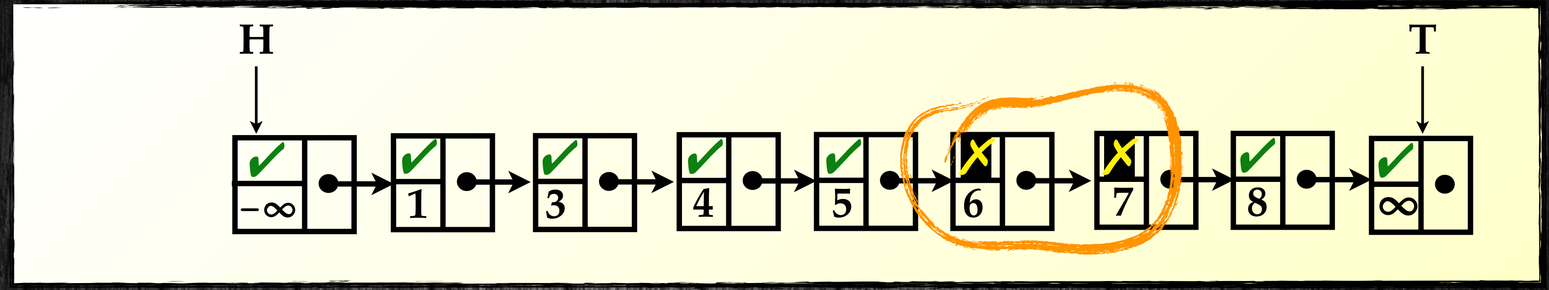



# Sets | Fine-Grained Synchronization | Linearization Policy



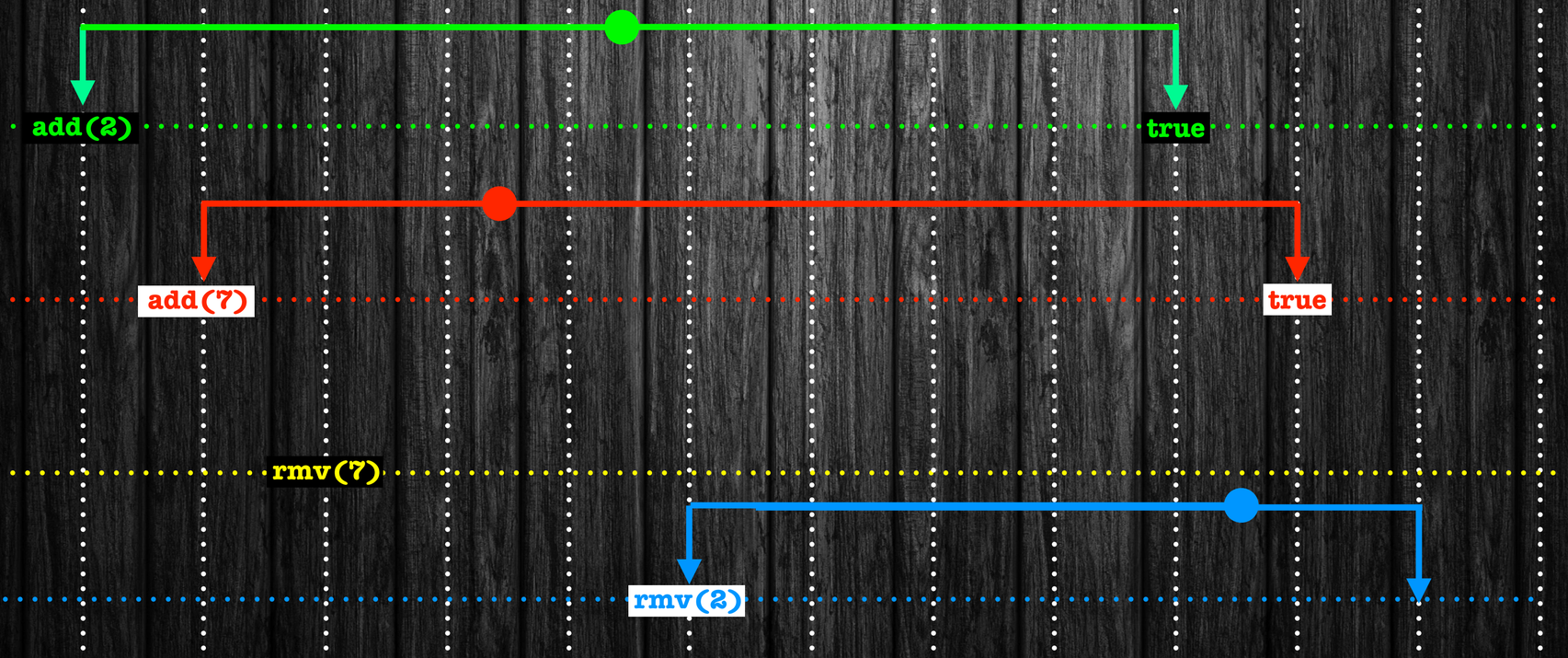



# Sets | Fine-Grained Synchronization | Linearization Policy

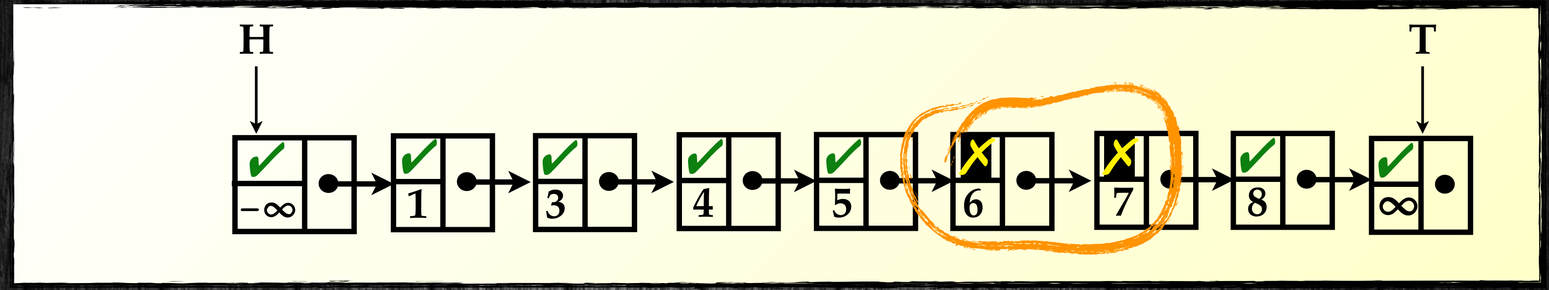



LP

"time point at which I find a key ≥ input key"

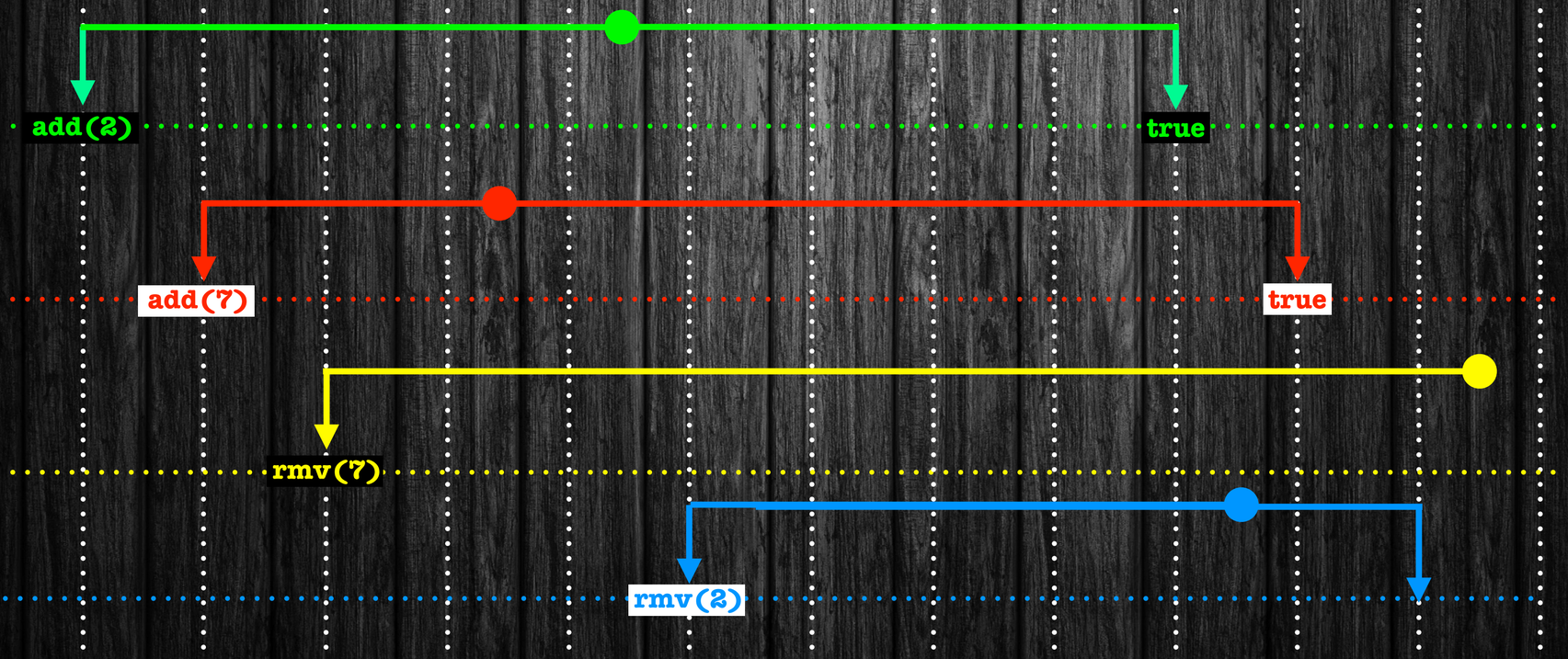


# Sets | Fine-Grained Synchronization | Linearization Policy

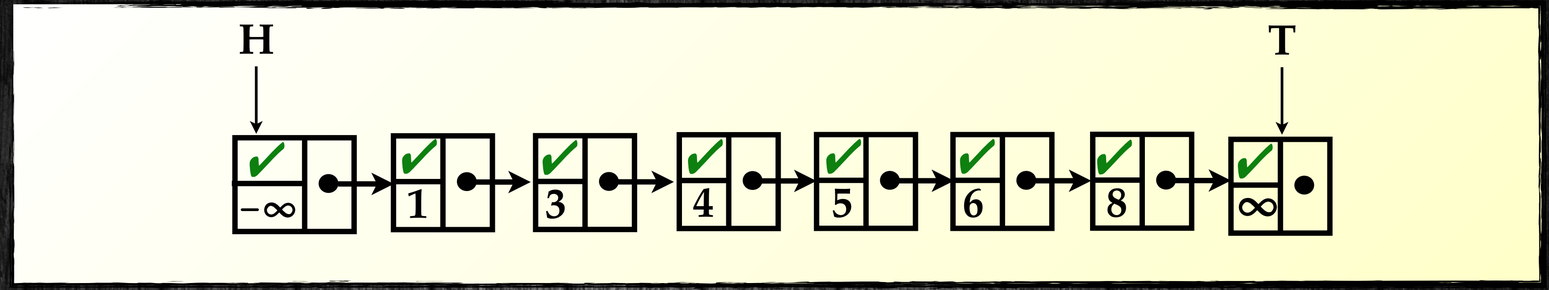



LP

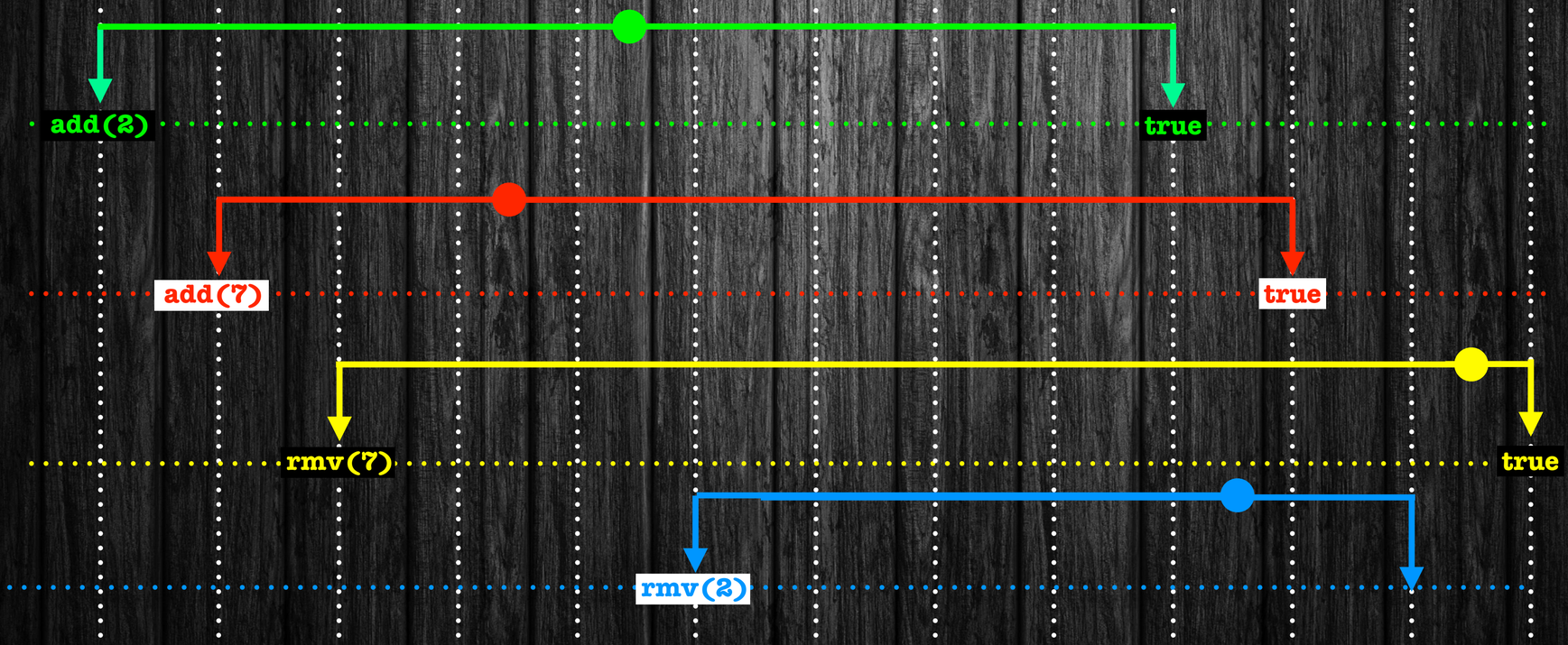
"time point at which I find a key ≥ input key"



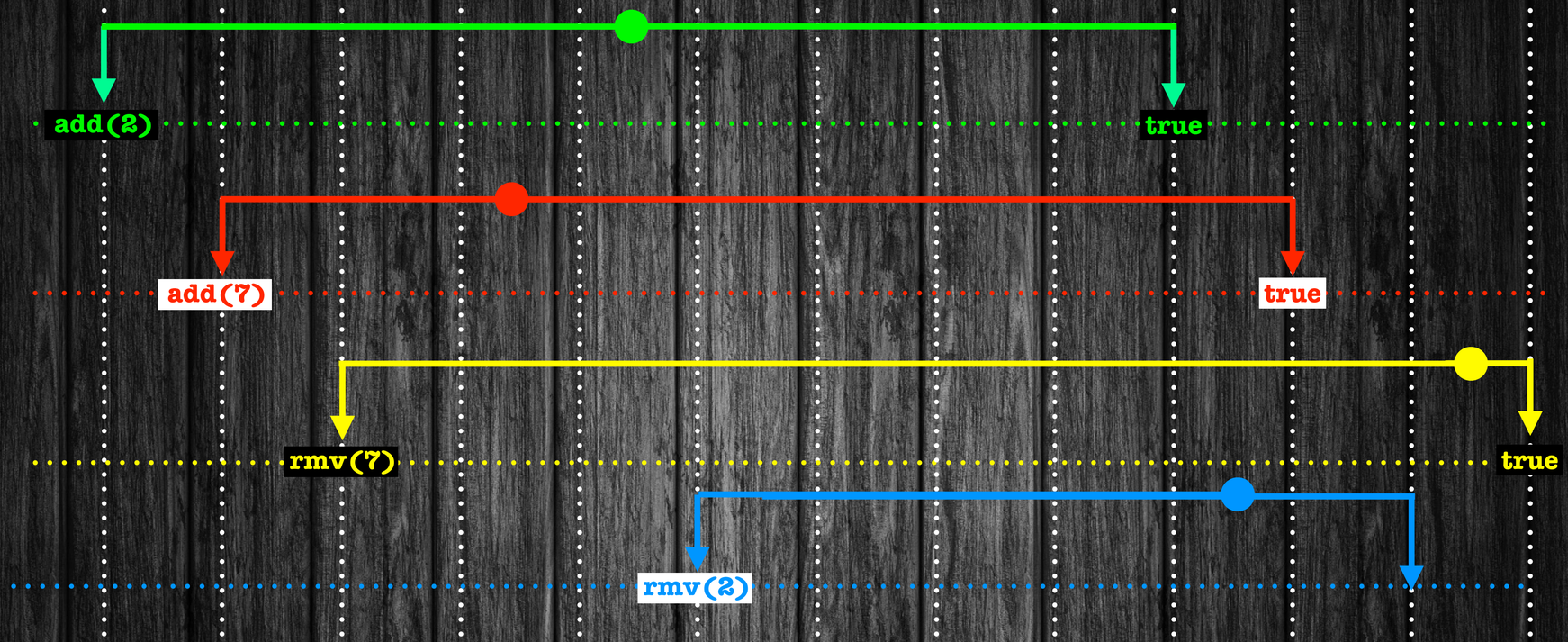
# Sets | Fine-Grained Synchronization | Linearization Policy

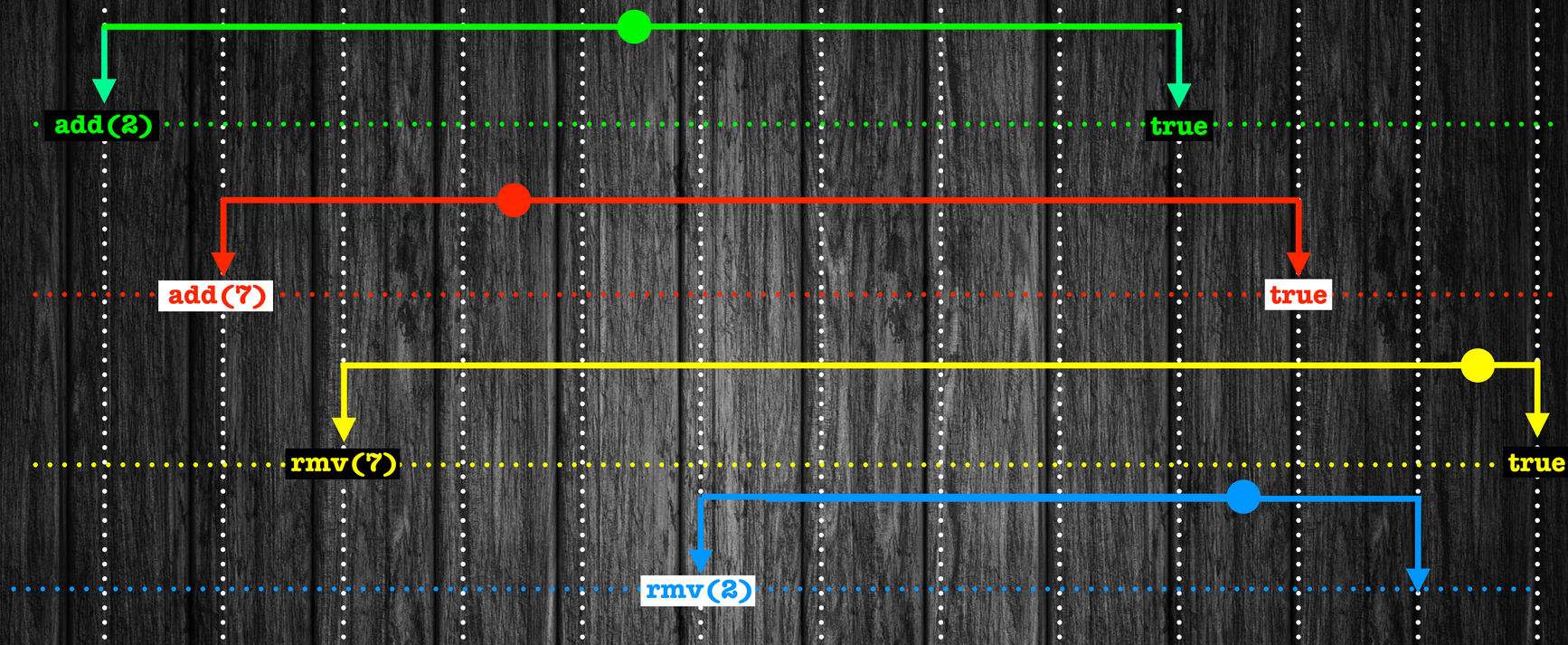




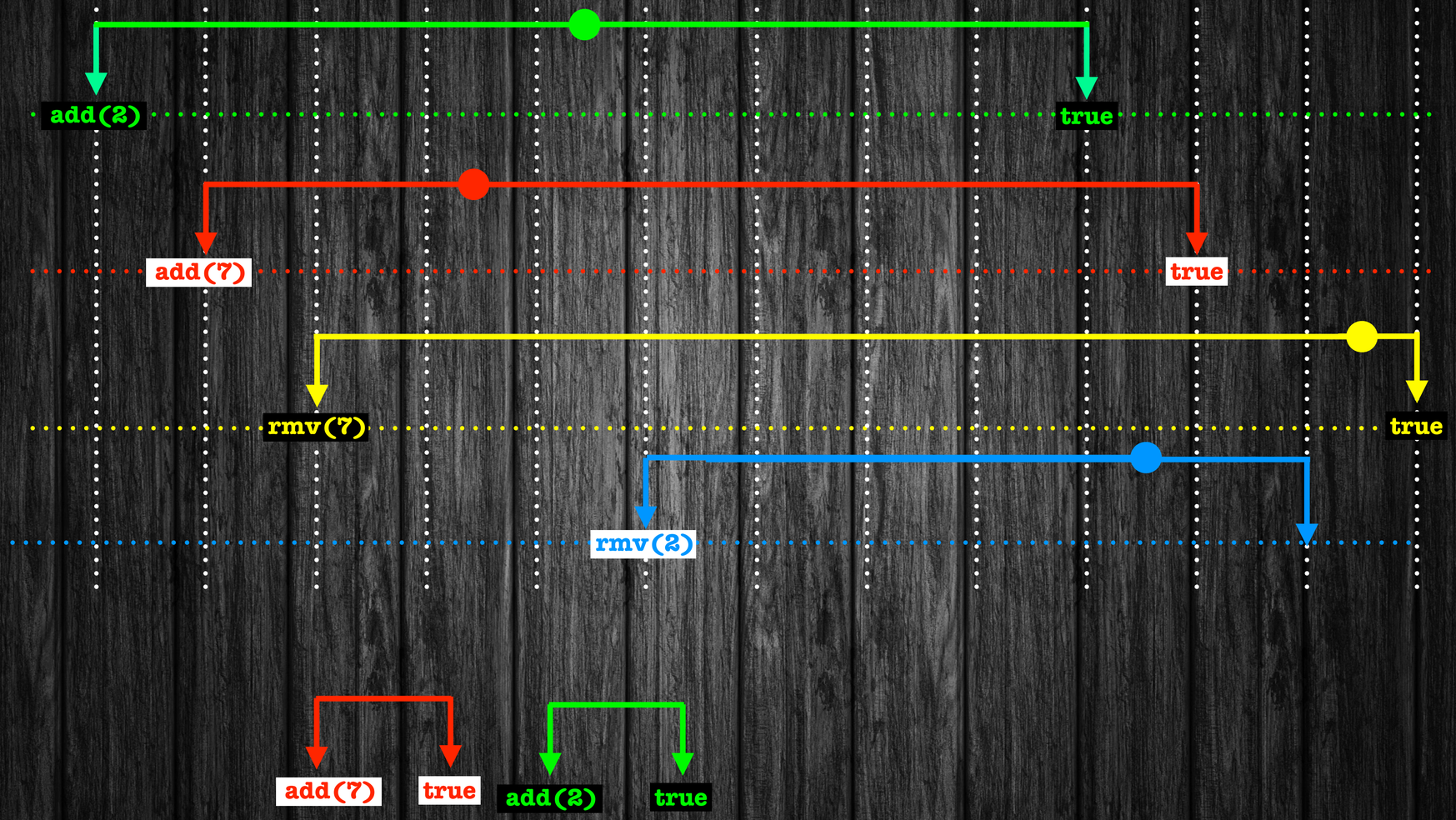
# Sets | Fine-Grained Synchronization | Linearization Policy



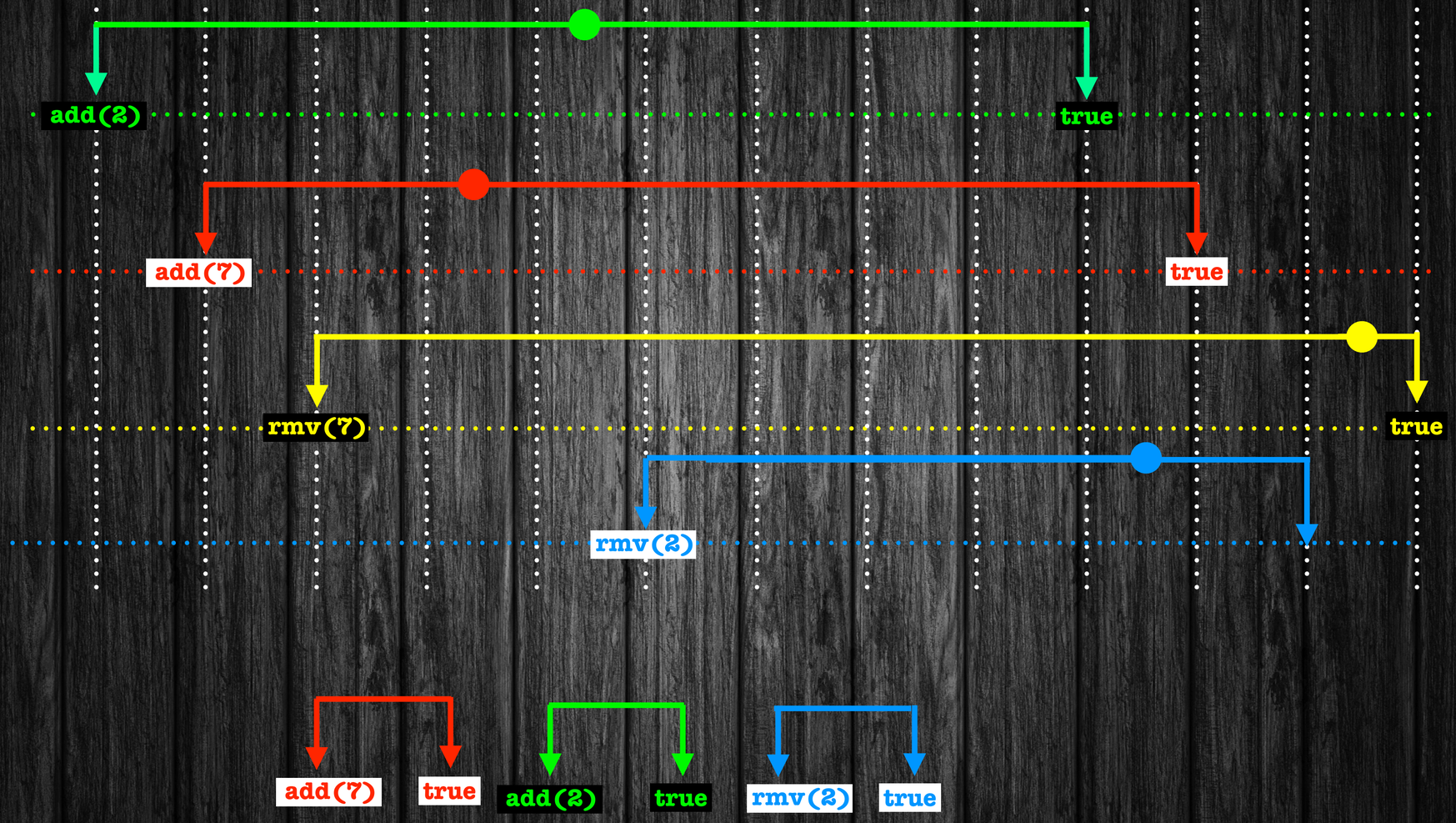

add(2)
true
add(7)
true
rmv(7)
true
rmv(2)


add(7)
true

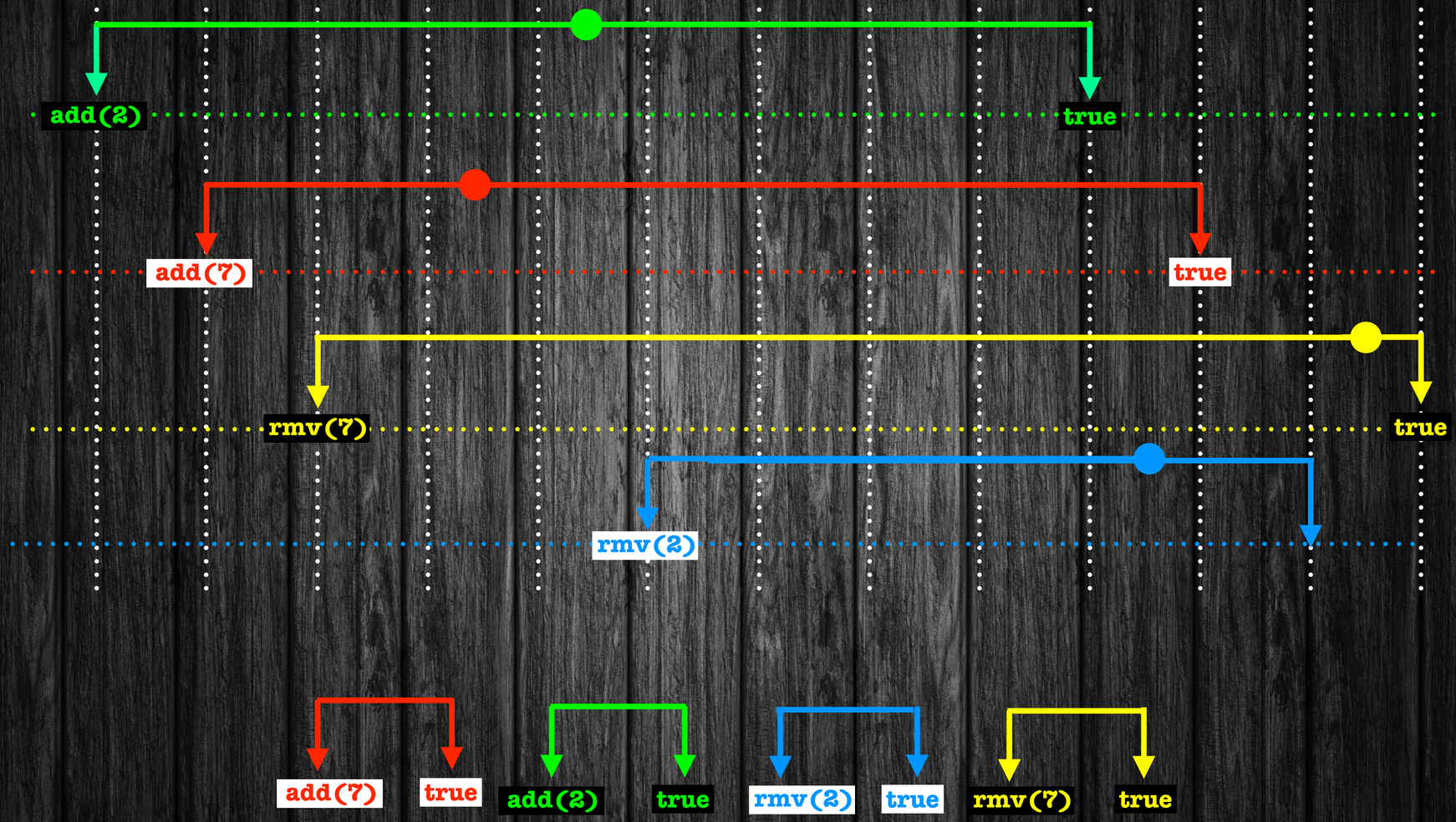
# Sets | Fine-Grained Synchronization | Linearization Policy



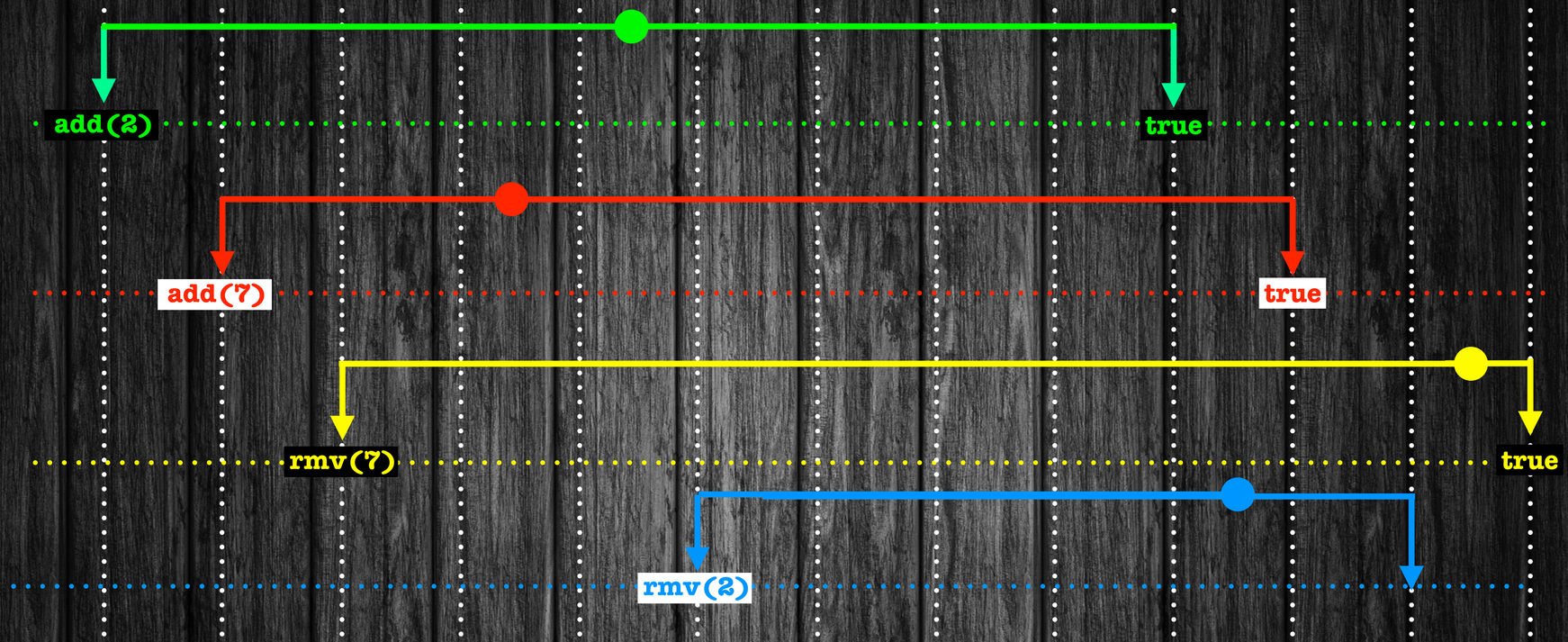
# Sets | Fine-Grained Synchronization | Linearization Policy

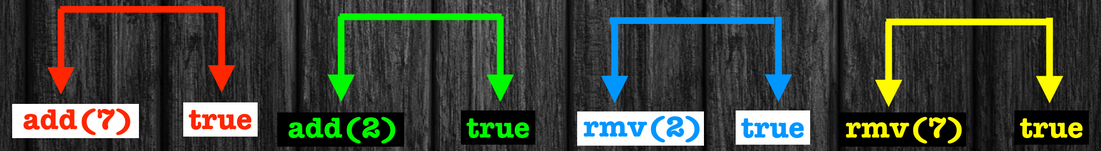


# Sets | Fine-Grained Synchronization | Linearization Policy



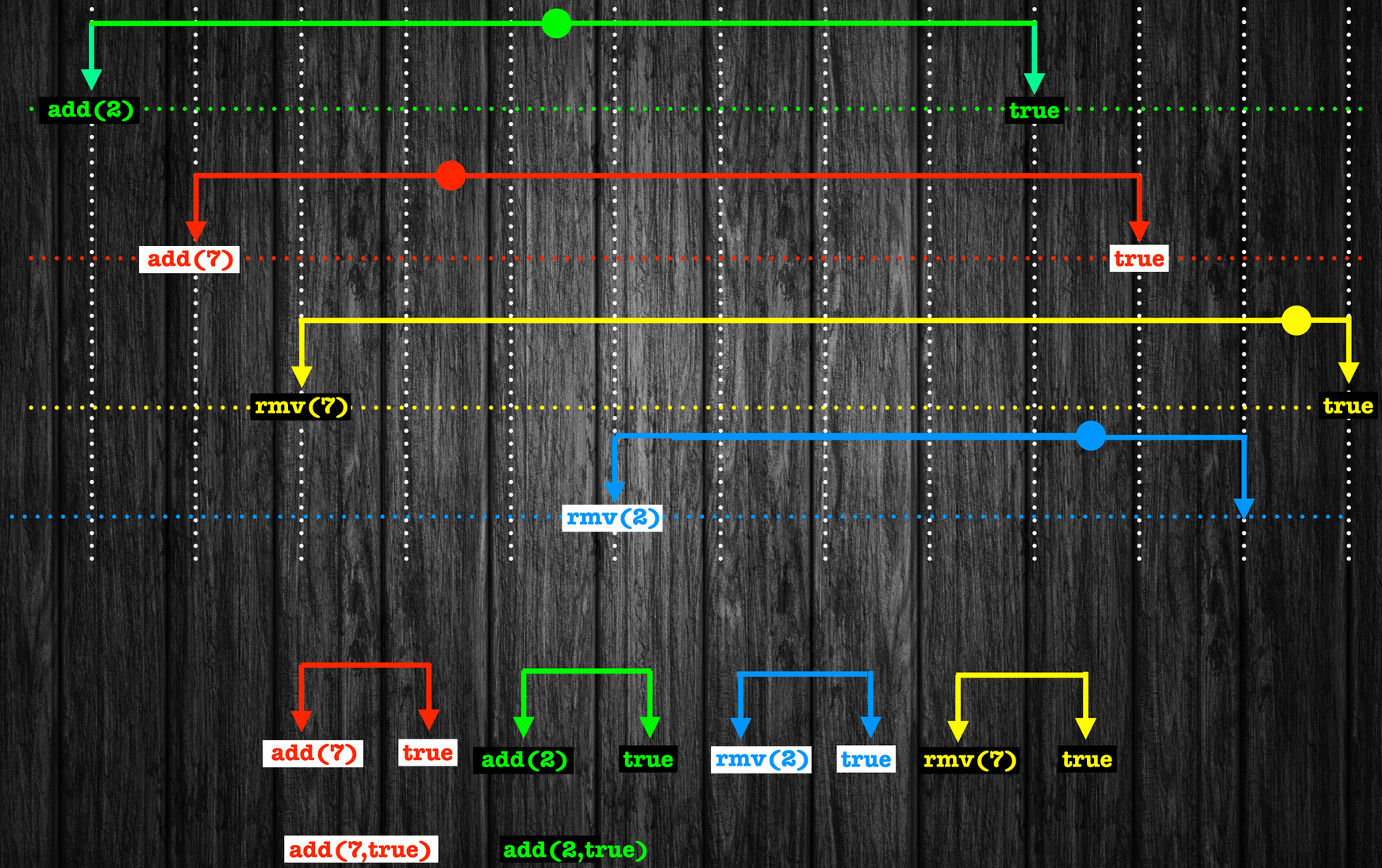
# Sets | Fine-Grained Synchronization | Linearization Policy



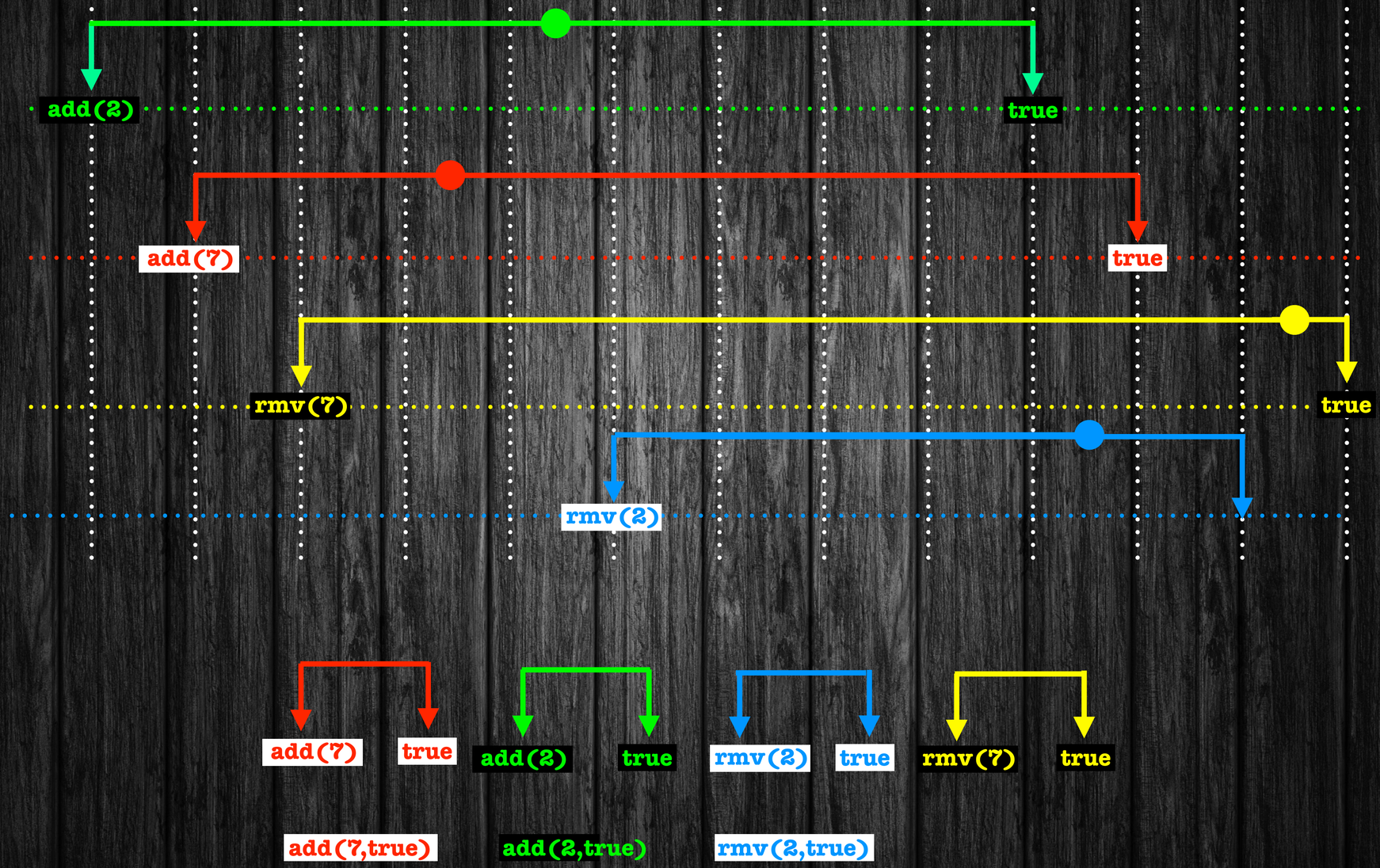




add(7,true)

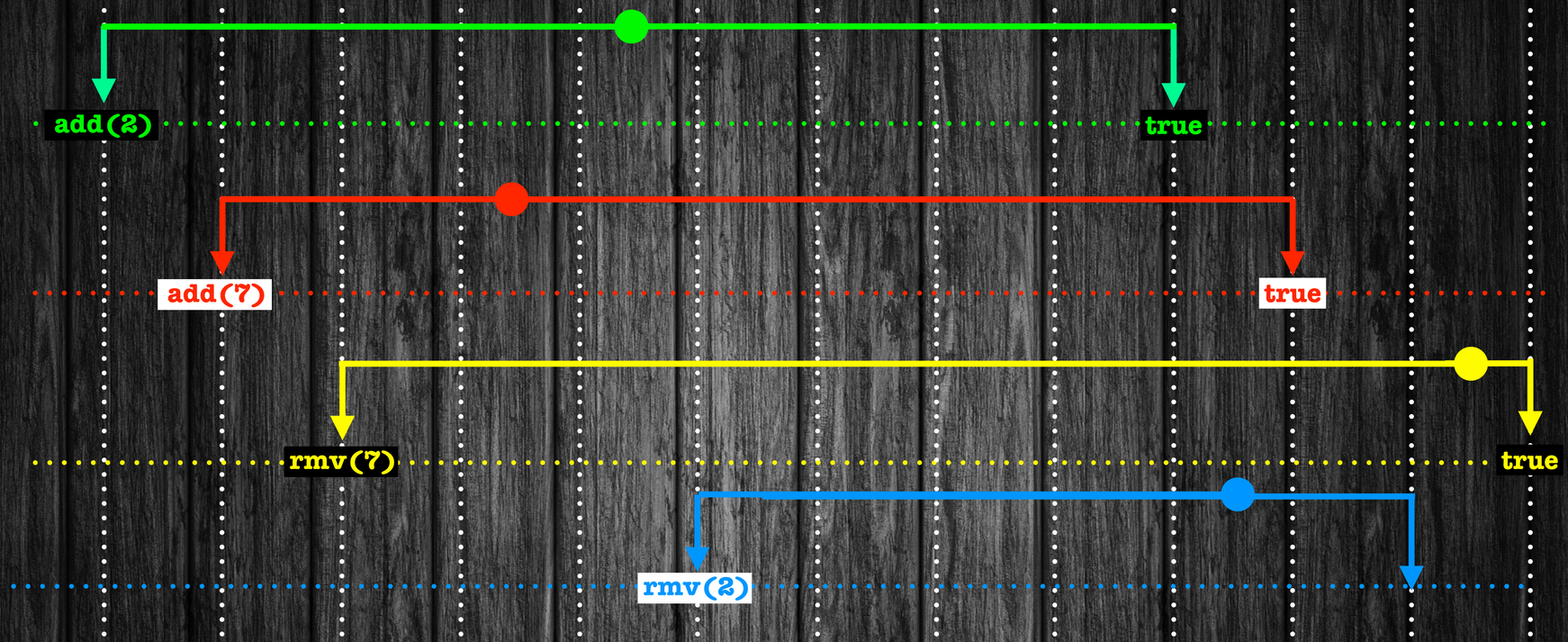
# Sets | Fine-Grained Synchronization | Linearization Policy

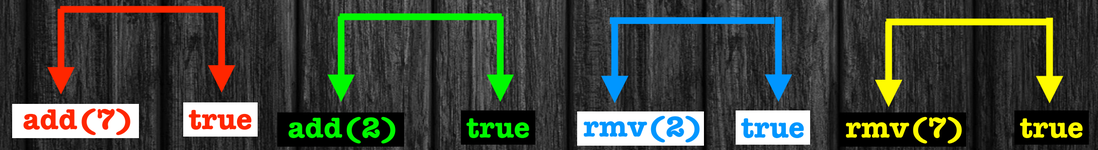


# Sets | Fine-Grained Synchronization | Linearization Policy



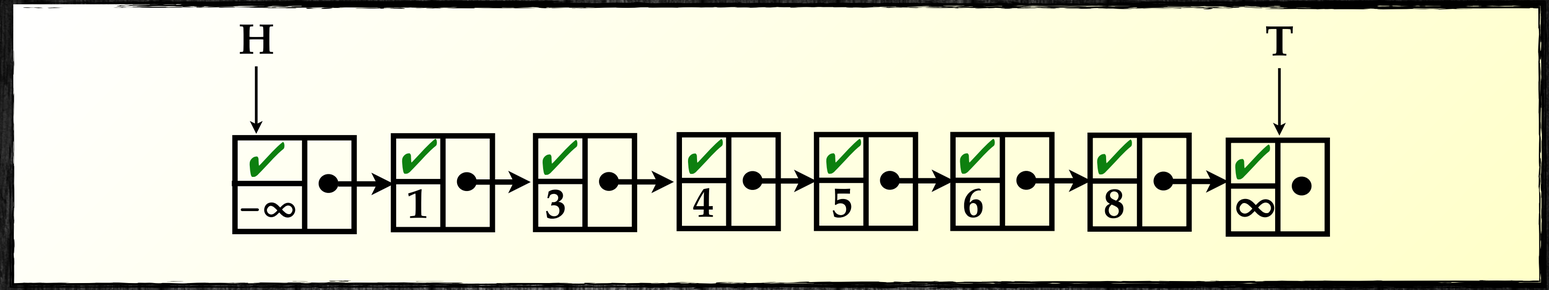
# Sets | Fine-Grained Synchronization | Linearization Policy



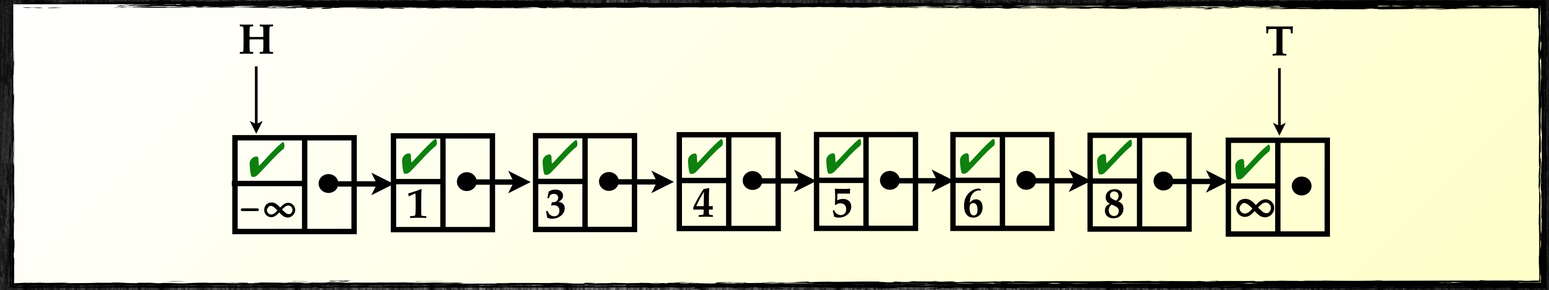




add(7,true) add(2,true) rmv(2,true) rmv(7,true)

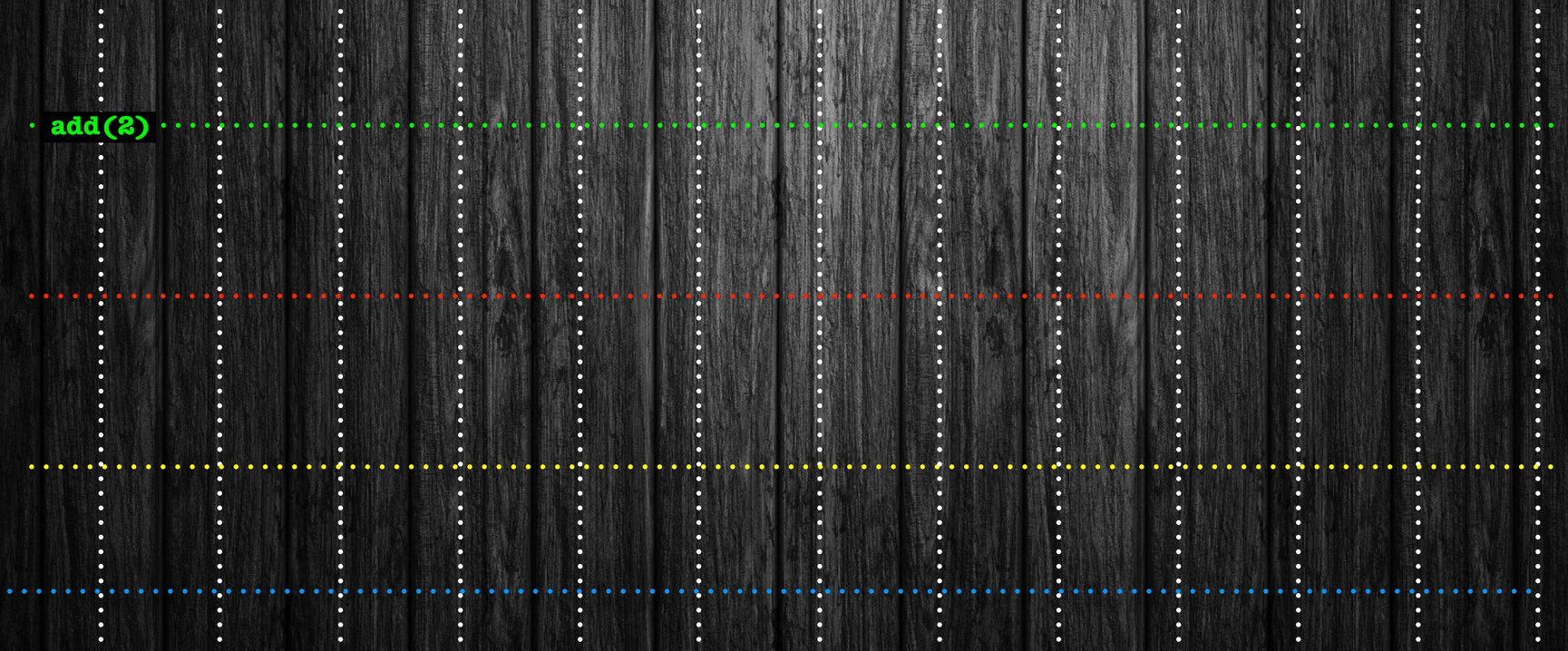
# Sets | Fine-Grained Synchronization | Linearization Policy



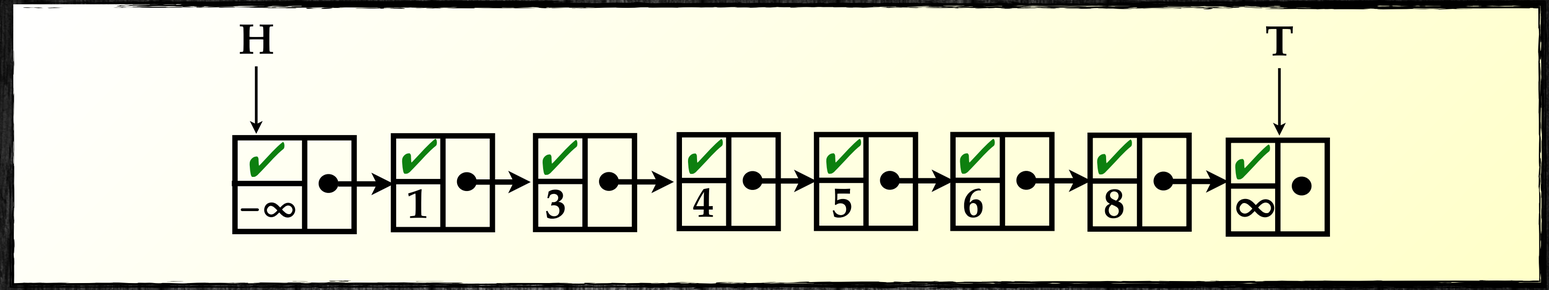
# Sets | Fine-Grained Synchronization | Linearization Policy

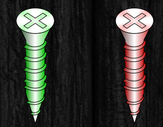


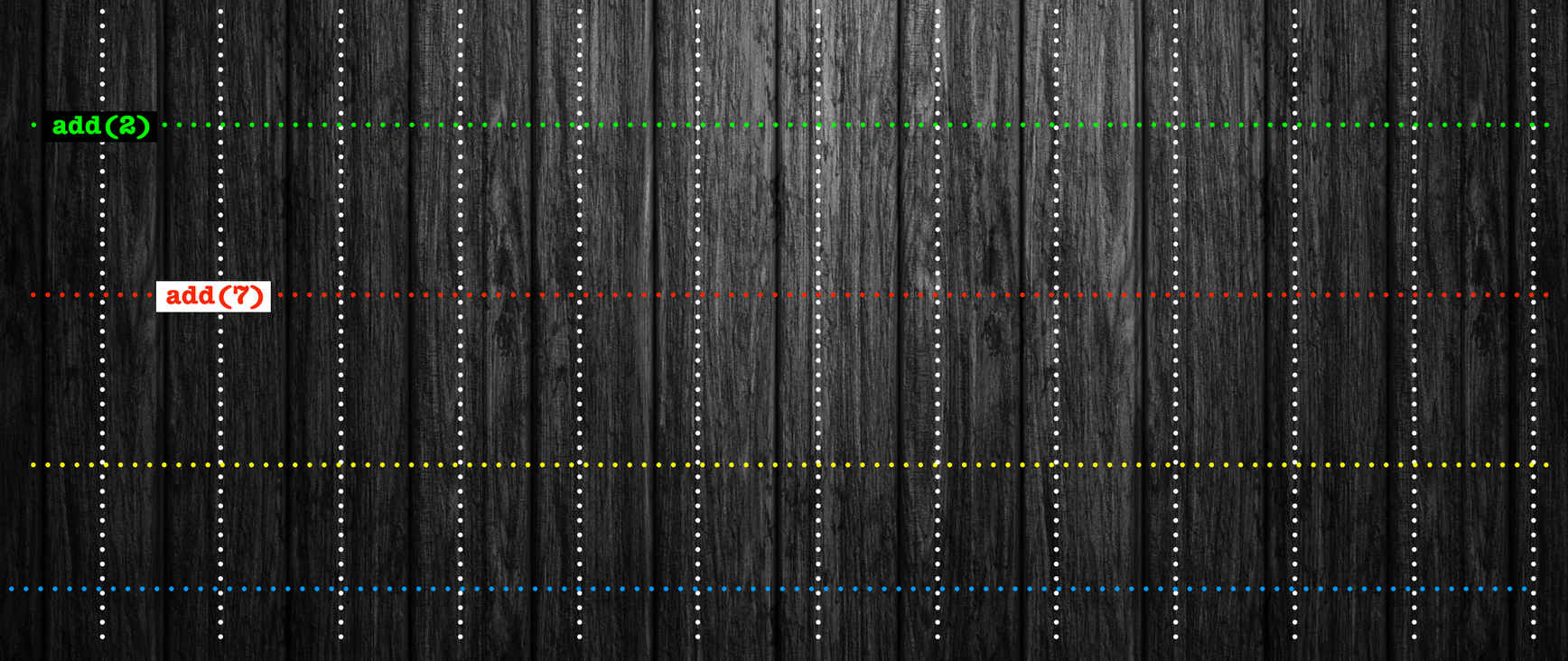


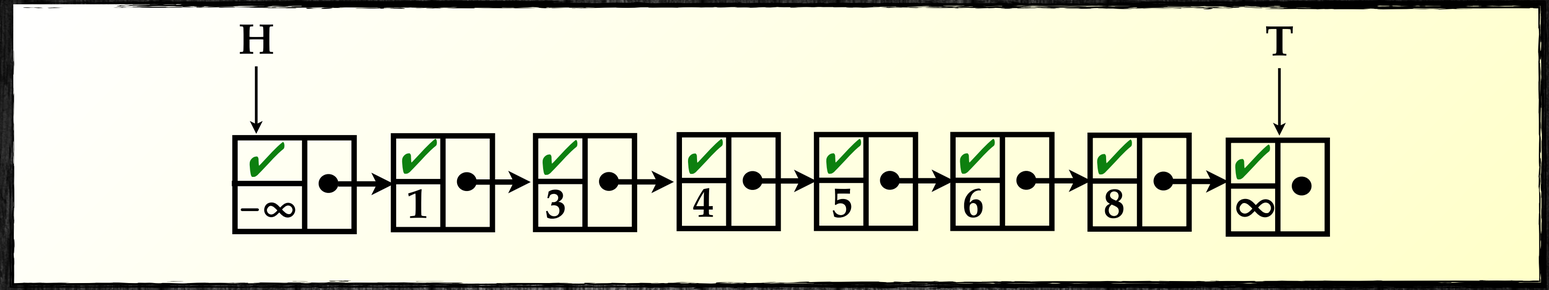
# Sets | Fine-Grained Synchronization | Linearization Policy





# Sets | Fine-Grained Synchronization | Linearization Policy



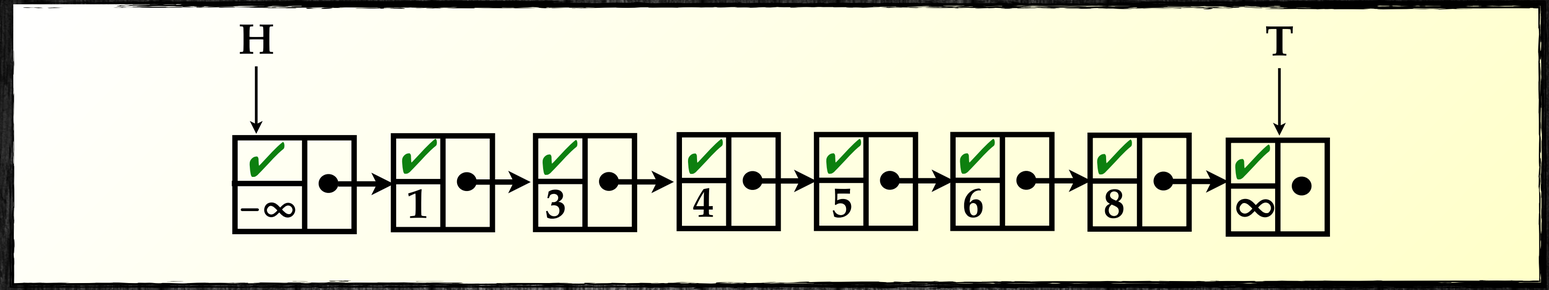



add(2)

add(7)

rmv(7)

# Sets | Fine-Grained Synchronization | Linearization Policy

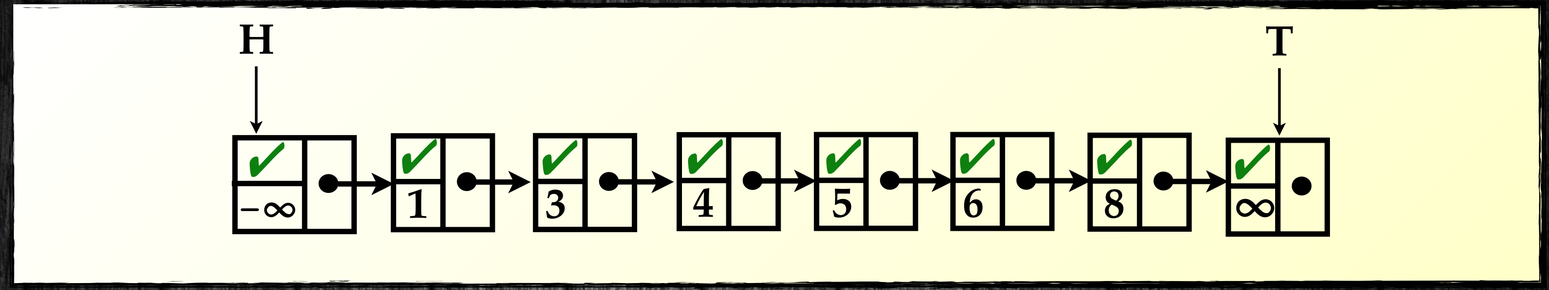


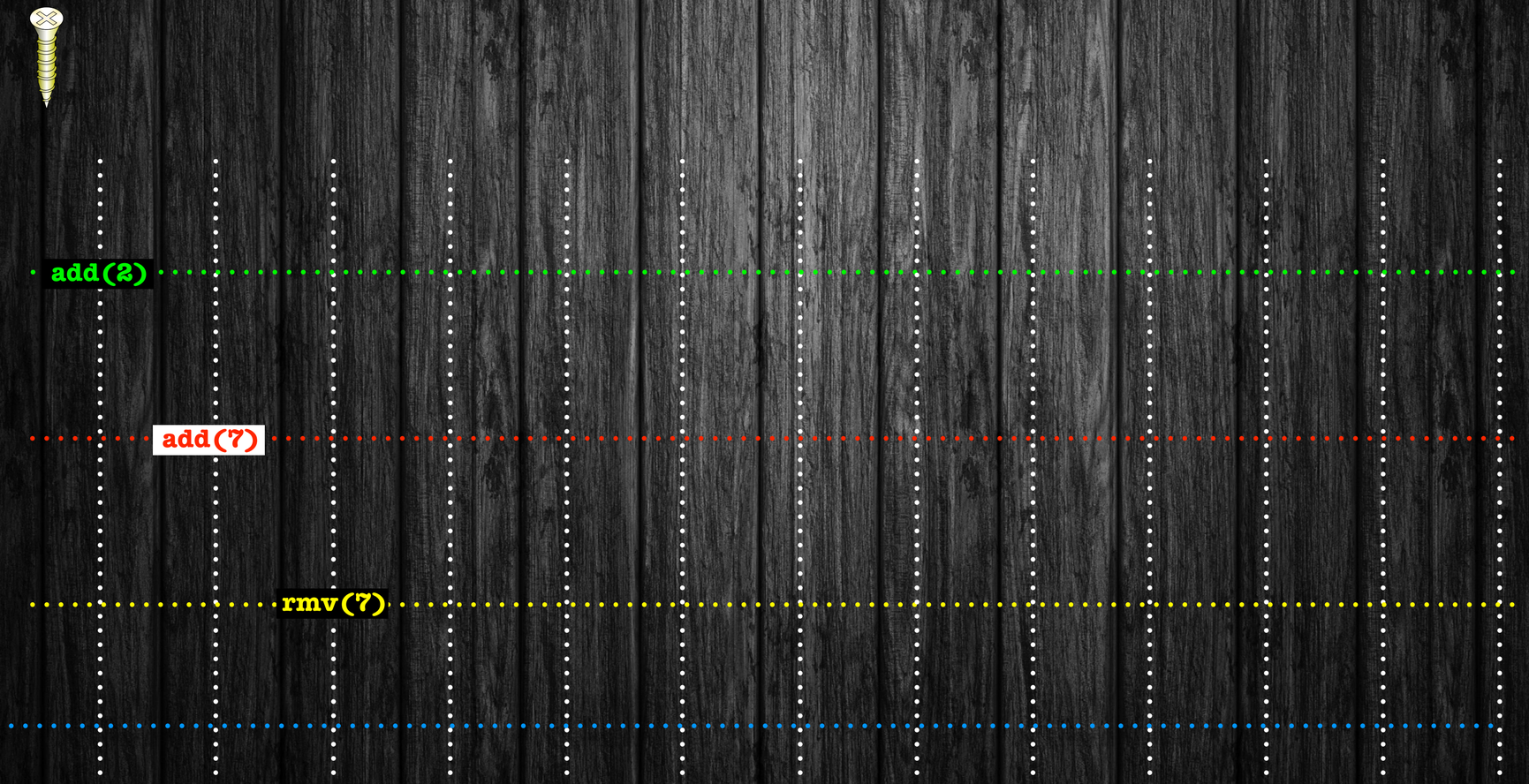

add(2)

add(7)

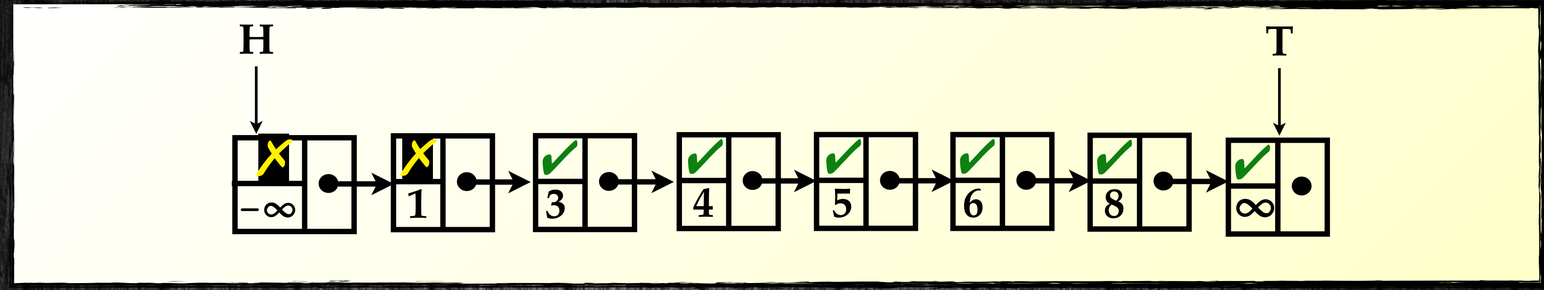
rmv(7)

# Sets | Fine-Grained Synchronization | Linearization Policy



# Sets | Fine-Grained Synchronization | Linearization Policy



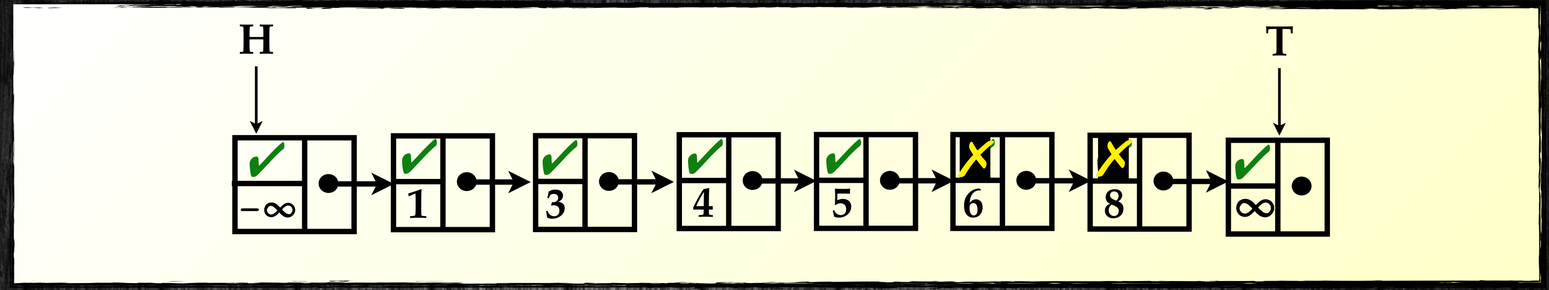



add(2)

add(7)

rmv(7)

# Sets | Fine-Grained Synchronization | Linearization Policy



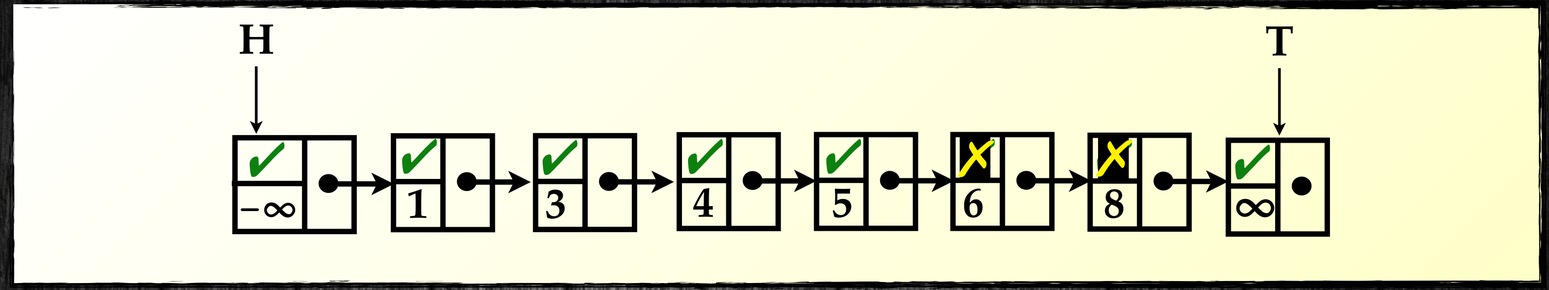



add(2)

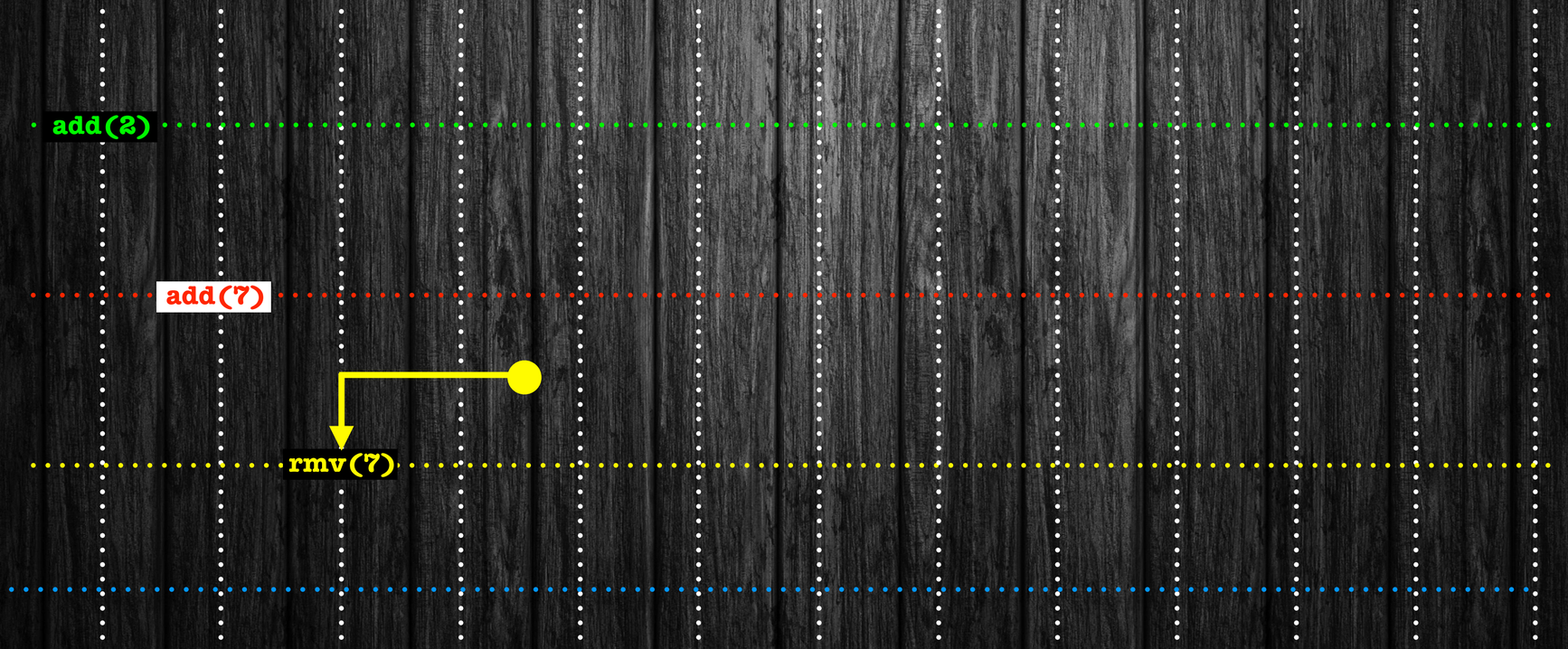
add(7)

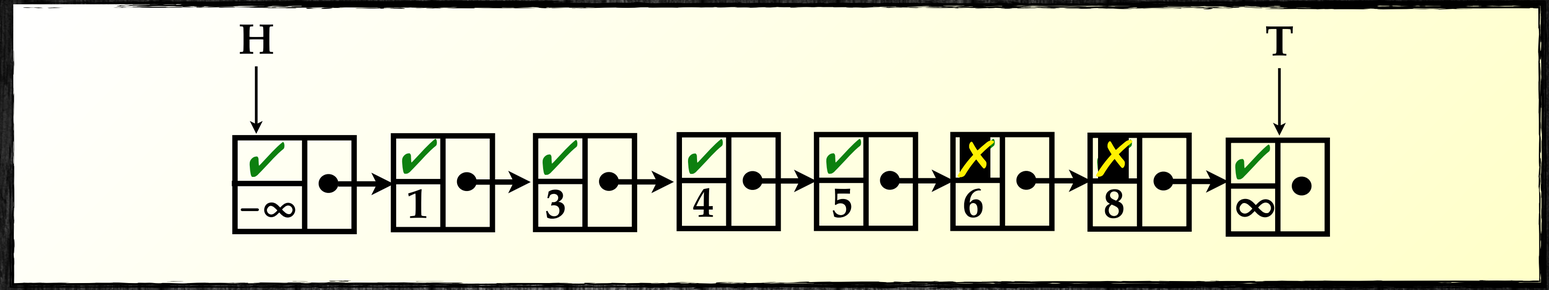
rmv(7)

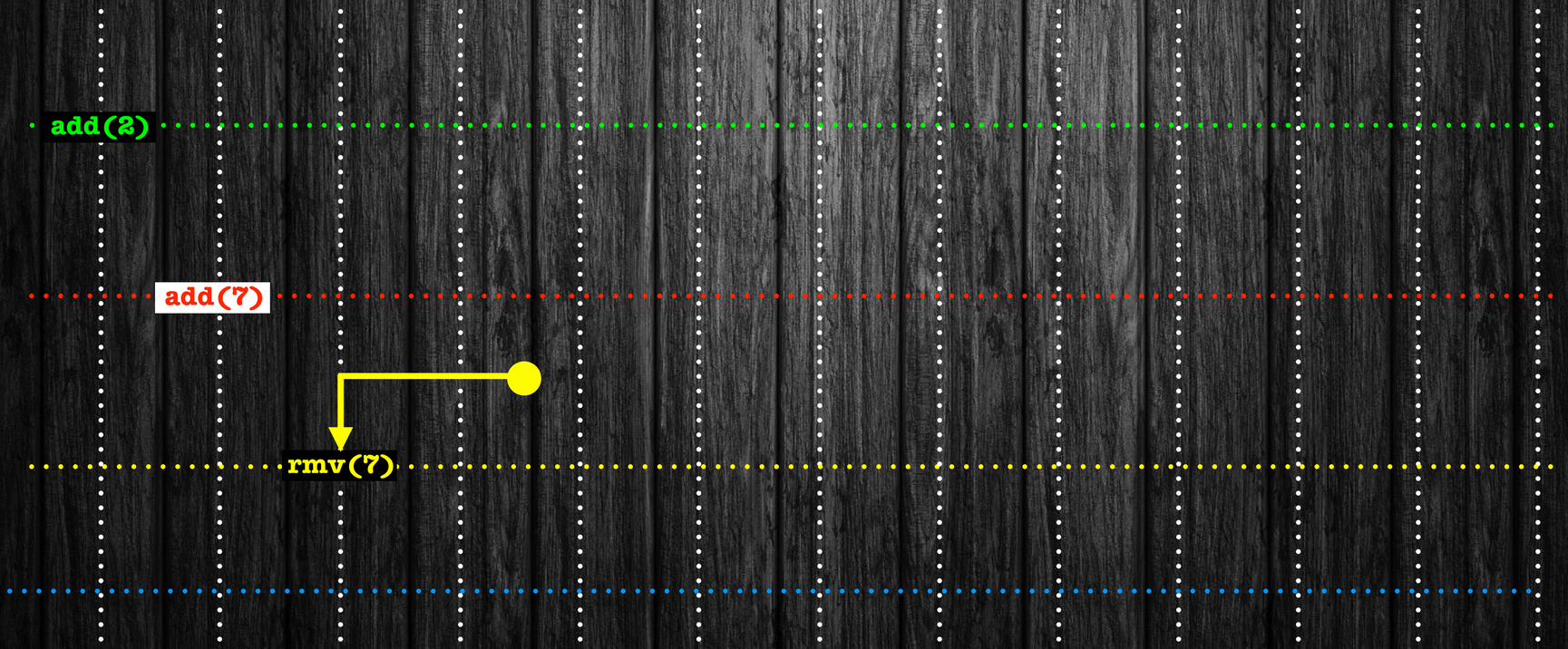
# Sets | Fine-Grained Synchronization | Linearization Policy



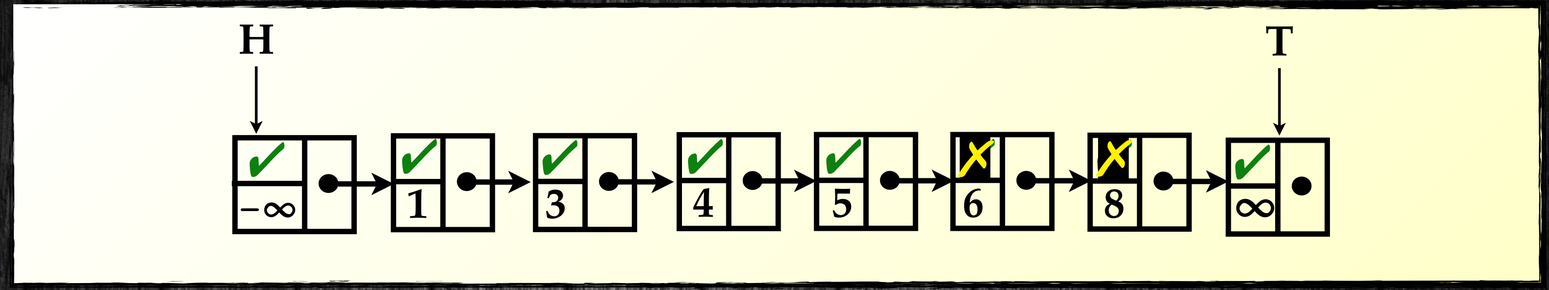


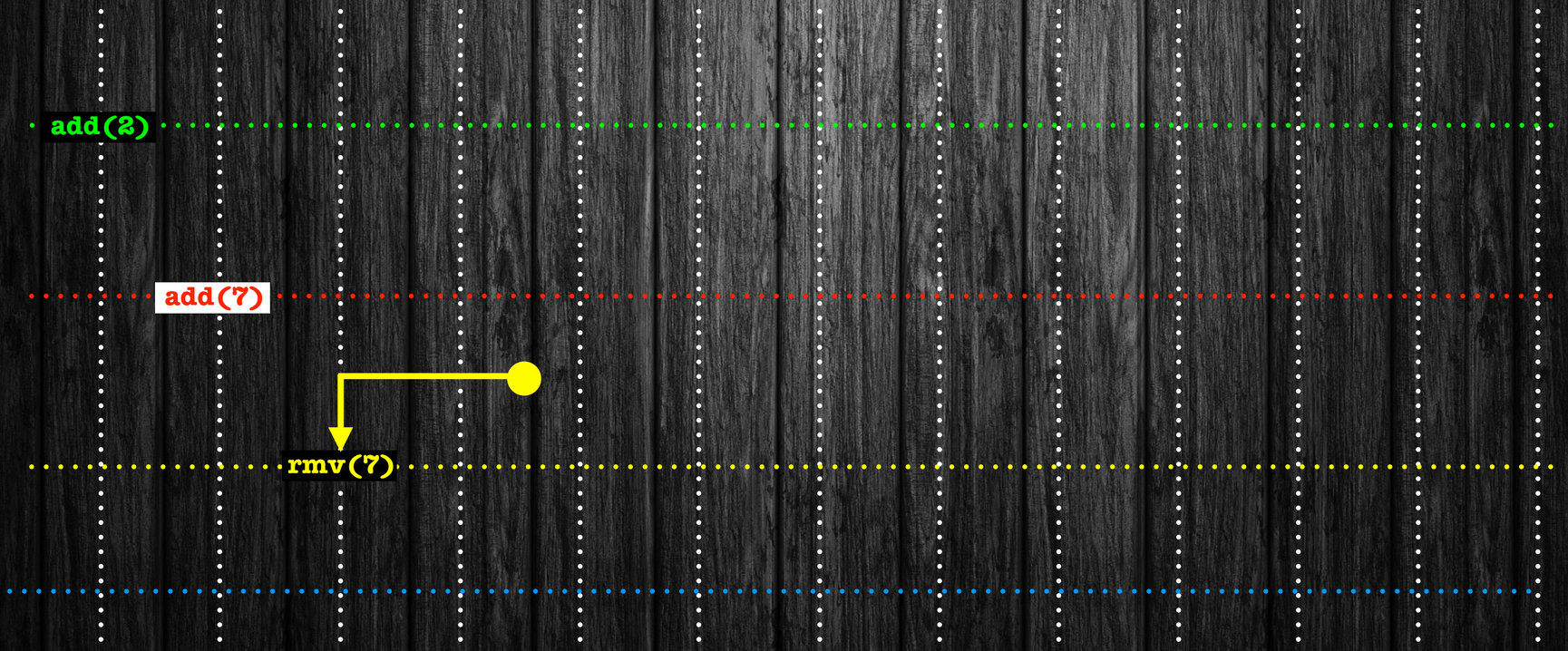

H
T
✓
−∞
✓
1
✓
3
✓
4
✓
5
✗
6
✗
8
✓
∞


add(2)
add(7)
rmv(7)

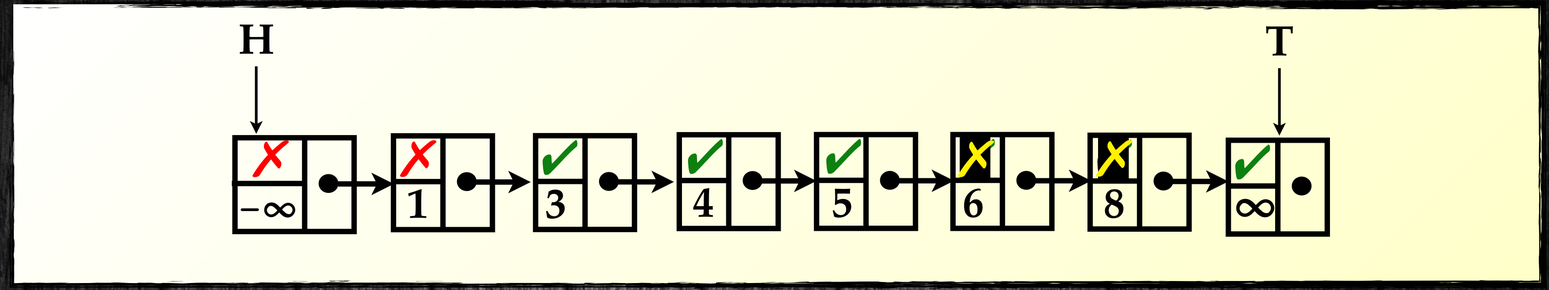
# Sets | Fine-Grained Synchronization | Linearization Policy



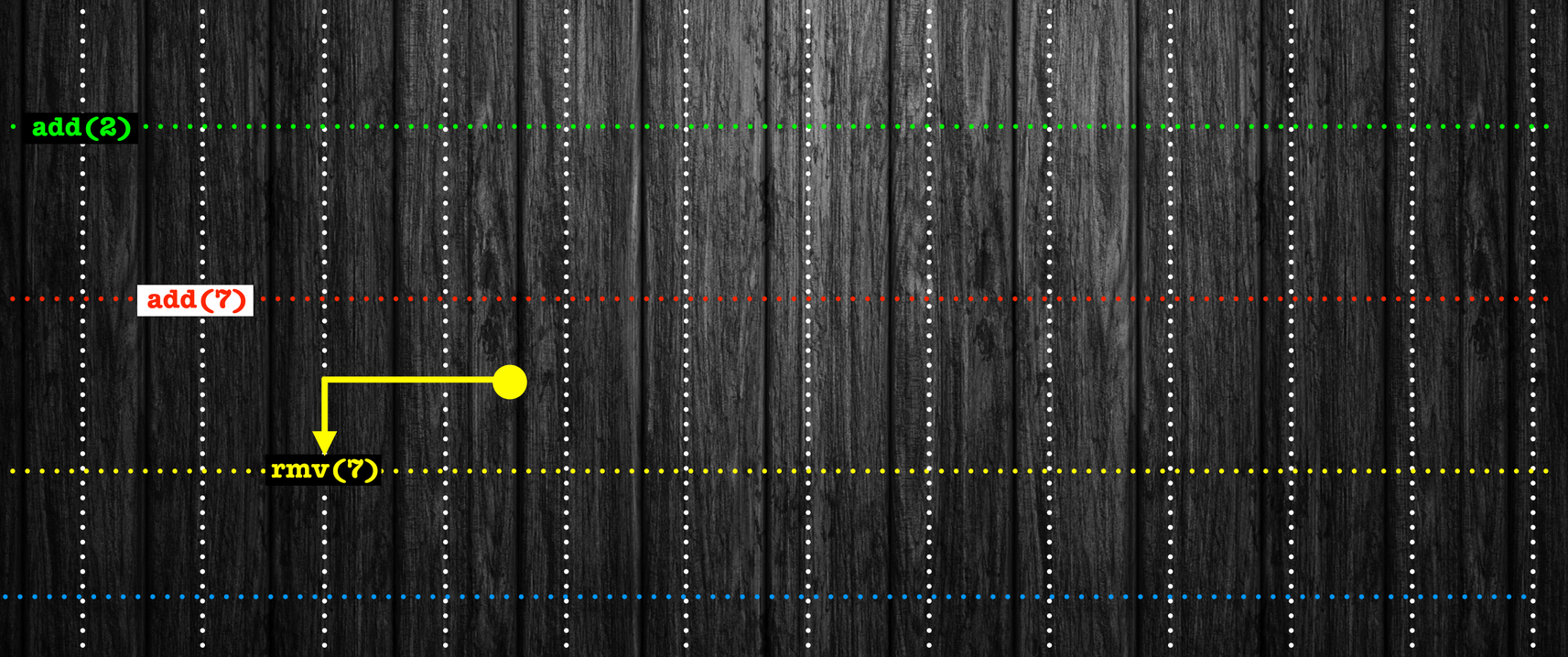


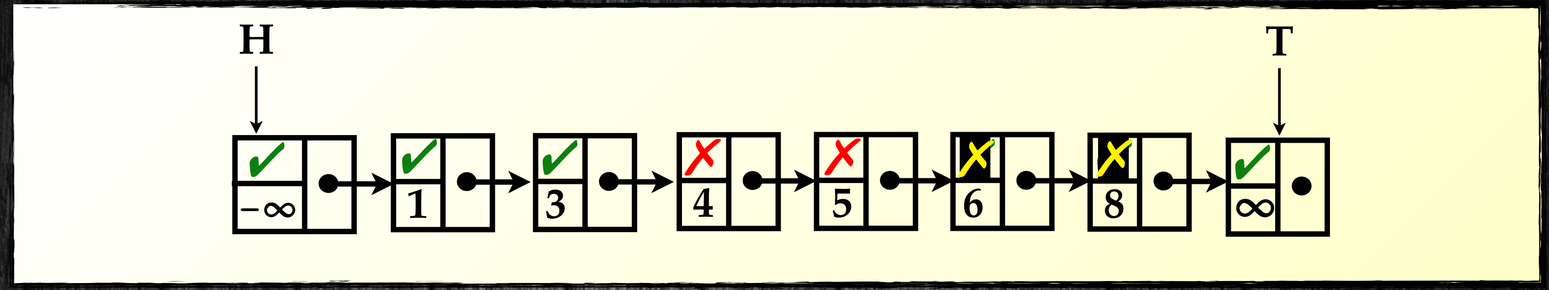
# Sets | Fine-Grained Synchronization | Linearization Policy



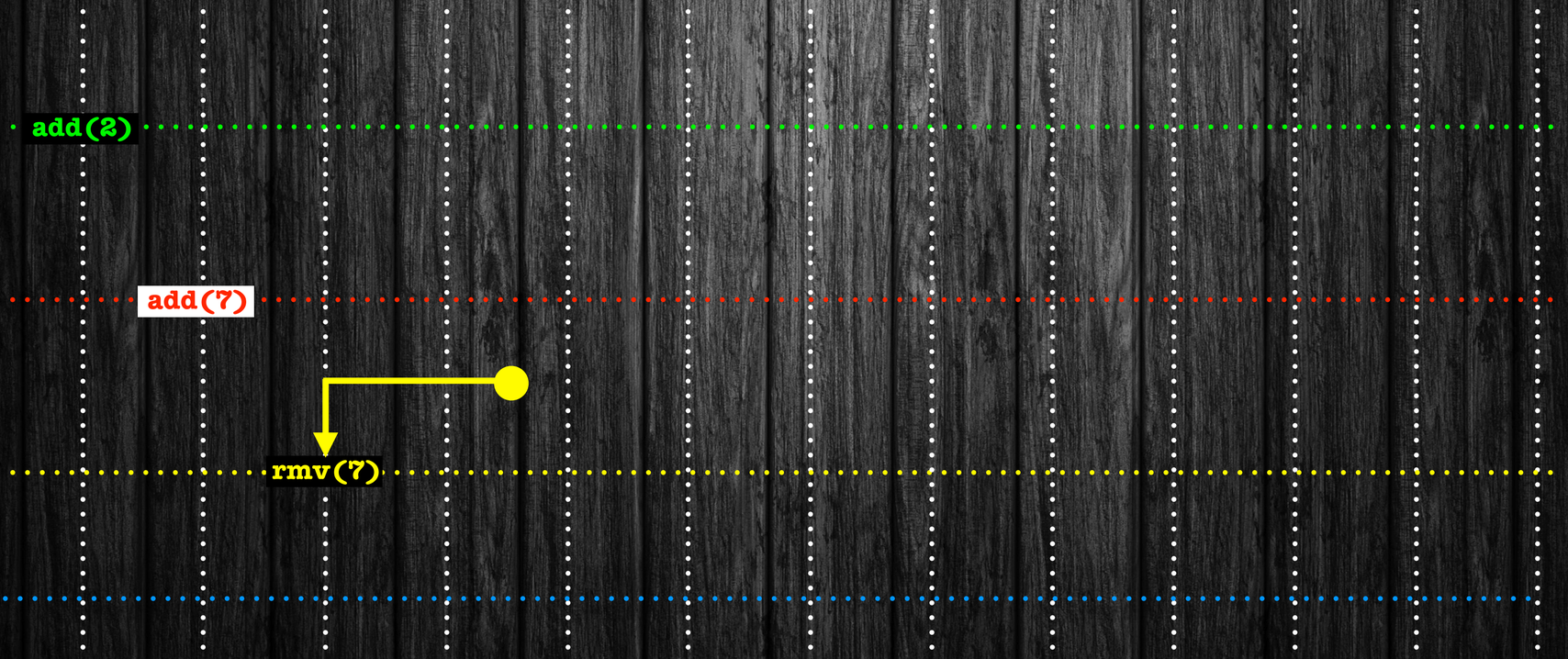


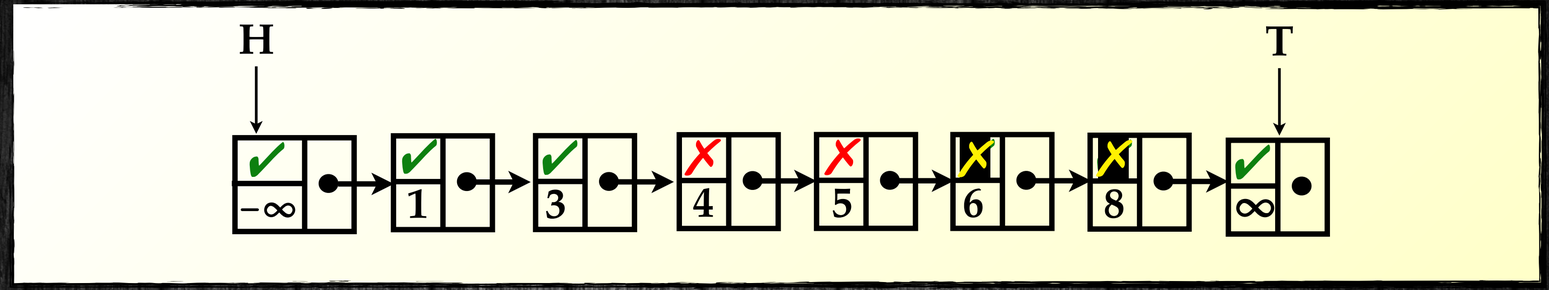



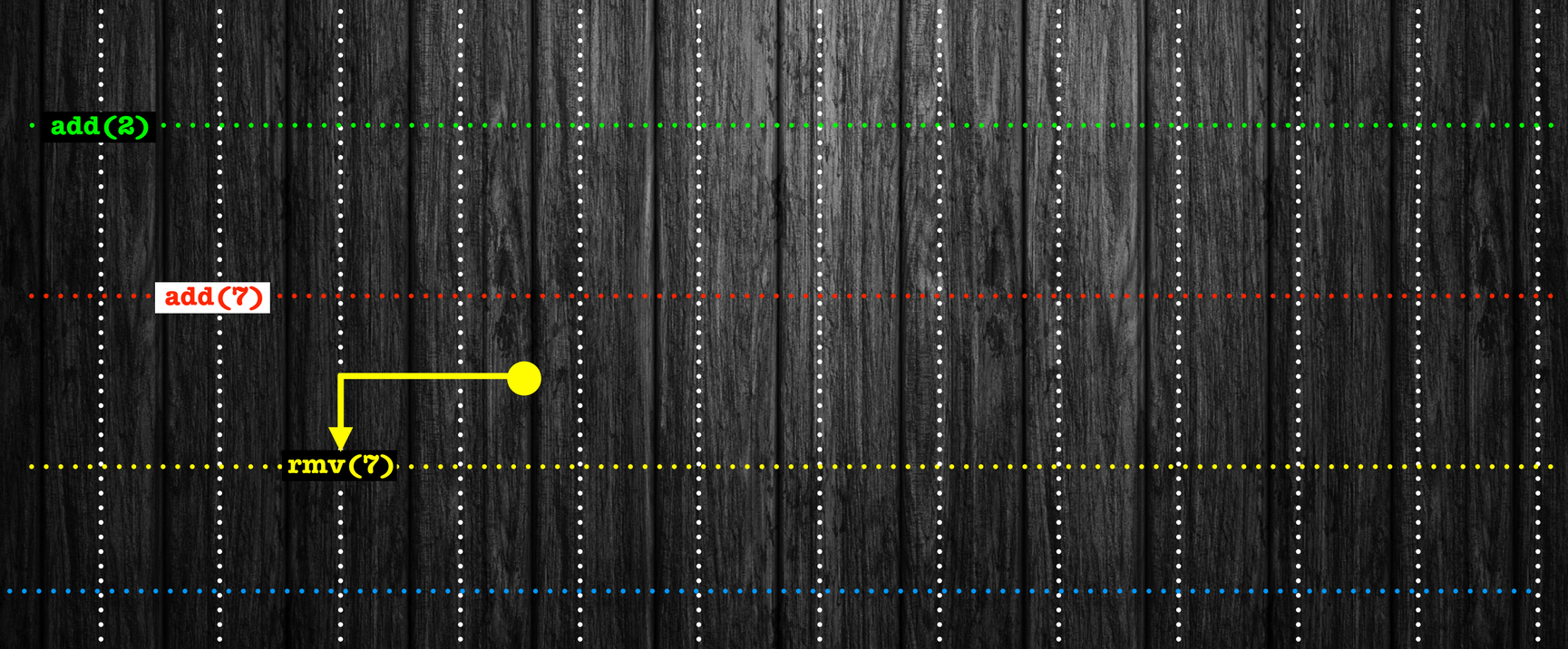
# Sets | Fine-Grained Synchronization | Linearization Policy



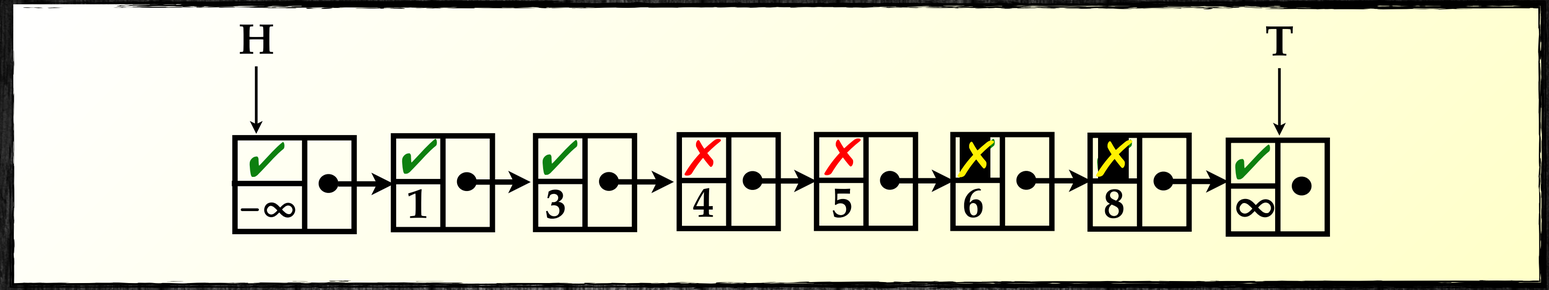


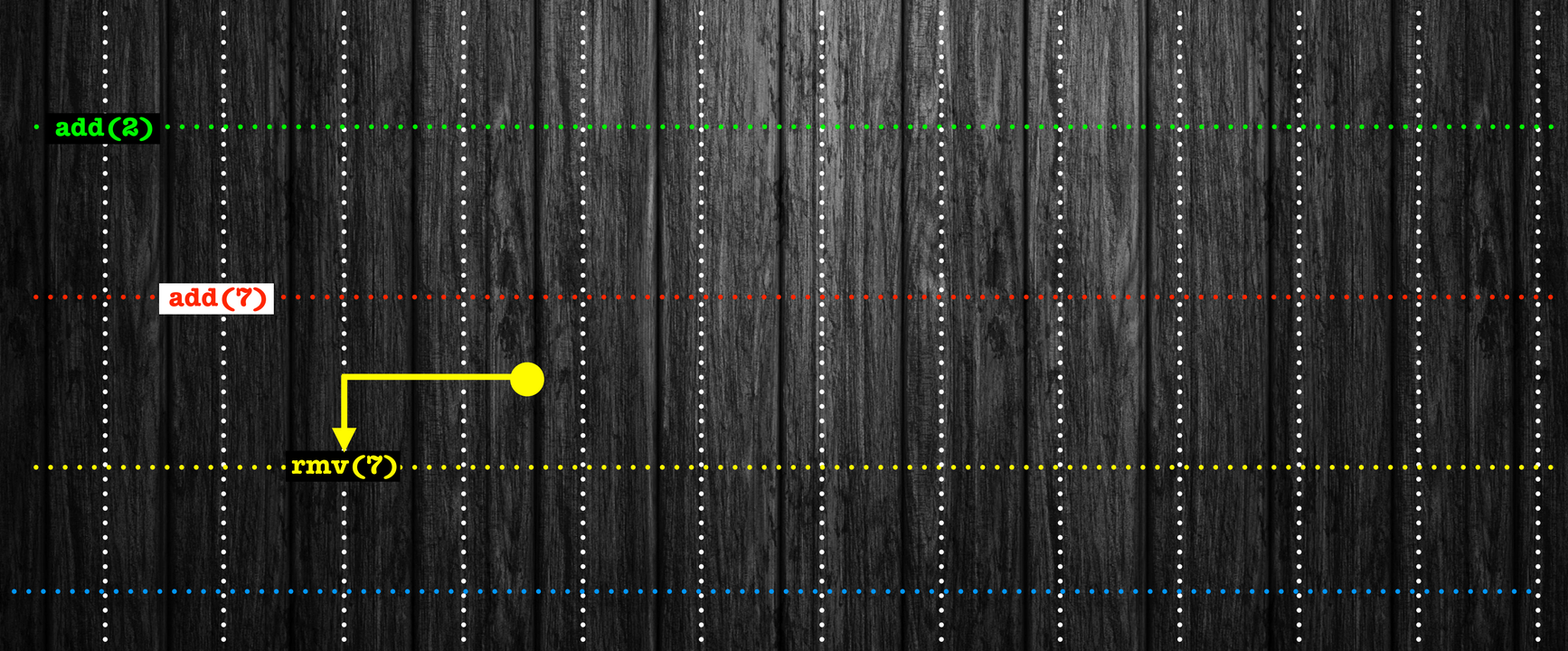

H
T
✓
−∞
✓
1
✓
3
✗
4
✗
5
✗
6
✗
8
✓
∞

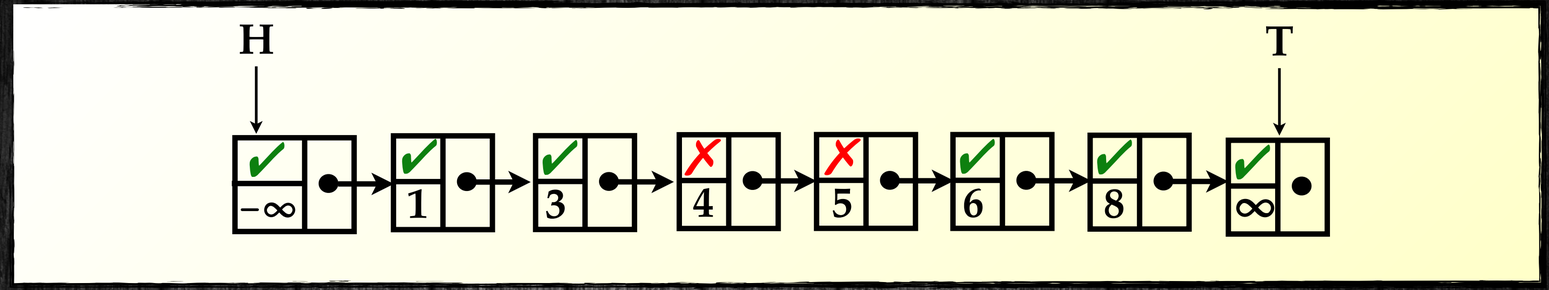

add(2)
add(7)
rmv(7)

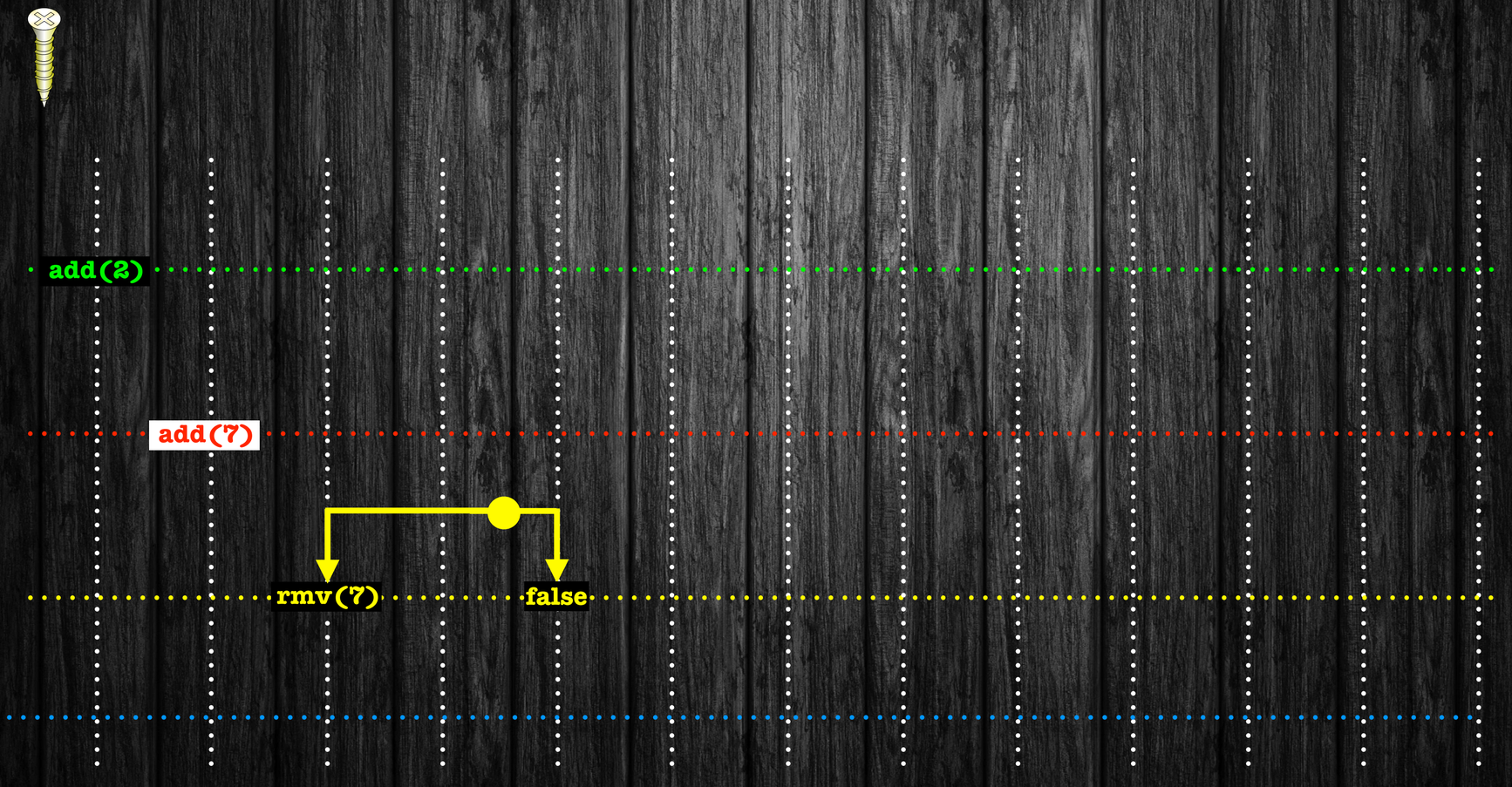
# Sets | Fine-Grained Synchronization | Linearization Policy

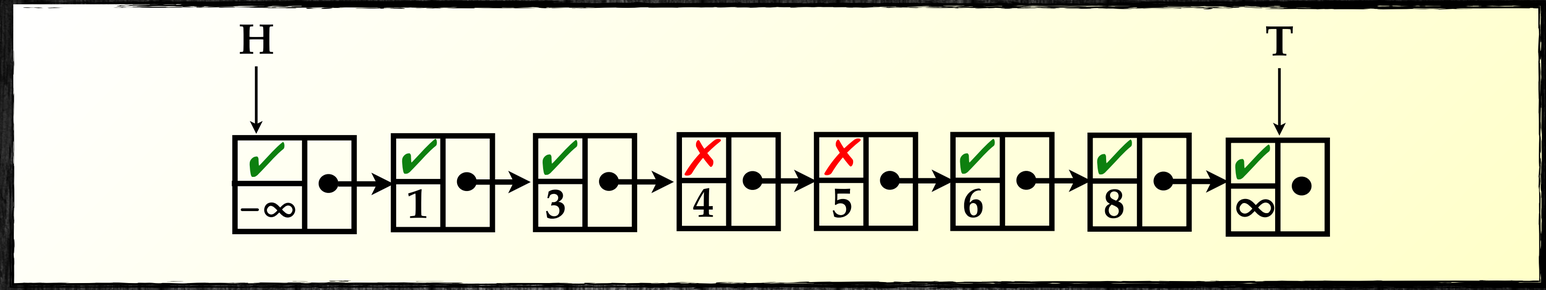


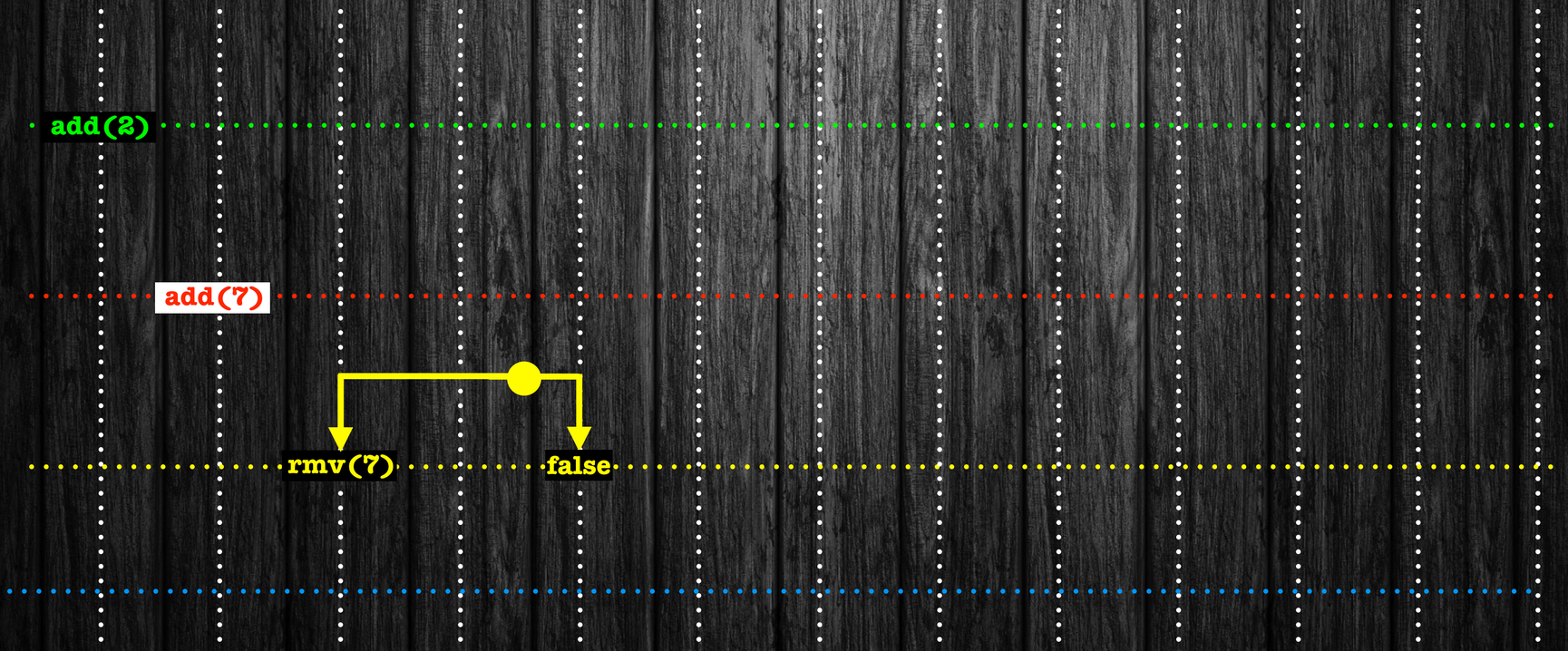


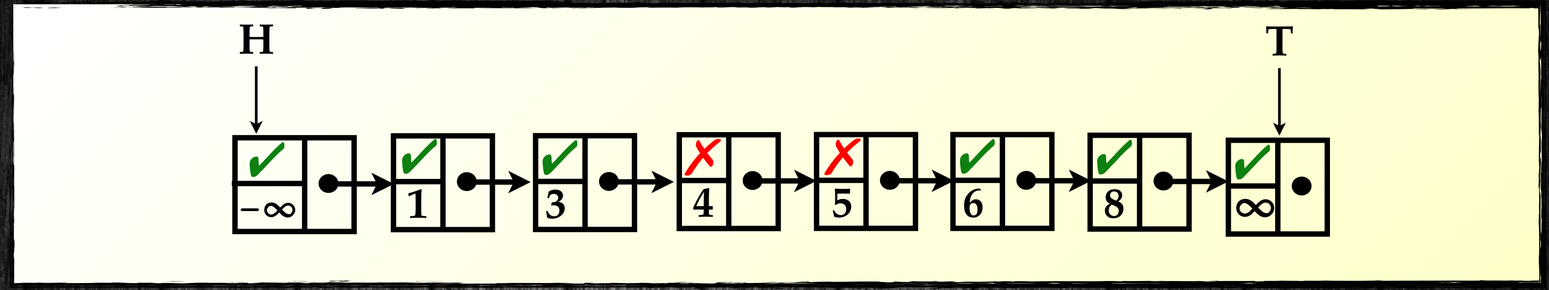

H
T
✓ −∞
✓ 1
✓ 3
✗ 4
✗ 5
✓ 6
✓ 8
✓ ∞


add(2)
add(7)
rmv(7)
false

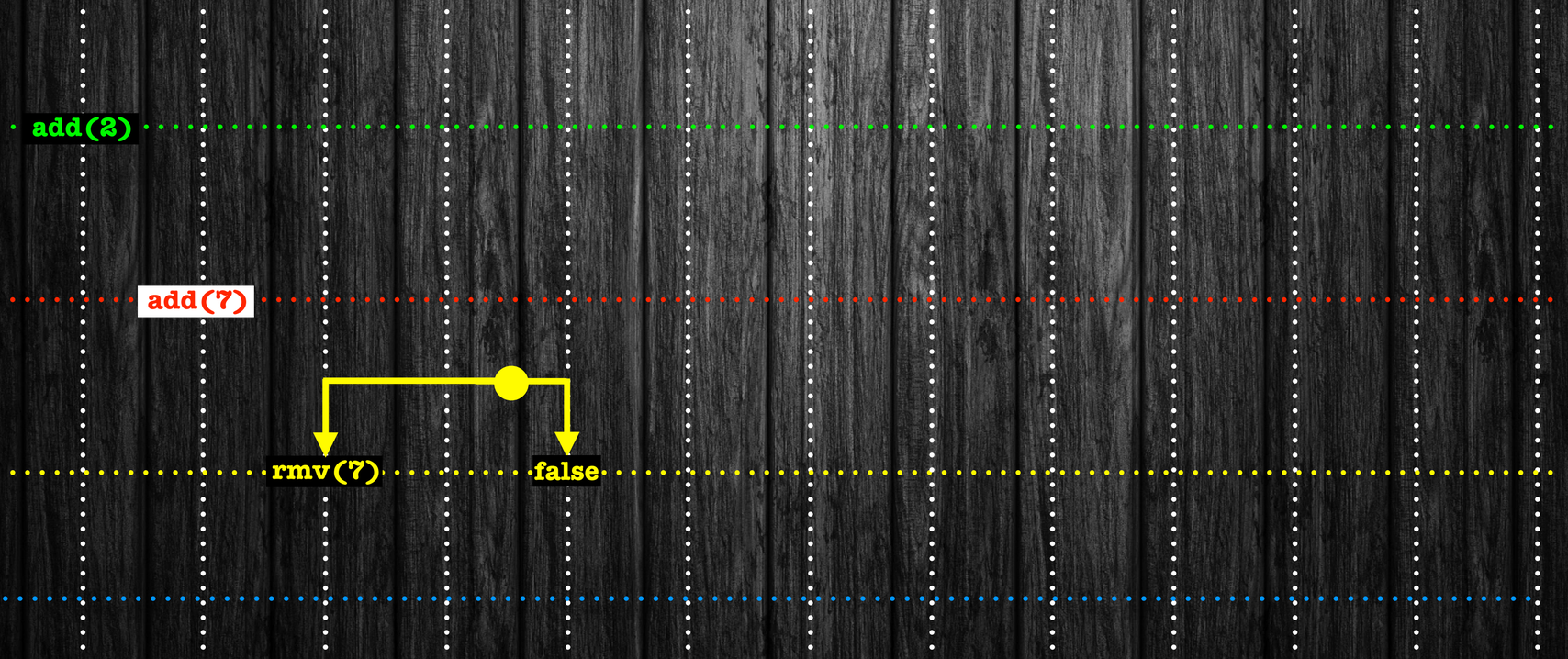

H
T
✓
−∞
✓
1
✓
3
✗
4
✗
5
✓
6
✓
8
✓
∞

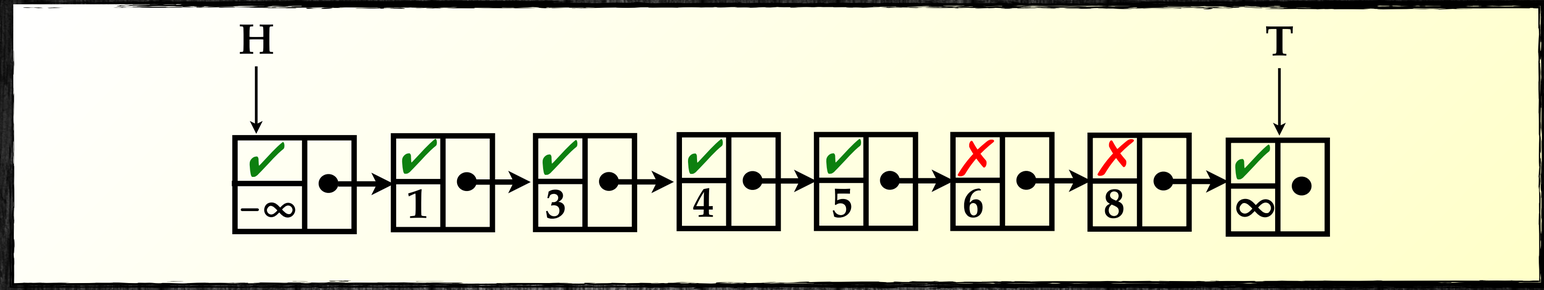

add(2)
add(7)
rmv(7)
false

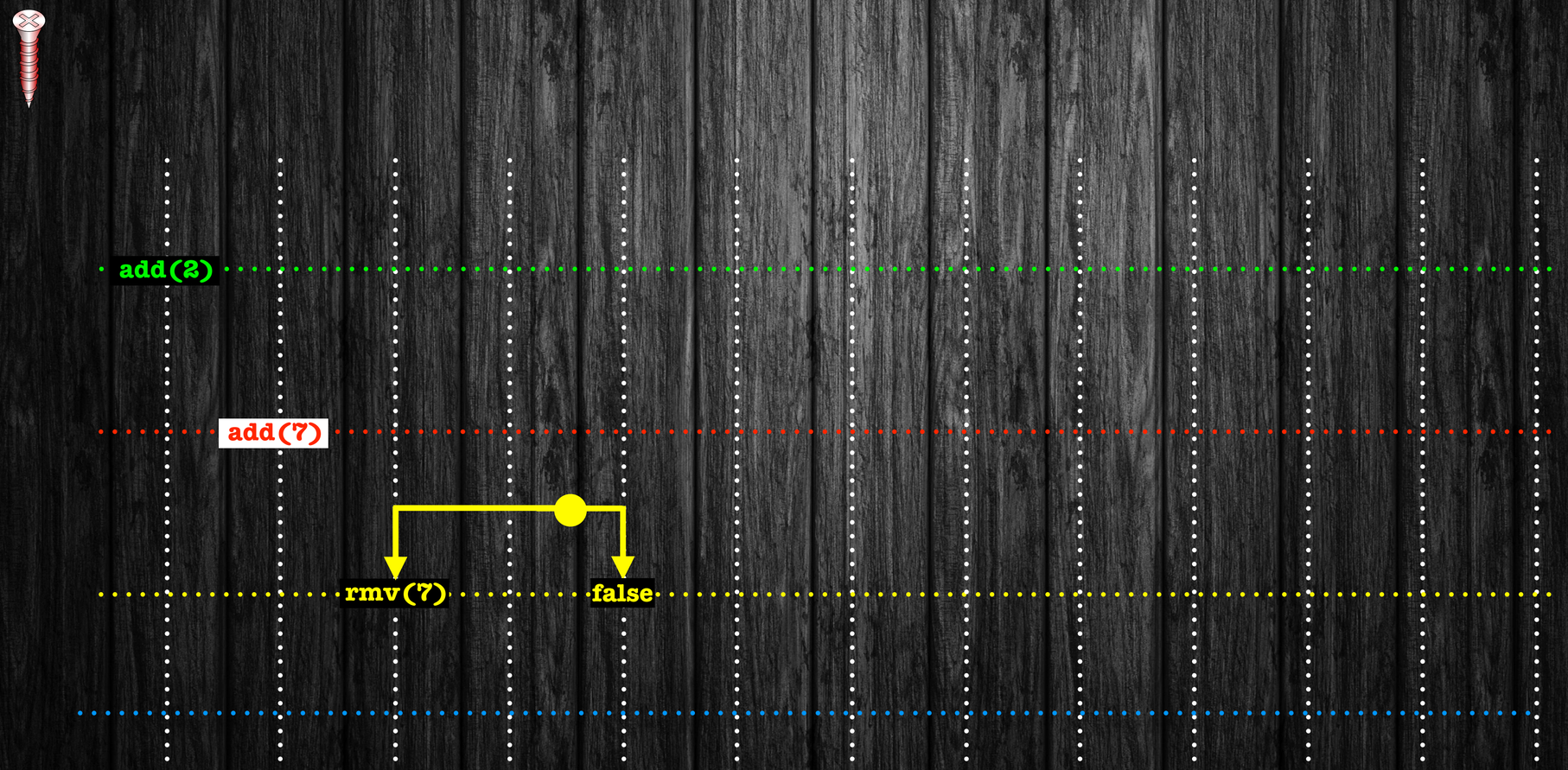
# Sets | Fine-Grained Synchronization | Linearization Policy

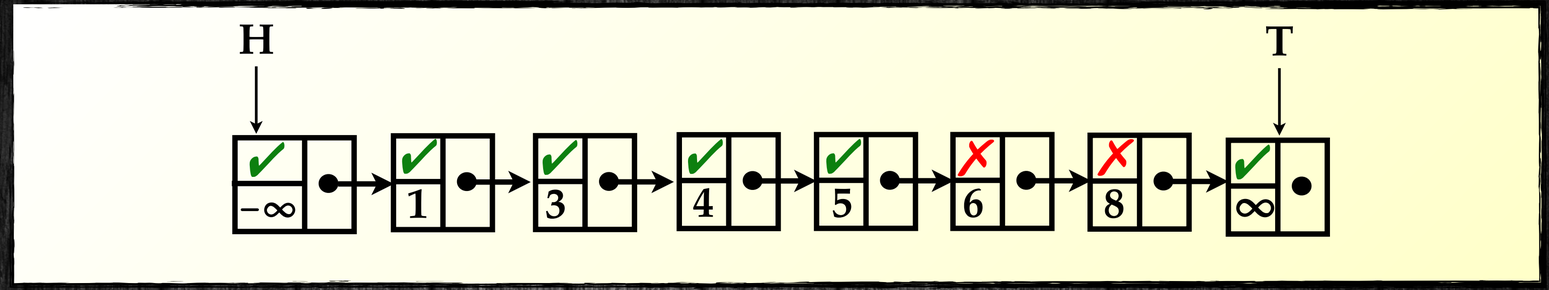




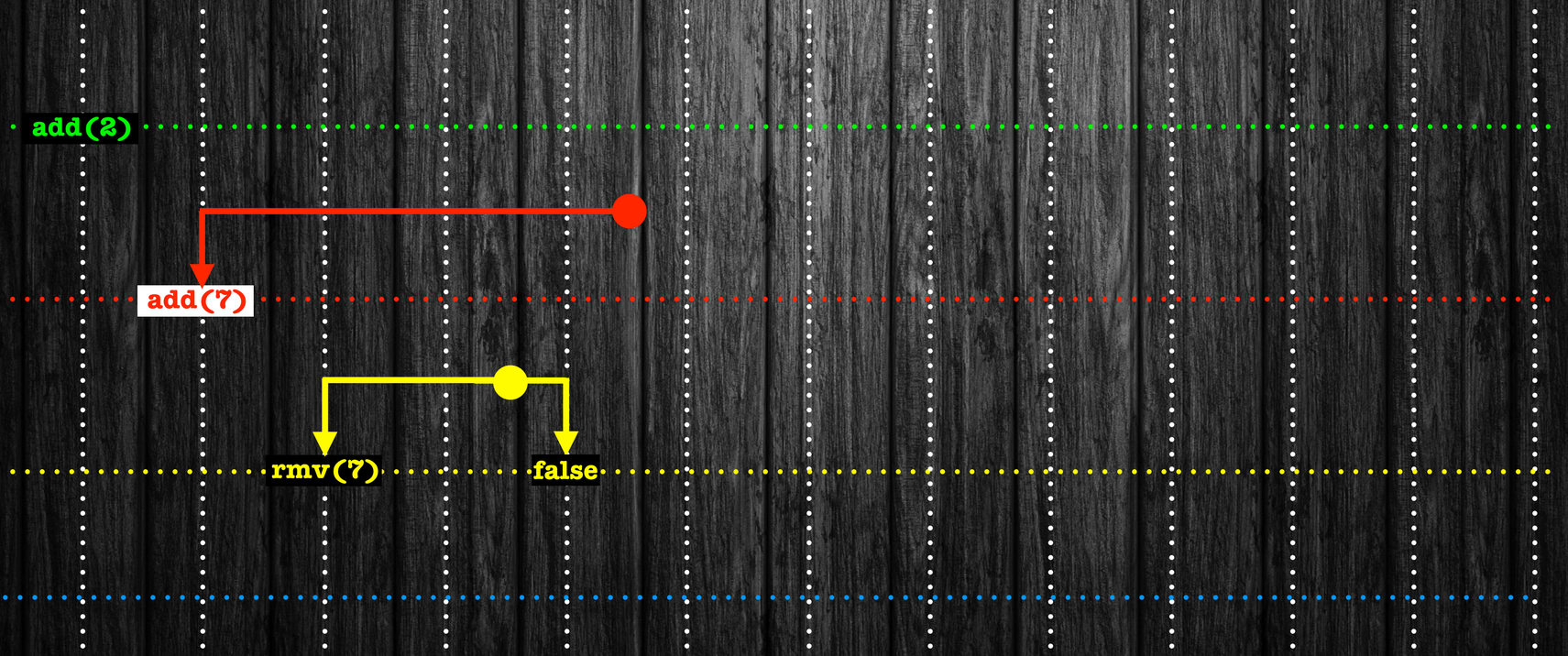




H
T
✓
−∞
✓
1
✓
3
✓
4
✓
5
✗
6
✗
8
✓
∞

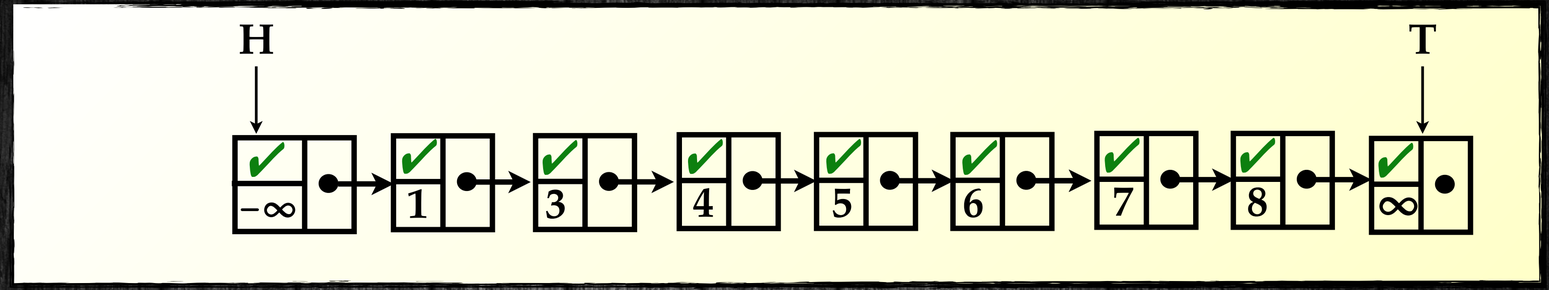

add(2)
add(7)
rmv(7)
false


H
T
✓
−∞
✓
1
✓
3
✓
4
✓
5
✗
6
✗
8
✓
∞

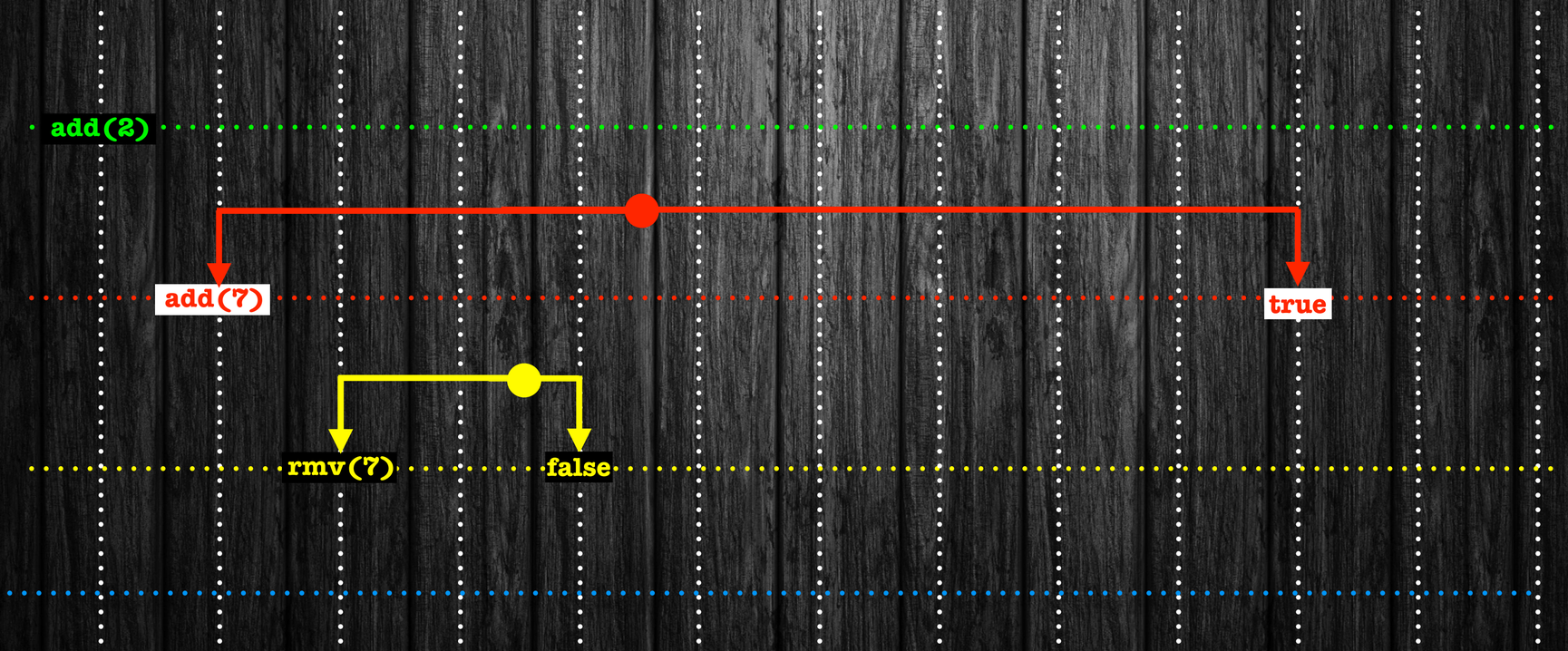


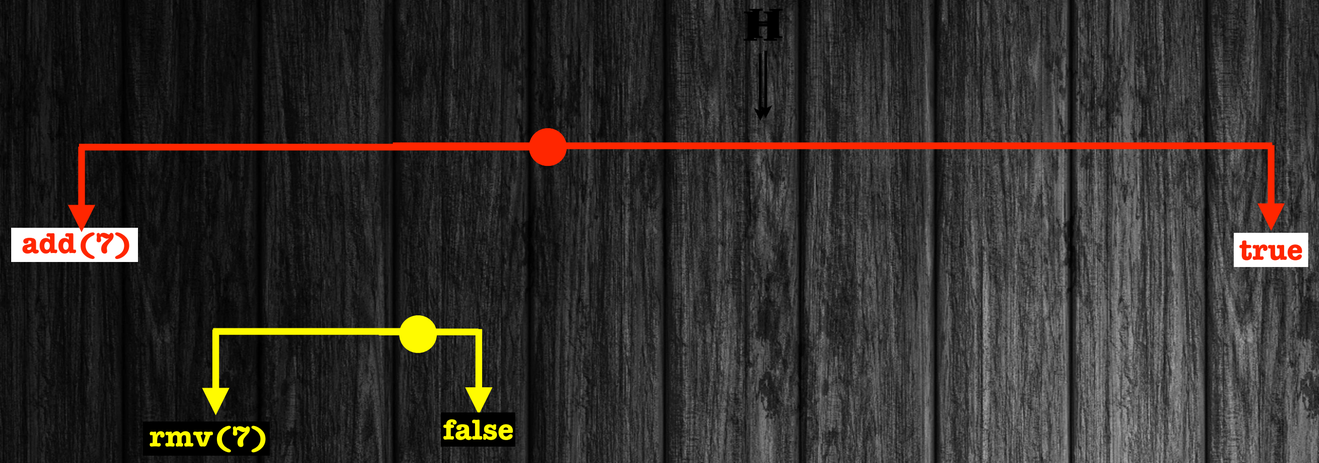

add(2)
add(7)
rmv(7)
false

# Sets | Fine-Grained Synchronization | Linearization Policy






H
T
add(7)
true
rmv(7)
false

# Sets | Fine-Grained Synchronization | Linearization Policy

