A Concurrent Program Logic with a Future and History

Roland Meyer, Thomas Wies, Sebastian Wolff

[OOPSLA'22]

Motivation

"***Programs = Algorithms + Data Structures***"

–Niklaus Wirth, 1978

Motivation

"***Programs = Algorithms + Data Structures***"

–Niklaus Wirth, 1978

Motivation

"***Programs = Algorithms + Data Structures***"

–Niklaus Wirth, 1978

Verification via Separation Logic (SL)

Motivation

"***Programs = Algorithms + Data Structures***"

–Niklaus Wirth, 1978

Verification via Separation Logic (SL)

Automation

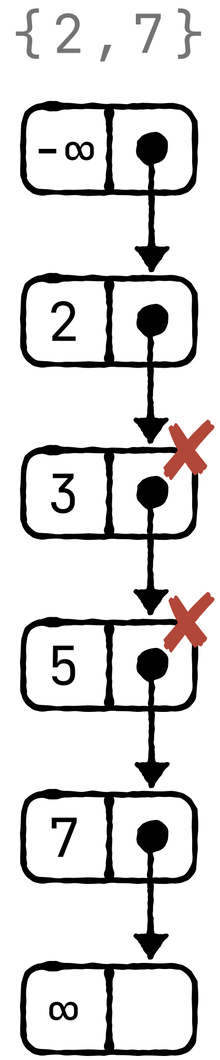
Verification Challenge 1: **Linearizability**

Verification Challenge 2: **Complex Unbounded Updates**

Harris' List: *contains*(7)

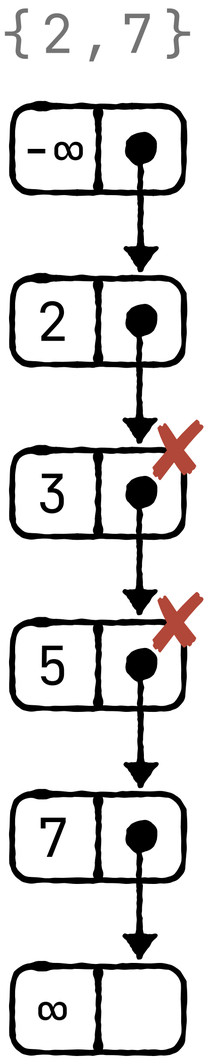
contains(k):

```
// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k;
```



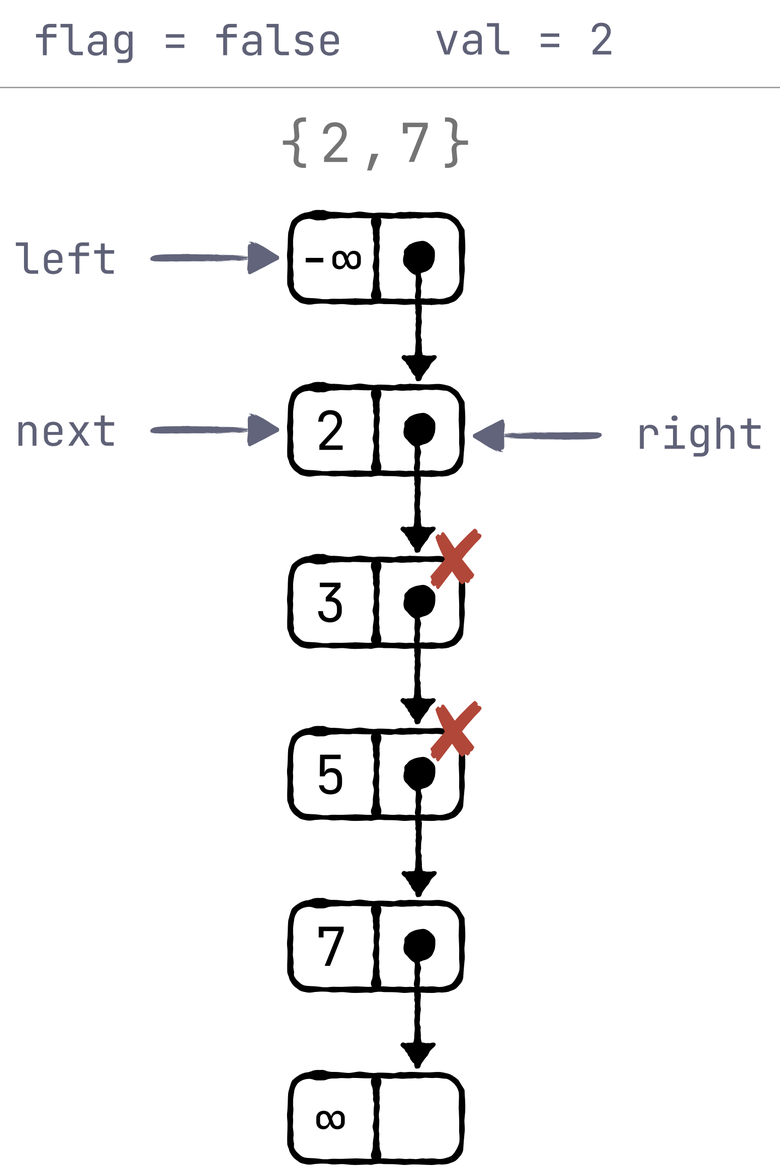
Harris' List: *contains*(7)

```
contains(k):
// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k;
```



Harris' List: *contains*(7)

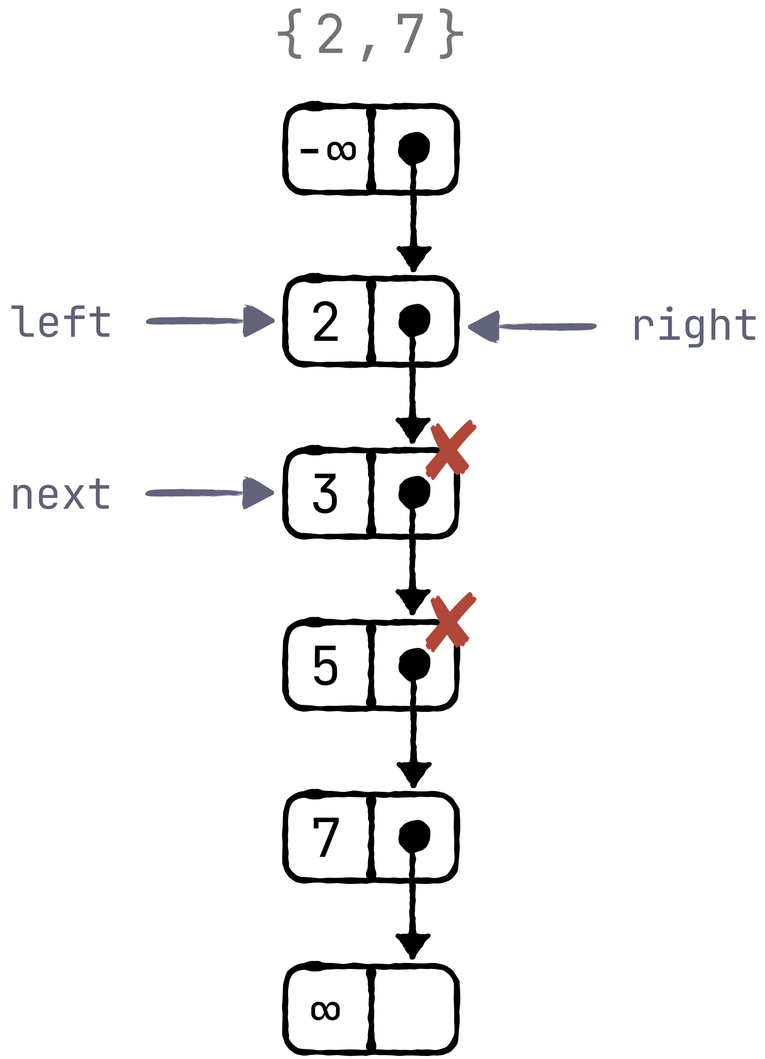
```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k;
```



Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k;
```

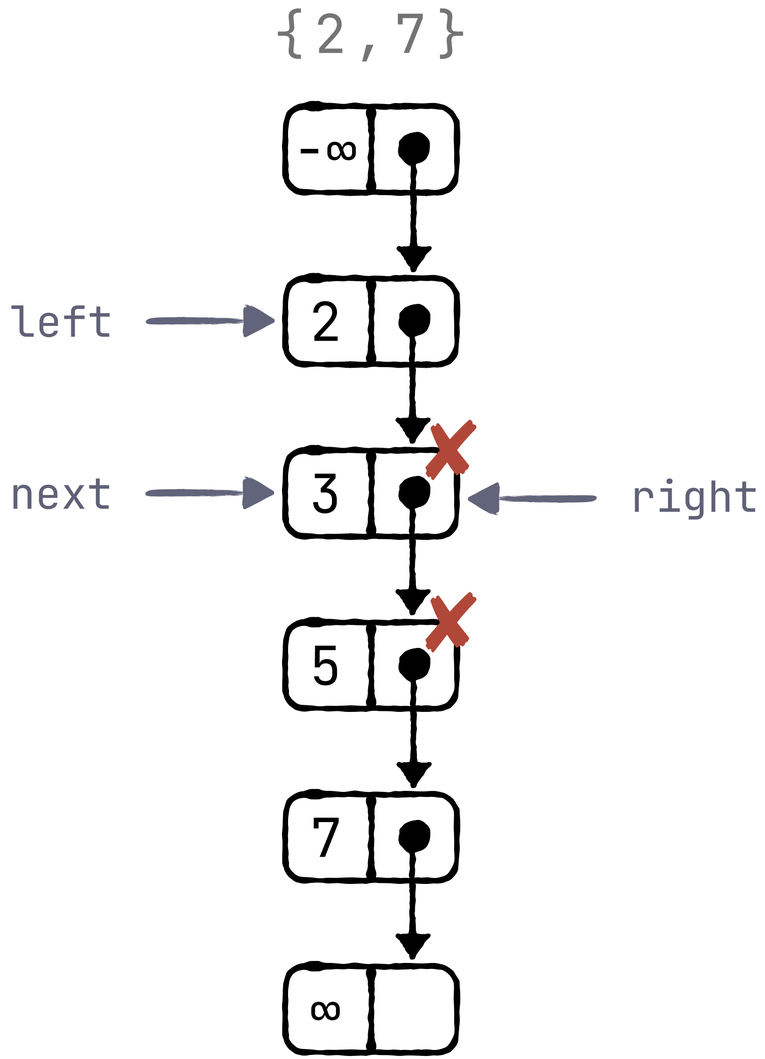
flag = false val = 2



Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k;
```

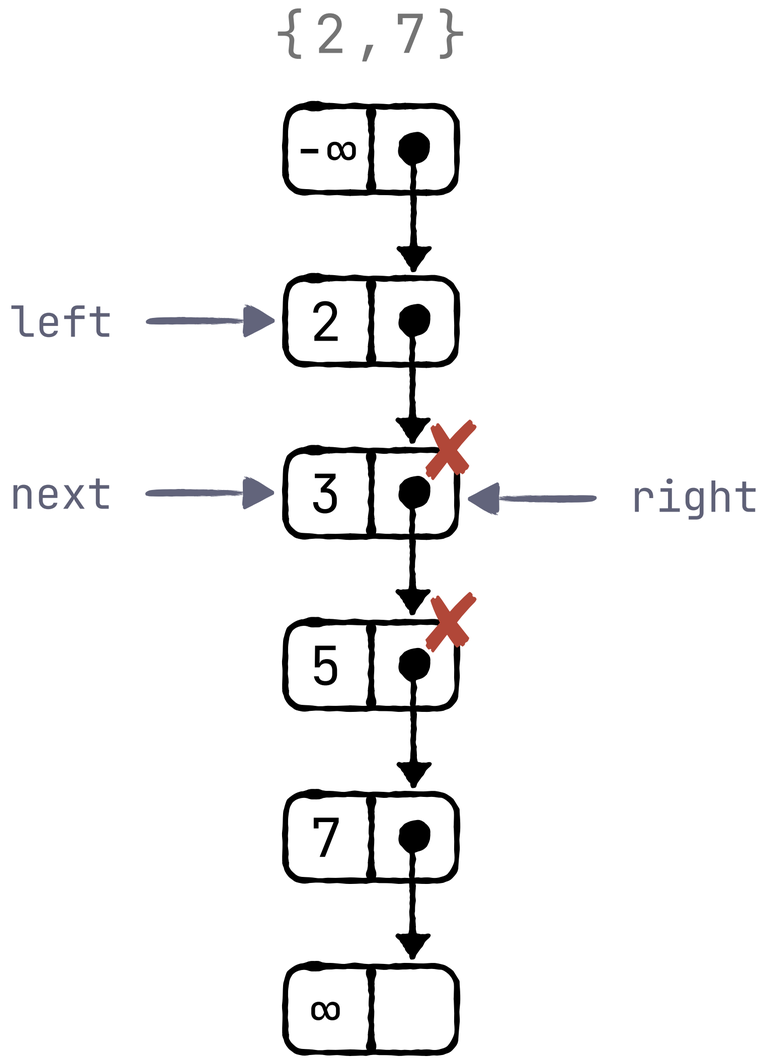
flag = false val = 2



Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k;
```

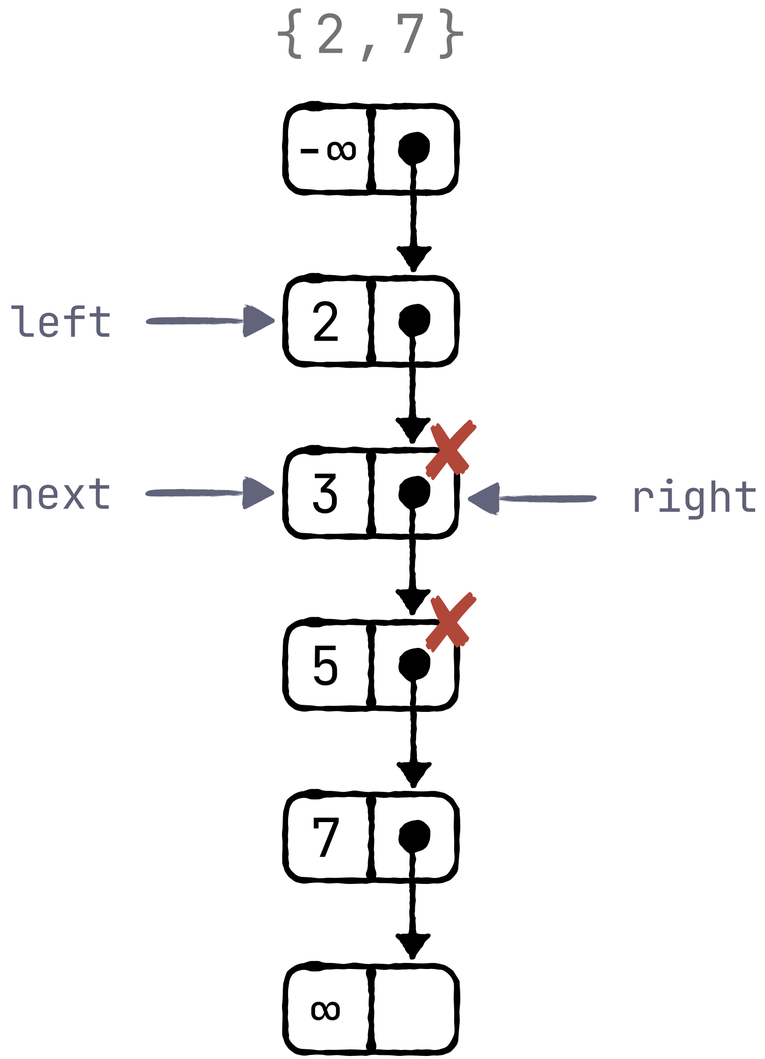
flag = true val = 3



Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k;
```

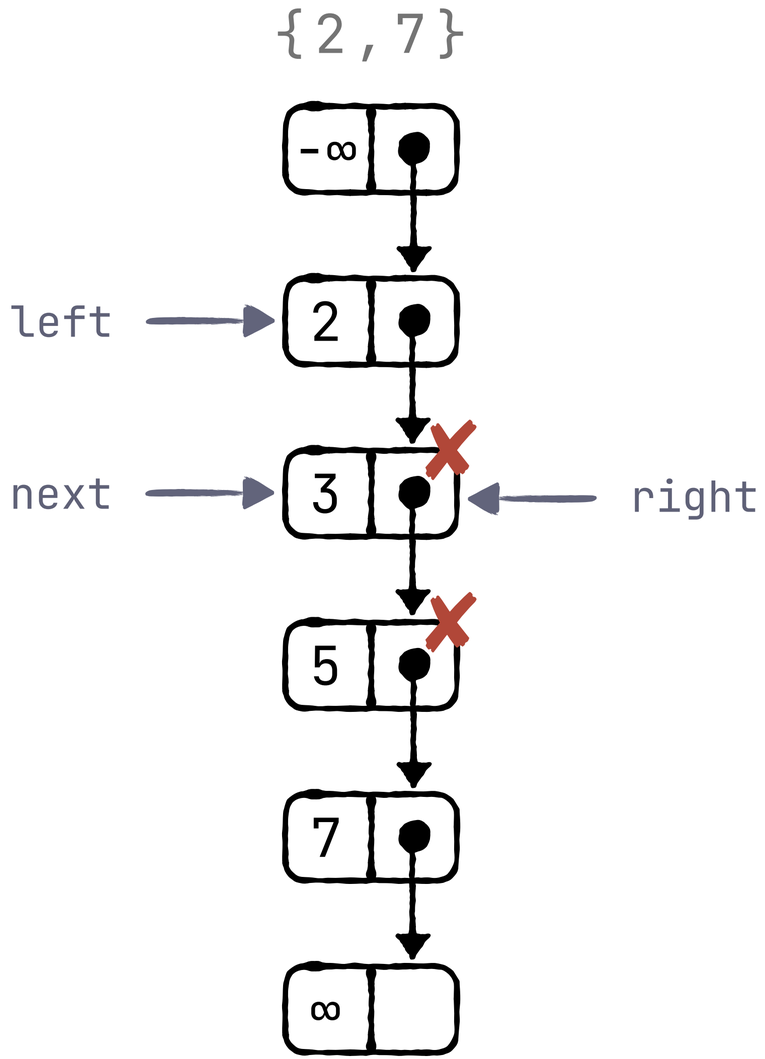
flag = true val = 3



Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k;
```

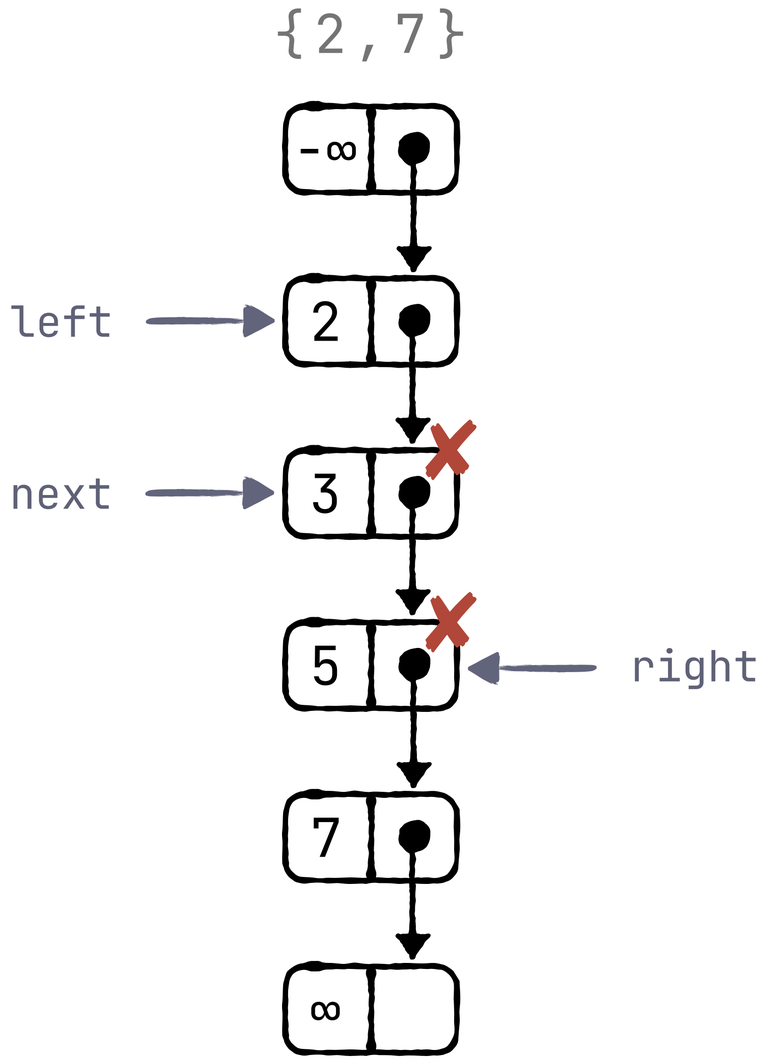
flag = true val = 3



Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k;
```

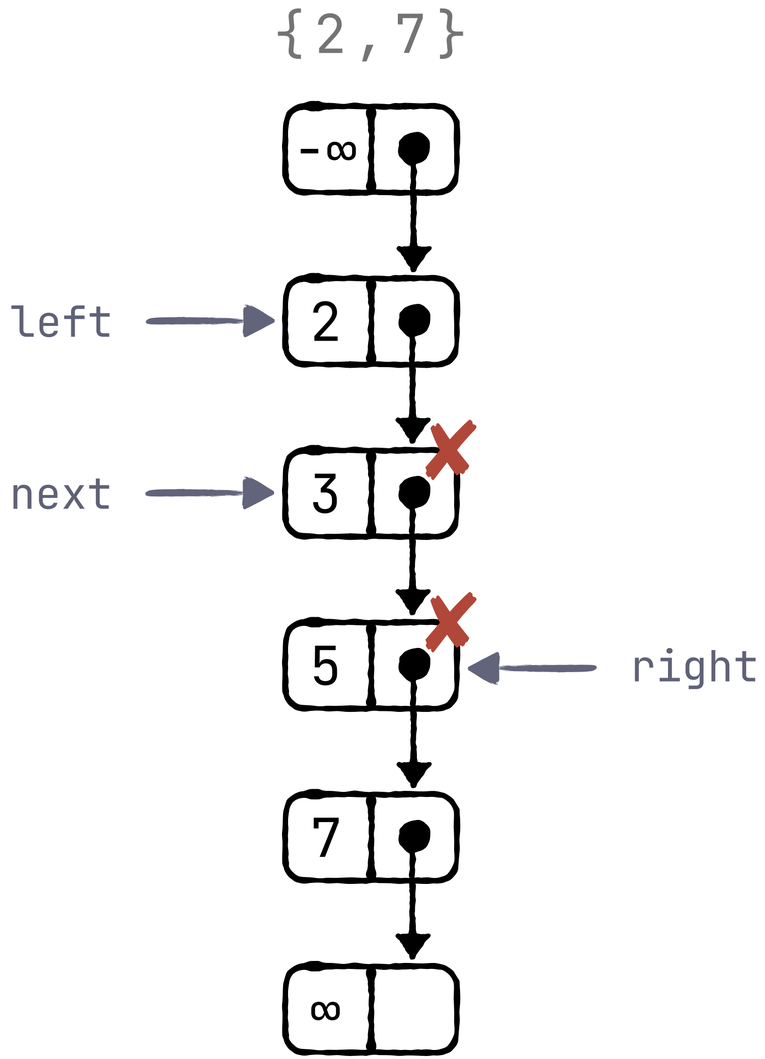
flag = true val = 3



Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k;
```

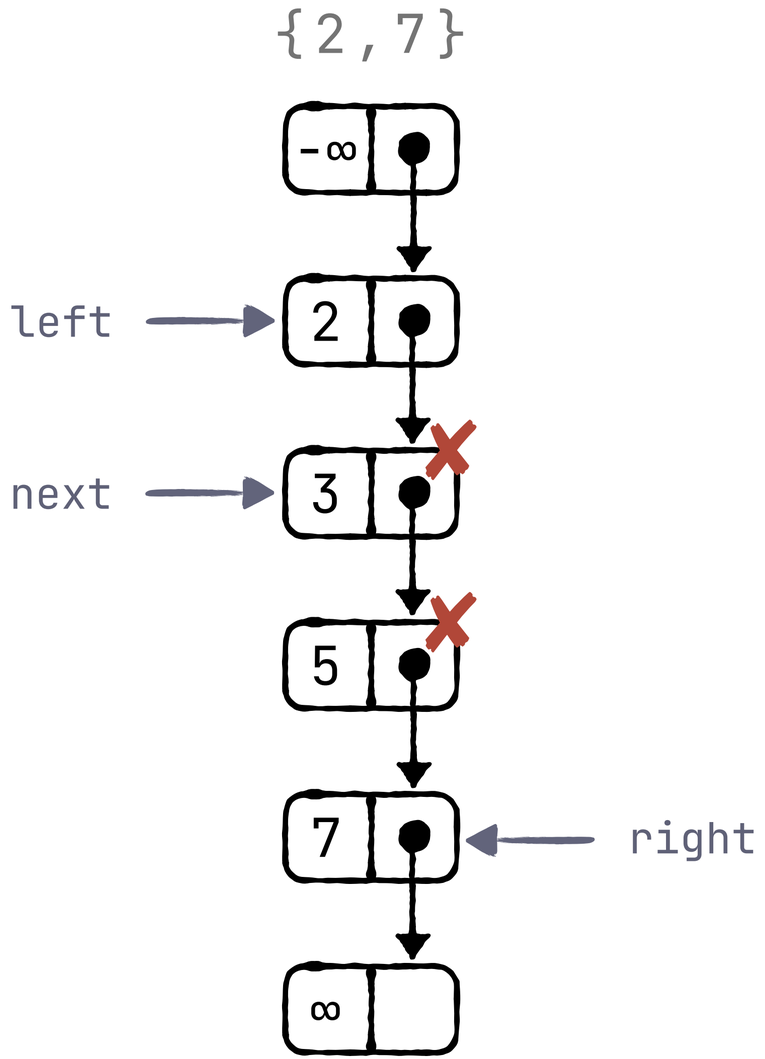
flag = true val = 5



Harris' List: *contains*(7)

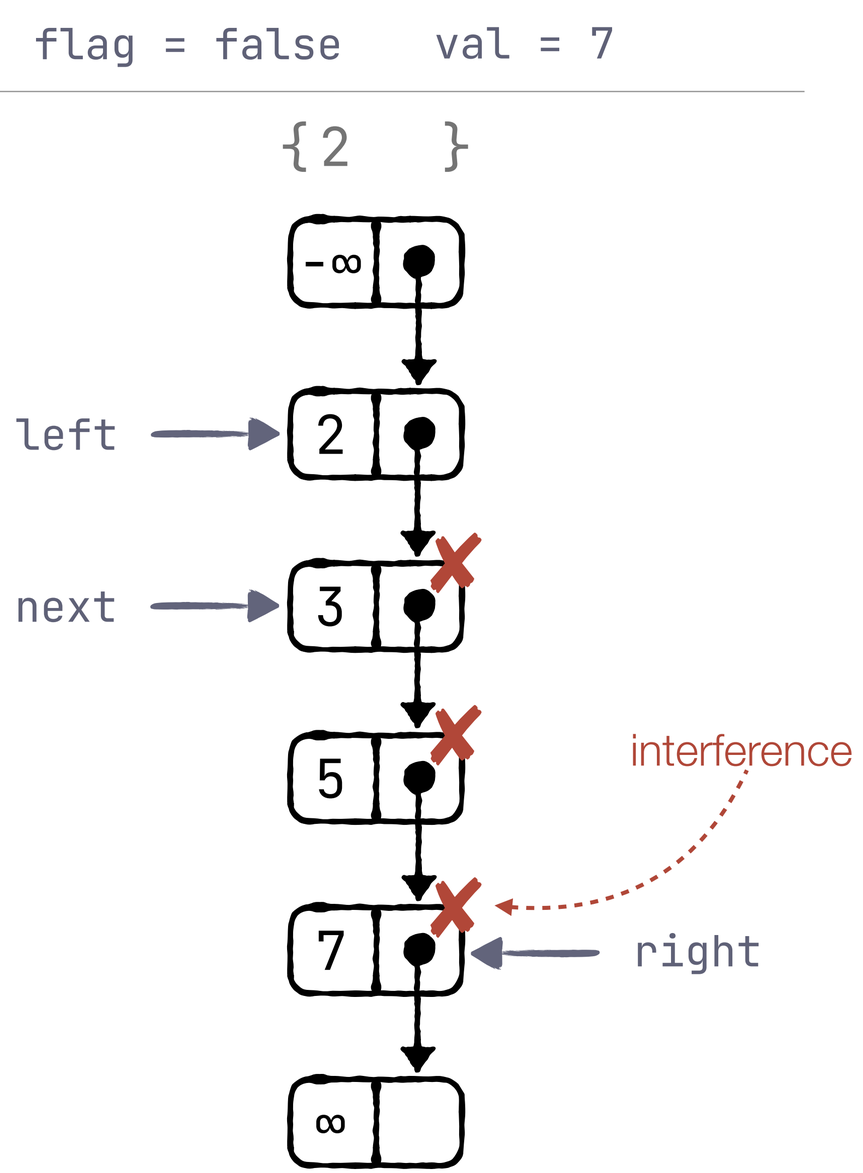
```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k;
```

flag = false val = 7



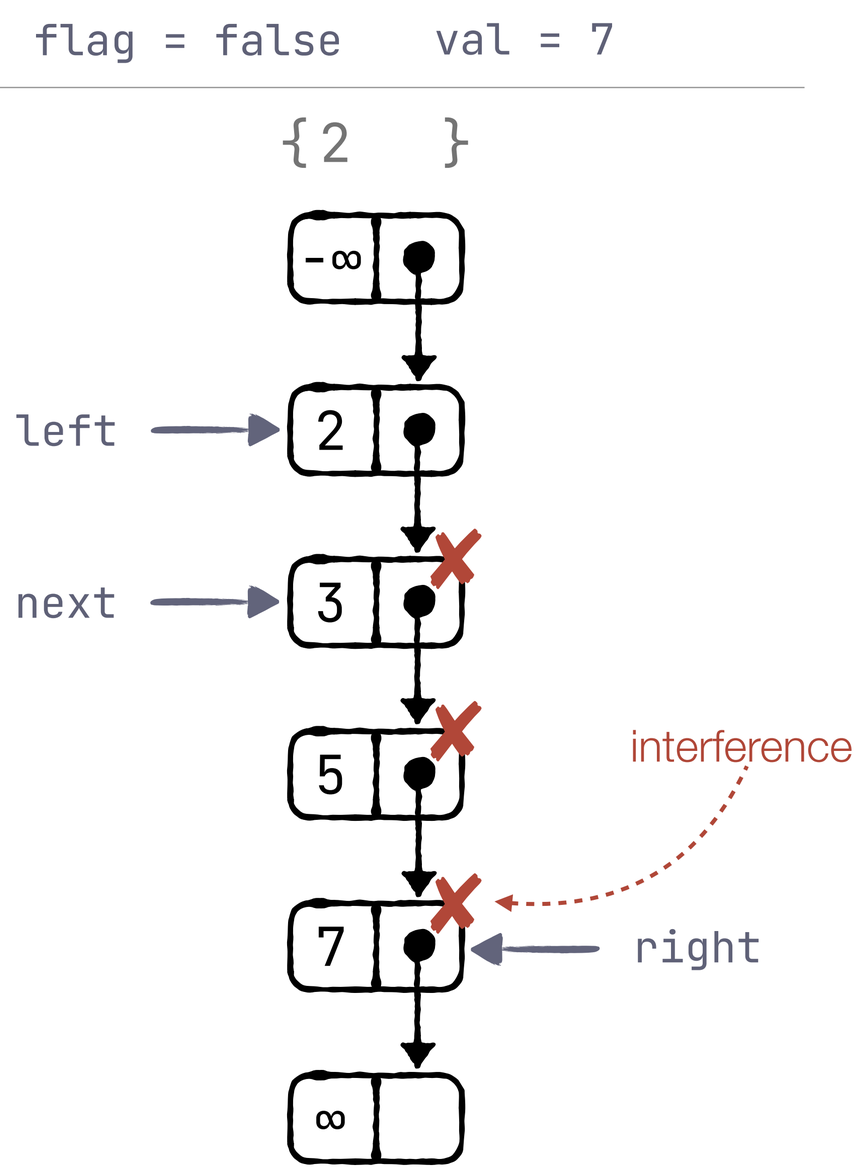
Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k;
```



Harris' List: *contains*(7)

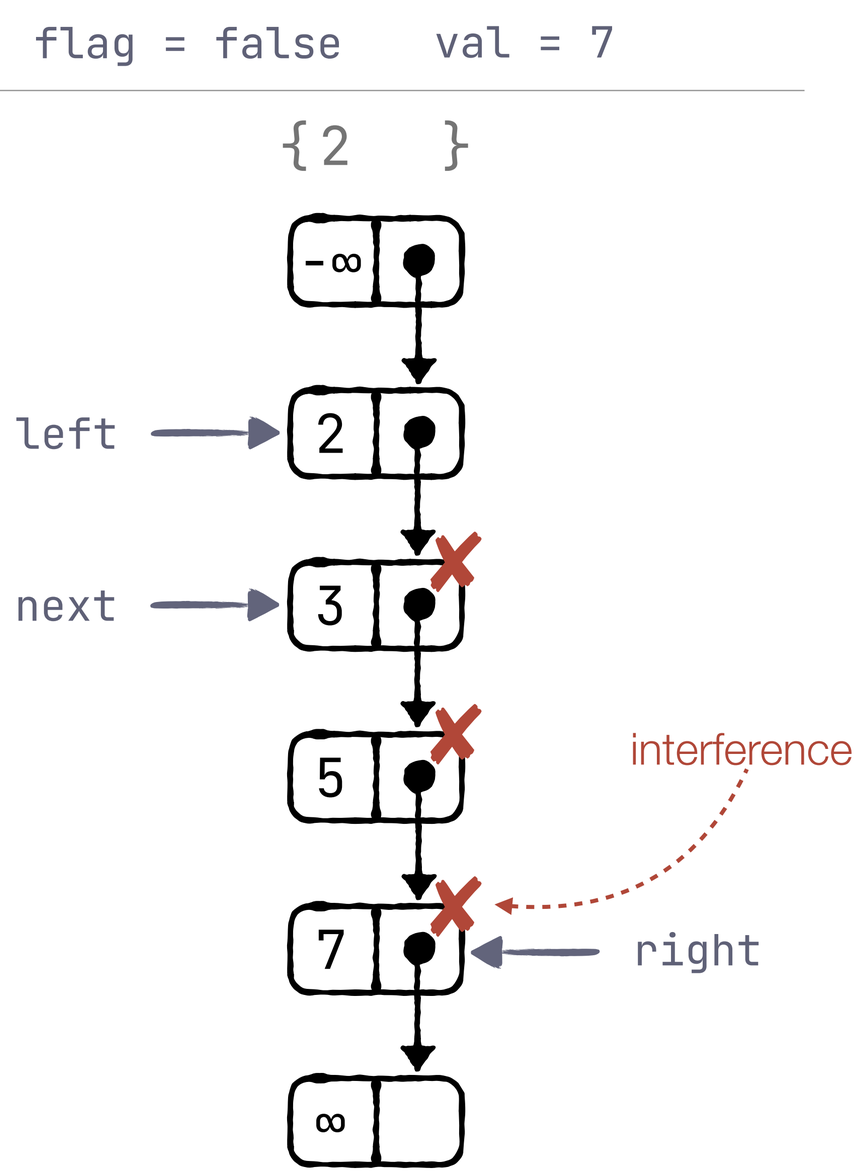
```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k;
```



Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k; // true
```

correct (*linearizable*)

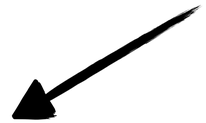


Linearizability

Problem: requires *hindsight* or *prophecies*

Linearizability

Problem: requires *hindsight* or *prophecies*



a posteriori reasoning

about bygone state

⇝ no SL integration

Linearizability

Problem: requires *hindsight* or *prophecies*

a posteriori reasoning

about bygone state

⇝ no SL integration

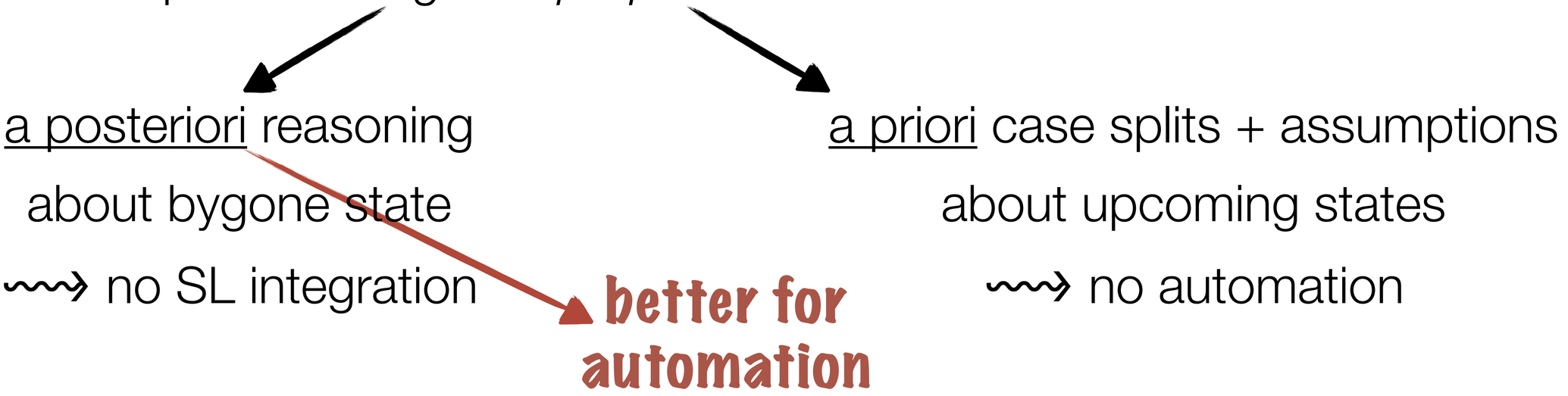
a priori case splits + assumptions

about upcoming states

⇝ no automation

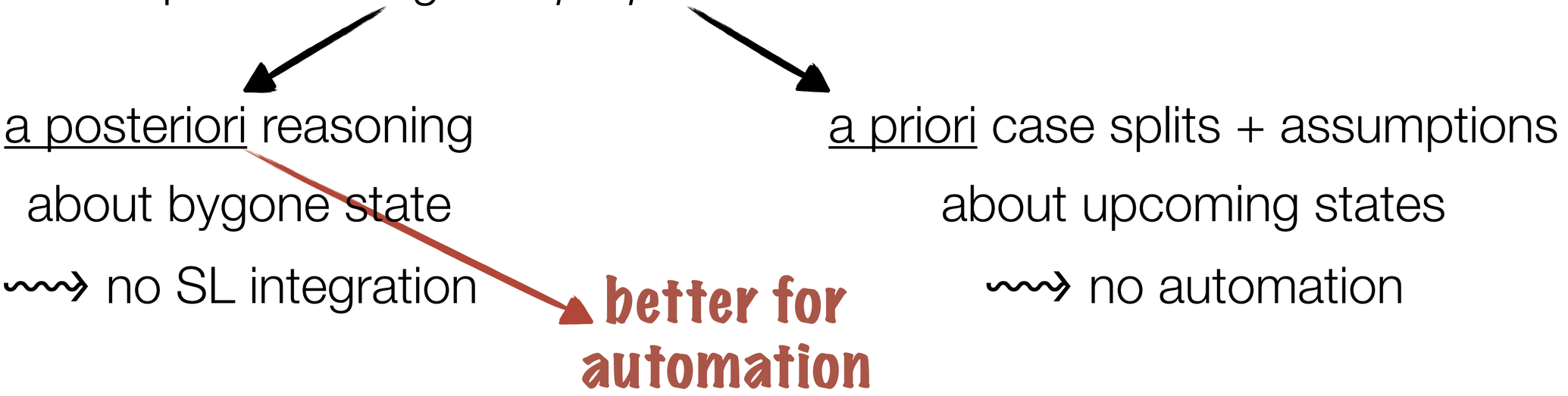
Linearizability

Problem: requires *hindsight* or *prophecies*



Linearizability

Problem: requires *hindsight* or *prophecies*



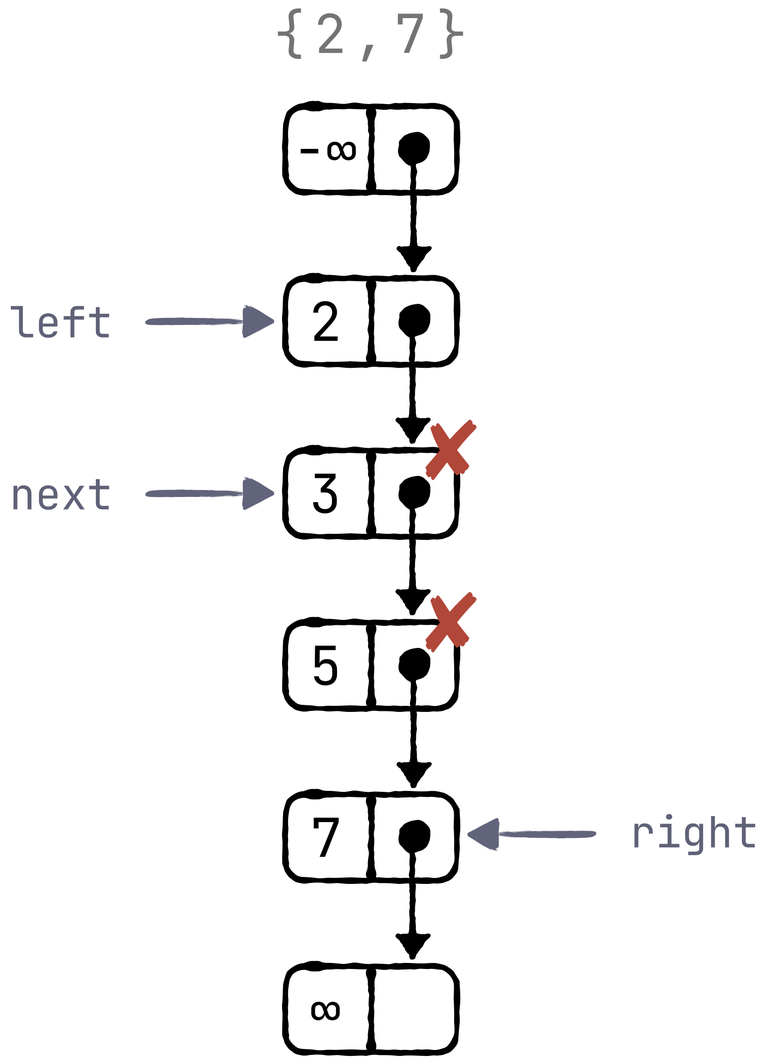
Contribution: *SL + Histories*

- lightweight lift of SL to computations
- elegant hindsight support (via logical variables)
- automated (in a tool)

Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k; // true
```

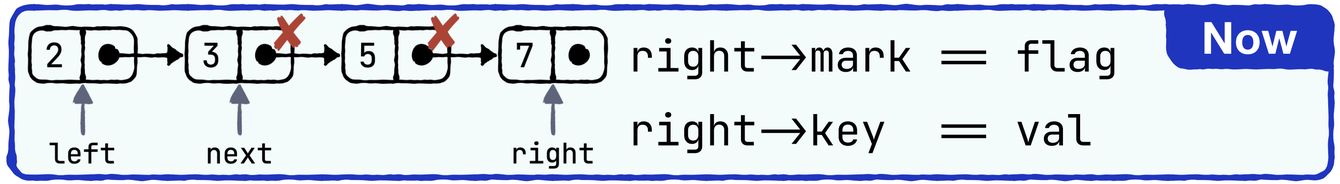
flag = false val = 7



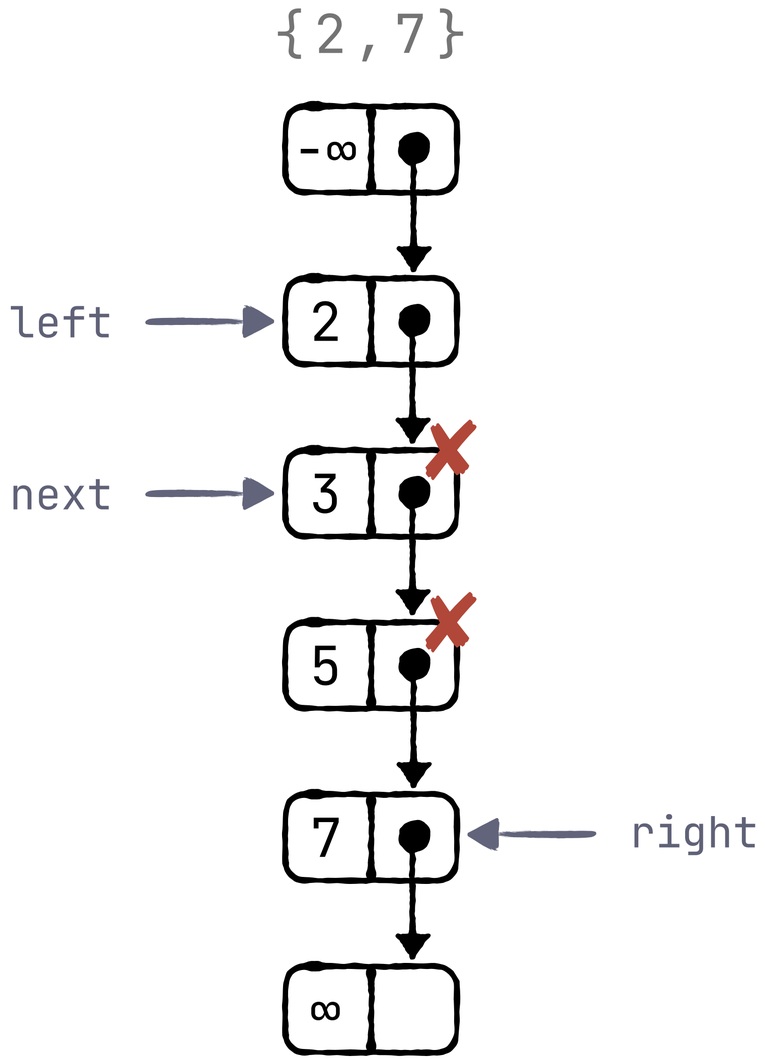
Harris' List: *contains*(7)

flag = false val = 7

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
```



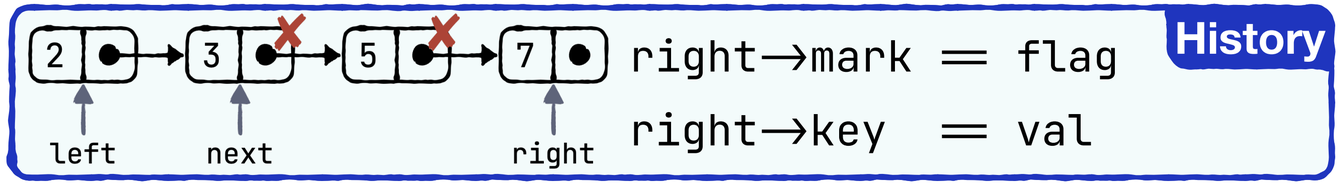

```
} while ( flag || val < k );
return val == k; // true
```



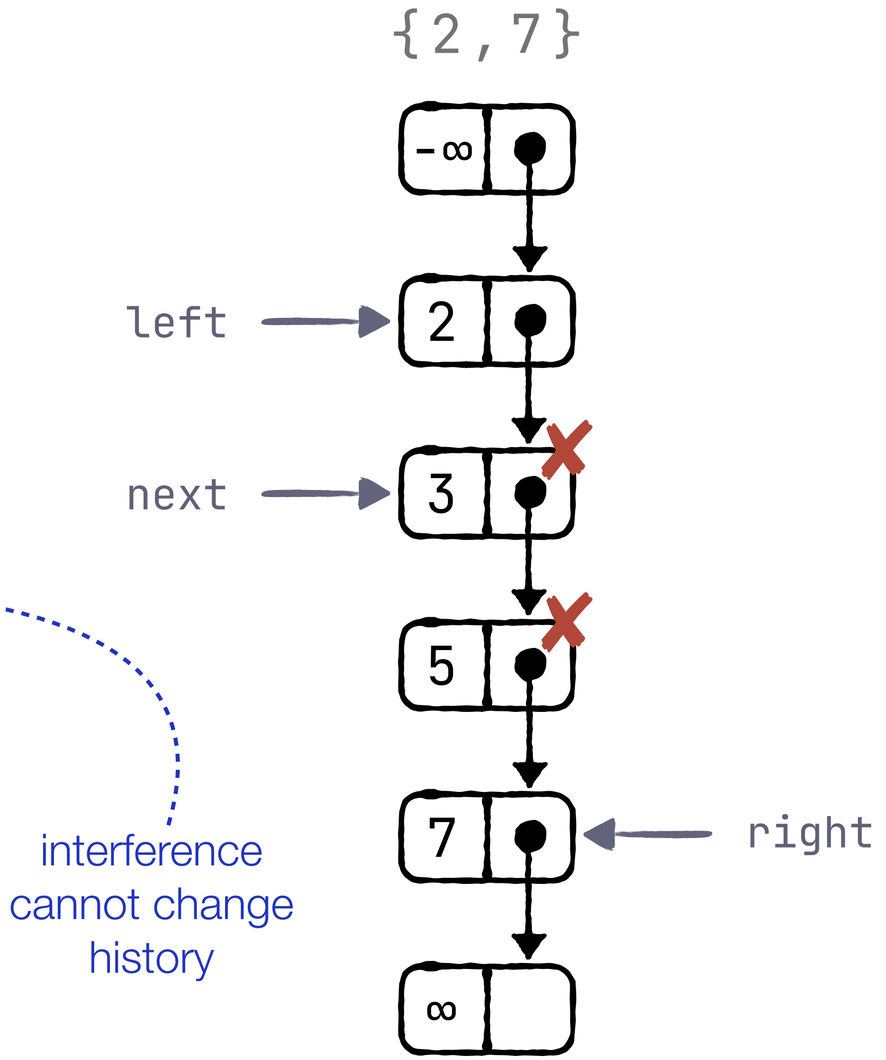
Harris' List: *contains*(7)

flag = false val = 7

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
```



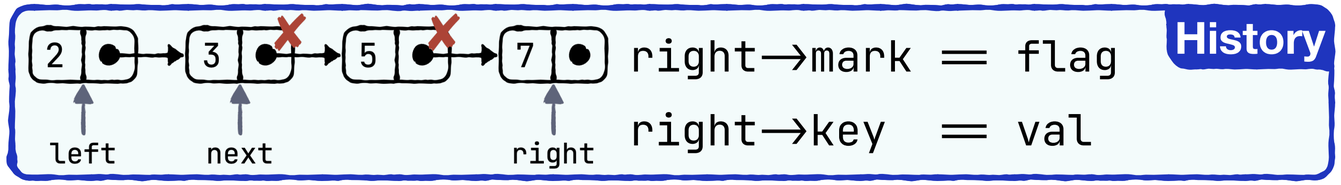

```
} while ( flag || val < k );
return val == k; // true
```



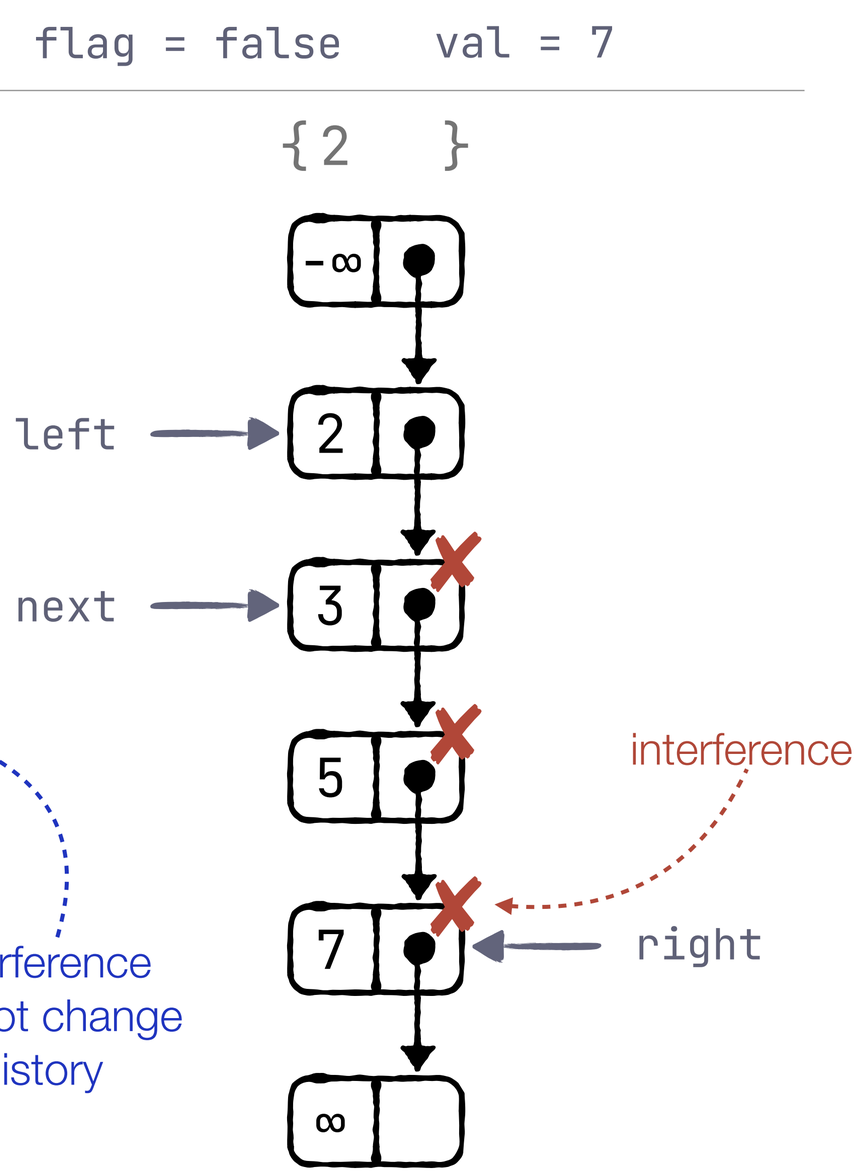
interference cannot change history

Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
```

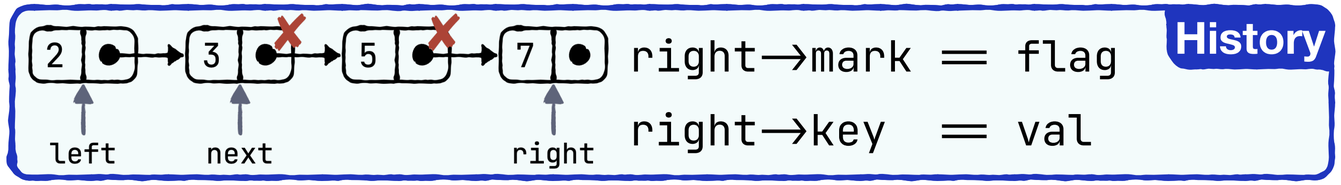



```
} while ( flag || val < k );
return val == k; // true
```



Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
```

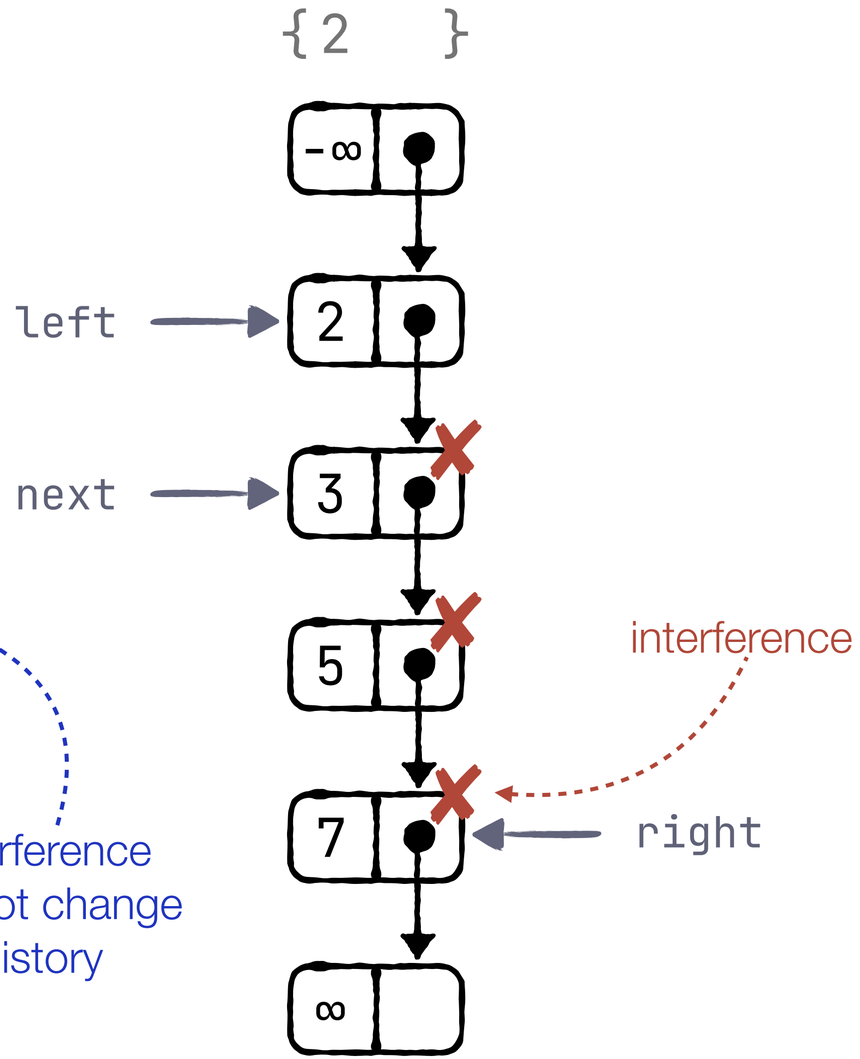



```
} while ( flag || val < k );
```

Now: `flag == false`

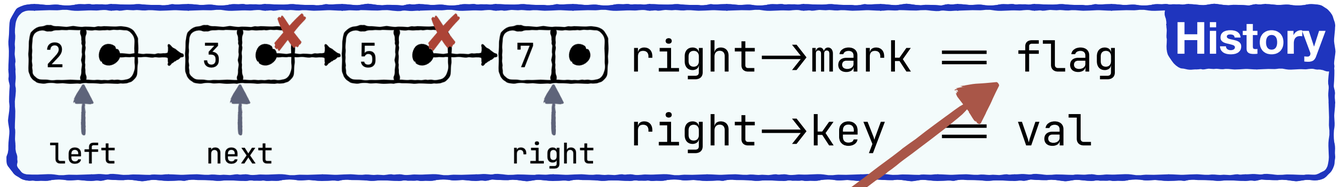
```
return val == k; // true
```

`flag = false` `val = 7`



Harris' List: *contains*(7)

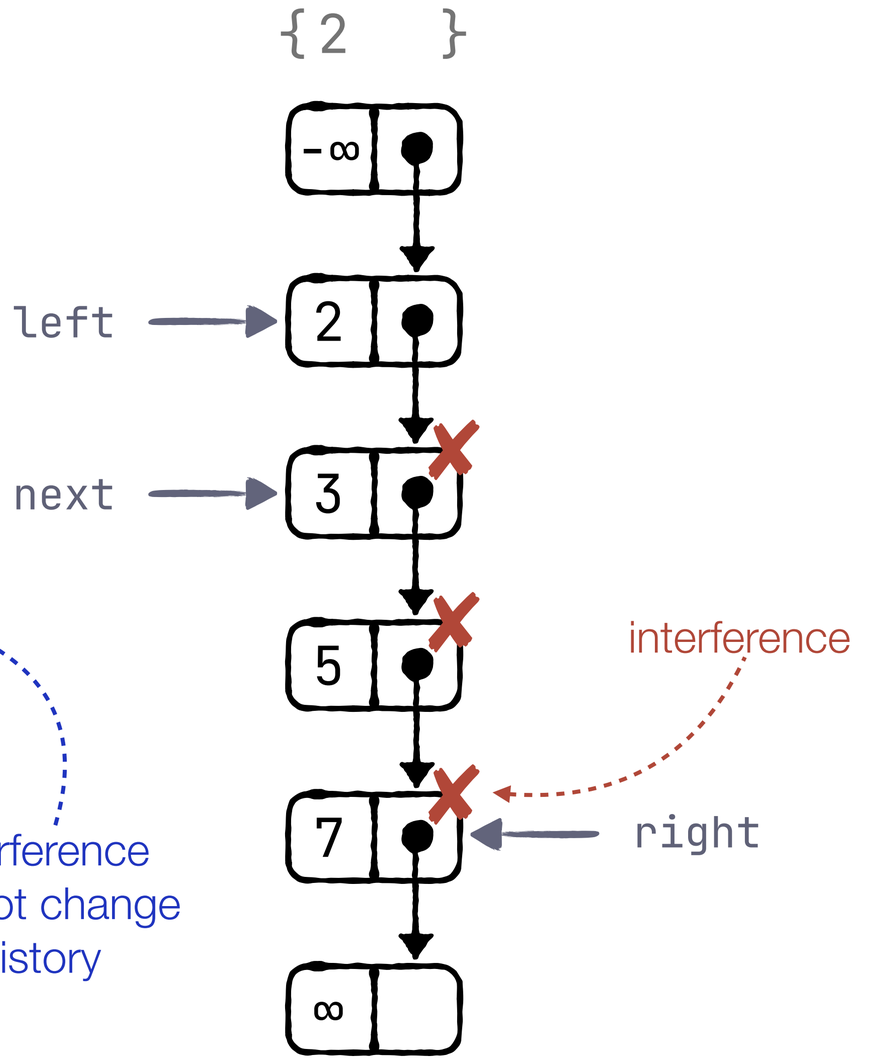
```
contains(k):

// ...
do {
    if (!mark) left, next := right, right→next;
    right := right→next;
    flag, val := right→mark, right→key;
```




```
} while ( flag || val < k );
```

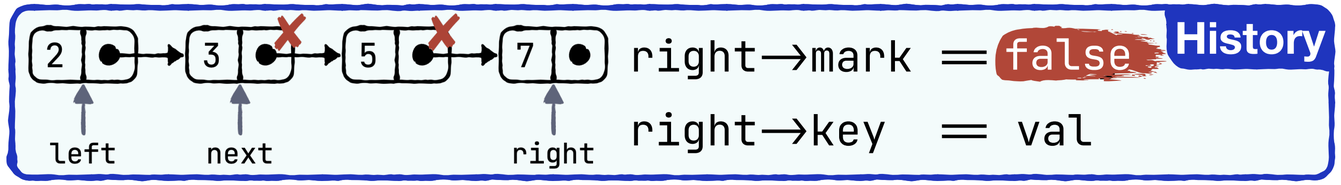

```
return val == k; // true
```

flag = false val = 7



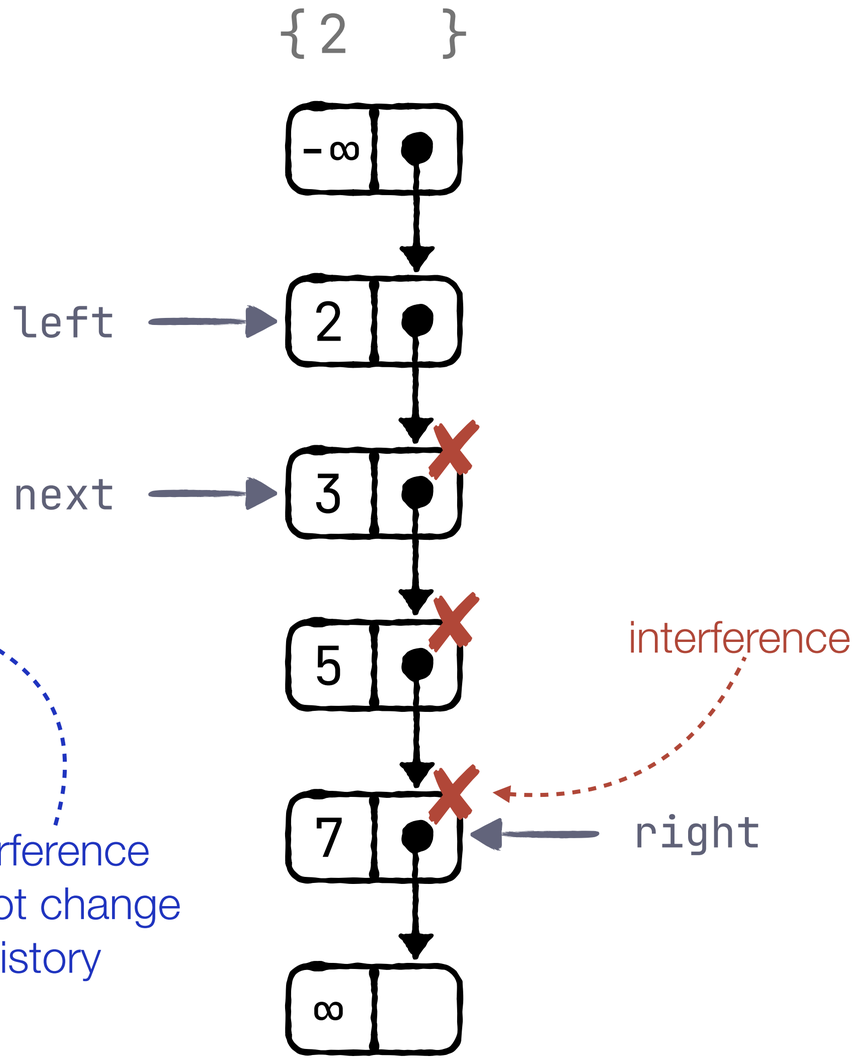
Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
```



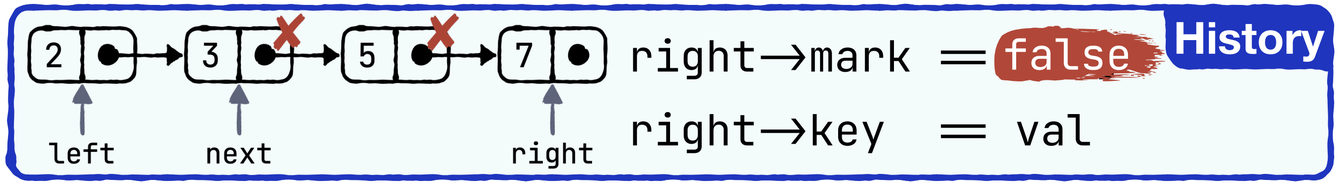

```
} while ( flag || val < k );
```


```
return val == k; // true
```



Harris' List: *contains*(7)

```
contains(k):

// ...
do {
    if (!mark) left, next := right, right→next;
    right := right→next;
    flag, val := right→mark, right→key;
```



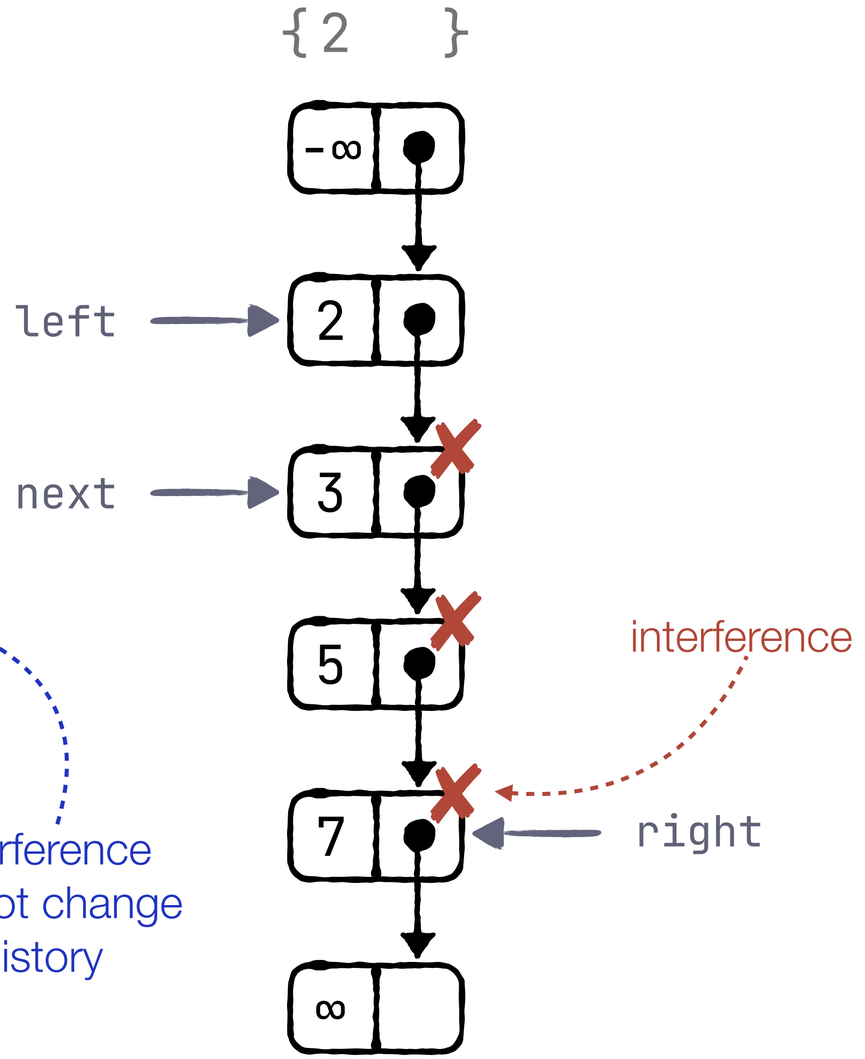

```
} while ( flag || val < k );
```

flag = false — **Now**

```
return val = k; // true
```

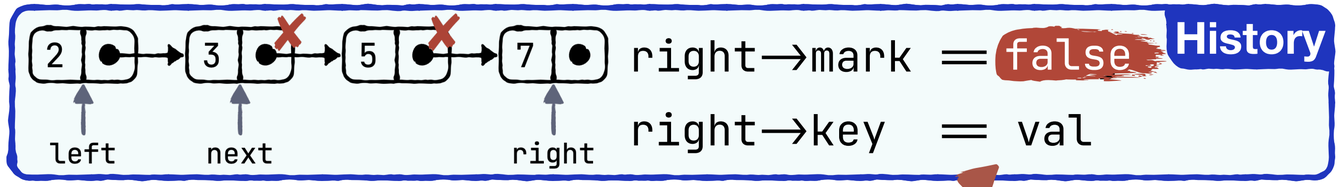
val = k — **Now**

flag = false val = 7



Harris' List: *contains*(7)

```
contains(k):

// ...
do {
    if (!mark) left, next := right, right→next;
    right := right→next;
    flag, val := right→mark, right→key;
```



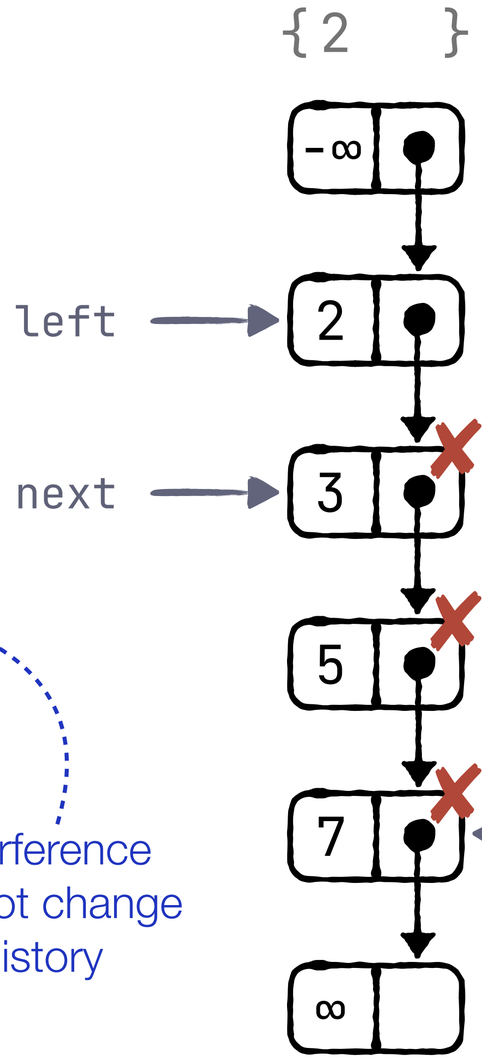

```
} while ( flag || val < k );
```

flag = false — **Now**

```
return val == k; // true
```

val = k — **Now**

flag = false val = 7

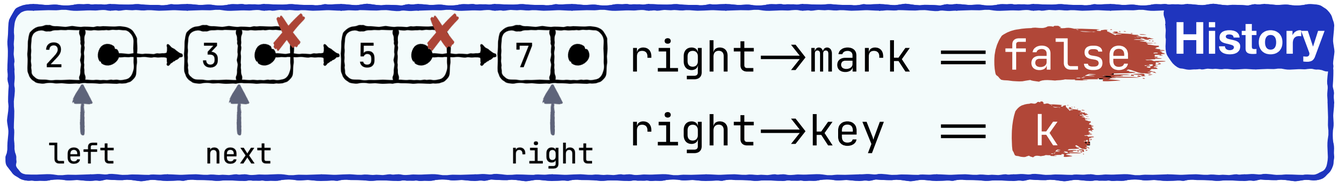


interference

interference cannot change history

Harris' List: *contains*(7)

```
contains(k):

// ...
do {
    if (!mark) left, next := right, right→next;
    right := right→next;
    flag, val := right→mark, right→key;
```




```
} while ( flag || val < k );
```

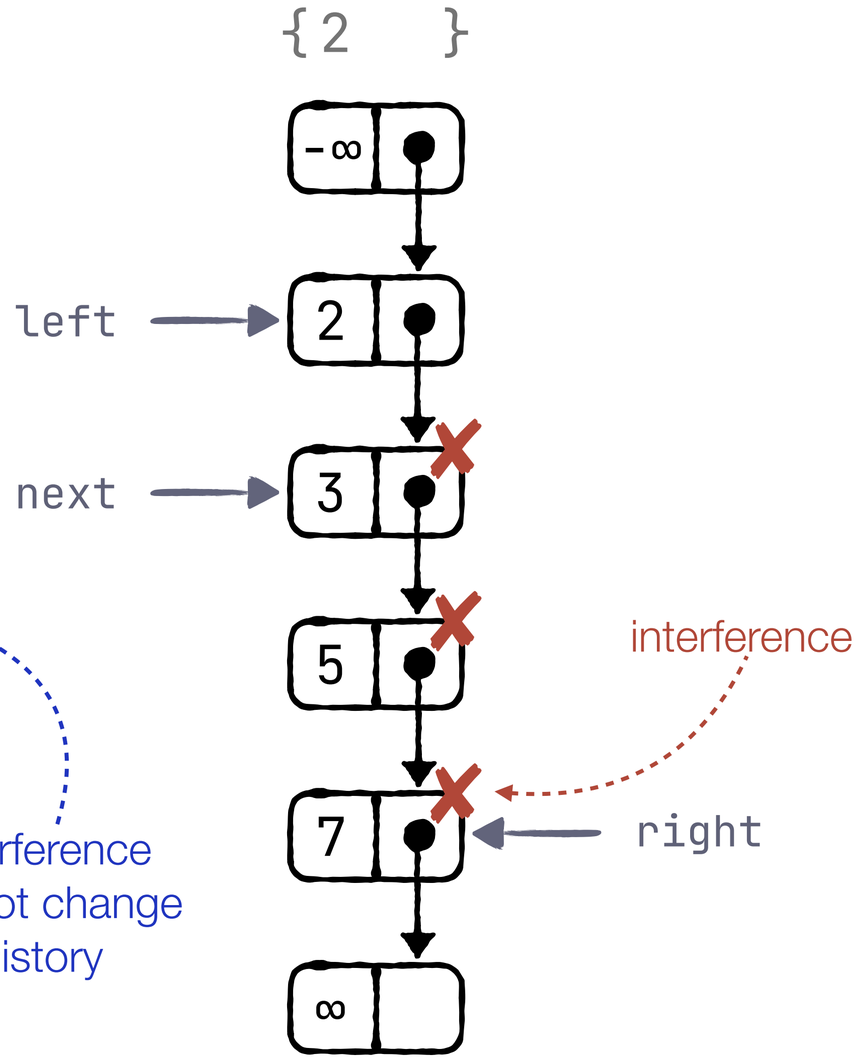
flag == false **Now**

```
return val == k; // true
```

val == k **Now**

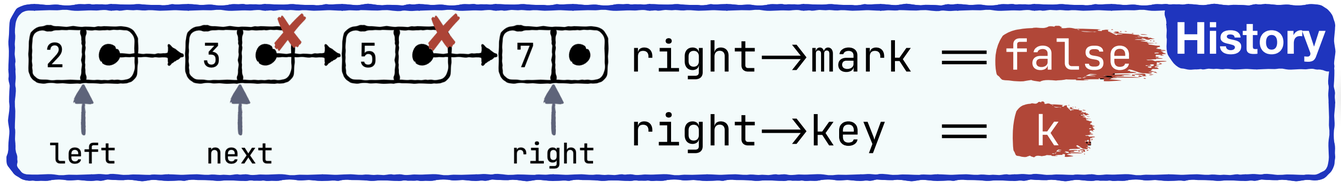
interference cannot change history

flag = false val = 7



Harris' List: *contains*(7)

```
contains(k):

// ...
do {
    if (!mark) left, next := right, right→next;
    right := right→next;
    flag, val := right→mark, right→key;
```




```
} while ( flag || val < k );
```

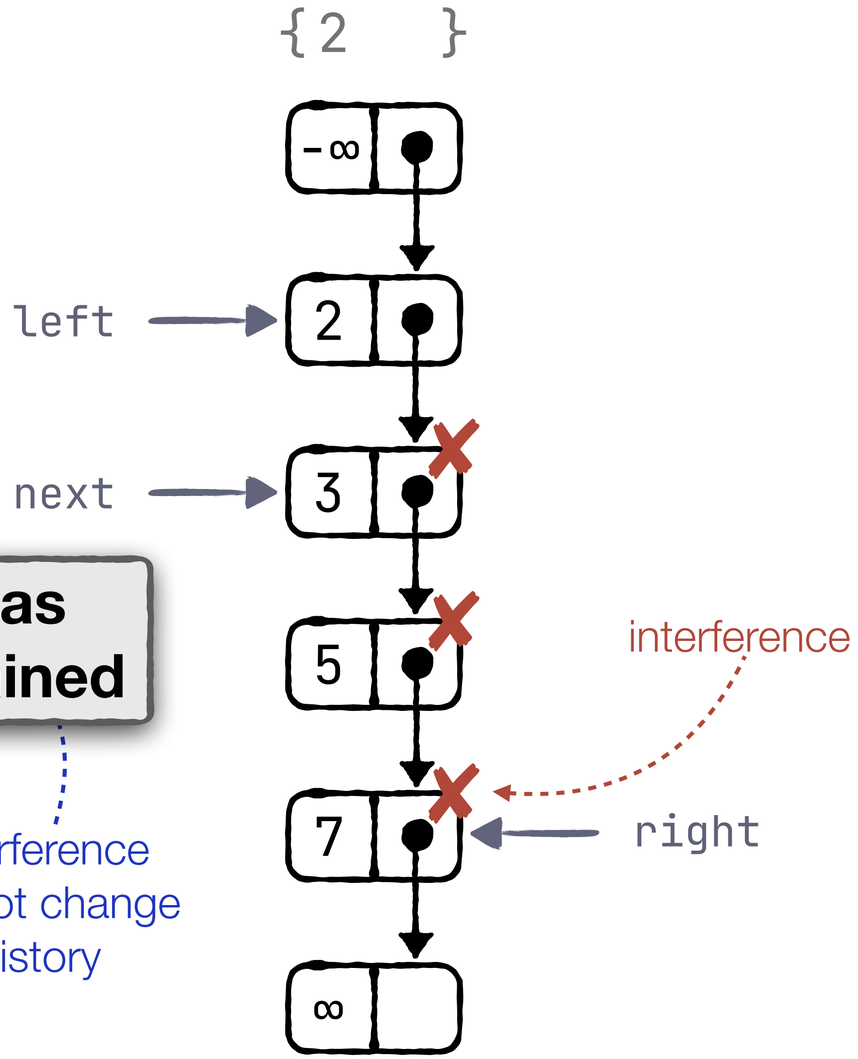
flag == false **Now**

```
return val == k; // true
```

val == k **Now**

k was contained

interference cannot change history



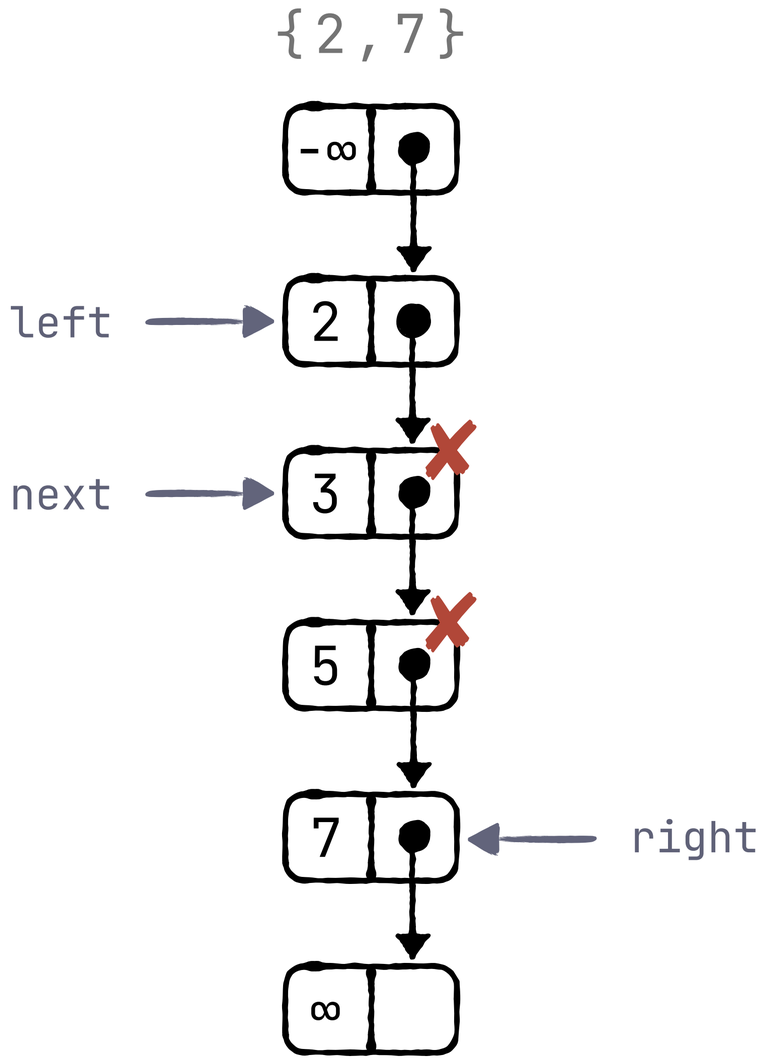
Verification Challenge 1: **Linearizability**

Verification Challenge 2: **Complex Unbounded Updates**

Harris' List: *contains*(7)

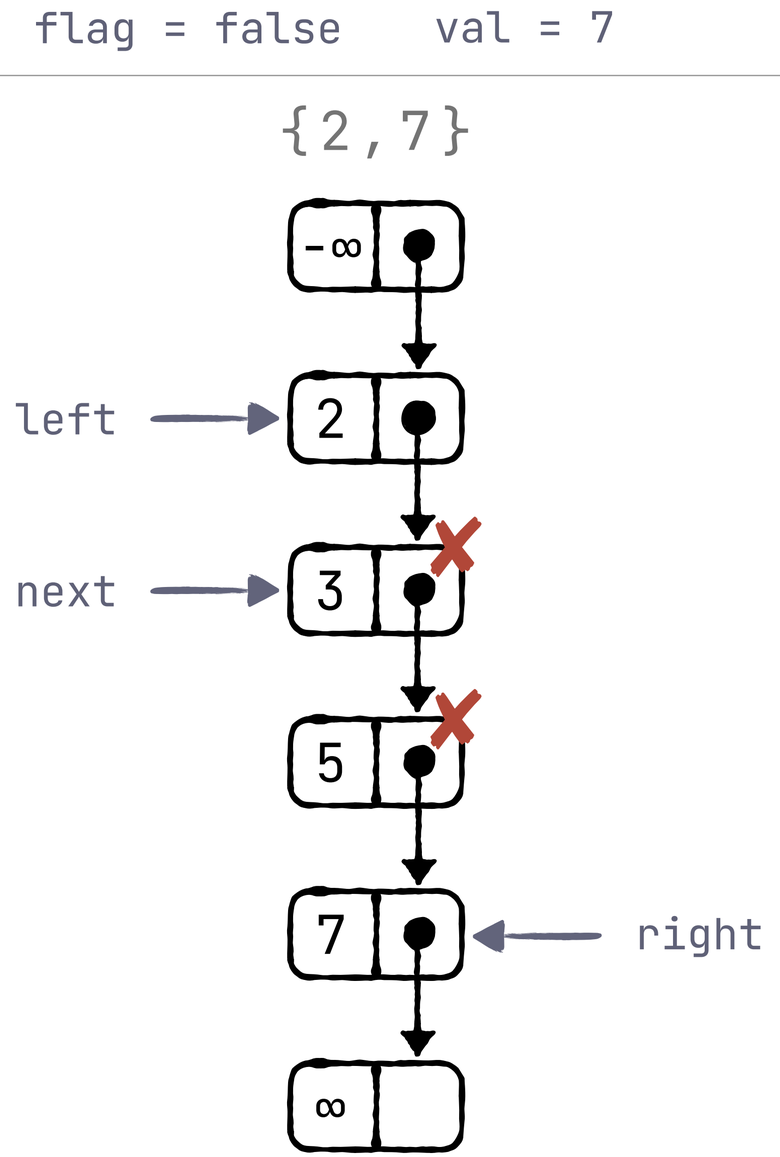
```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
return val == k; // true
```

flag = false val = 7



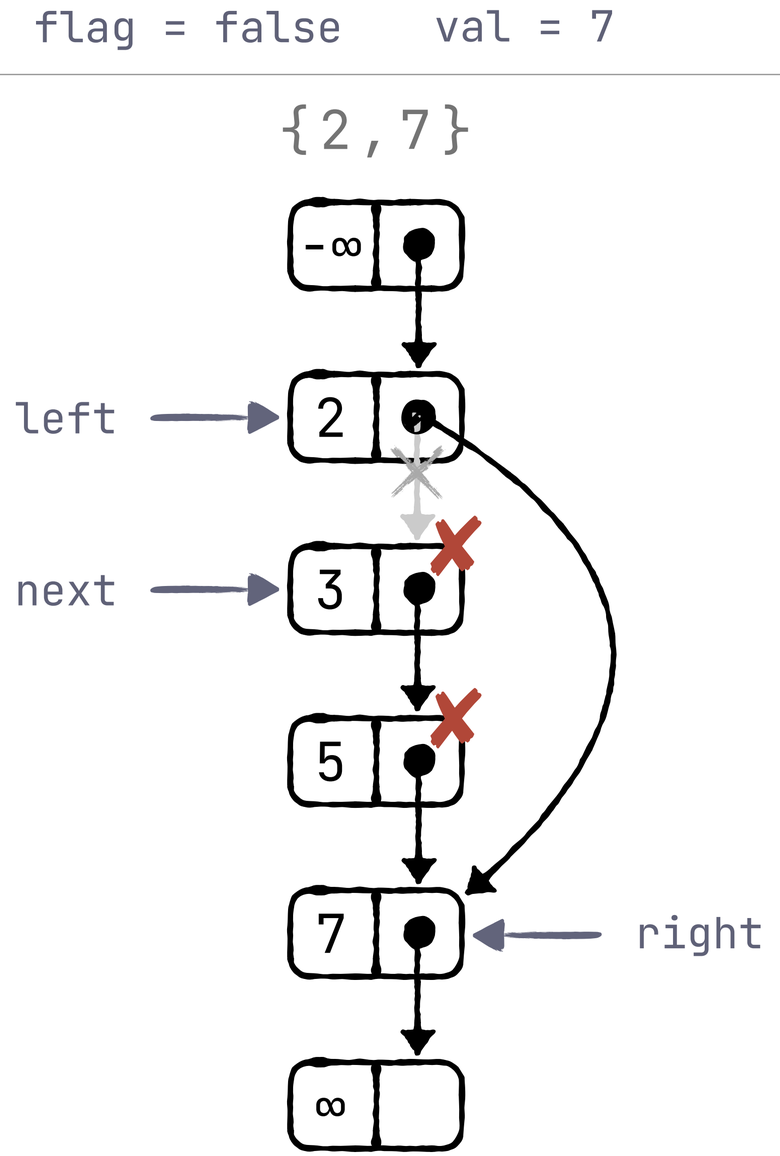
Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
if ( left→next == next && !left→mark )
   left→next := right;
return val == k; // true
```



Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
if ( left→next == next && !left→mark )
   left→next := right;
return val == k; // true
```



Complex Unbounded Updates

Problem: require

- *shape discovery* during traversal

  ⇝ e.g. inductive SL predicates

- *induction* over discovered shape

  ⇝ no automation

Complex Unbounded Updates

Problem: require

- *shape discovery* during traversal

  ⇝ e.g. inductive SL predicates

- *induction* over discovered shape

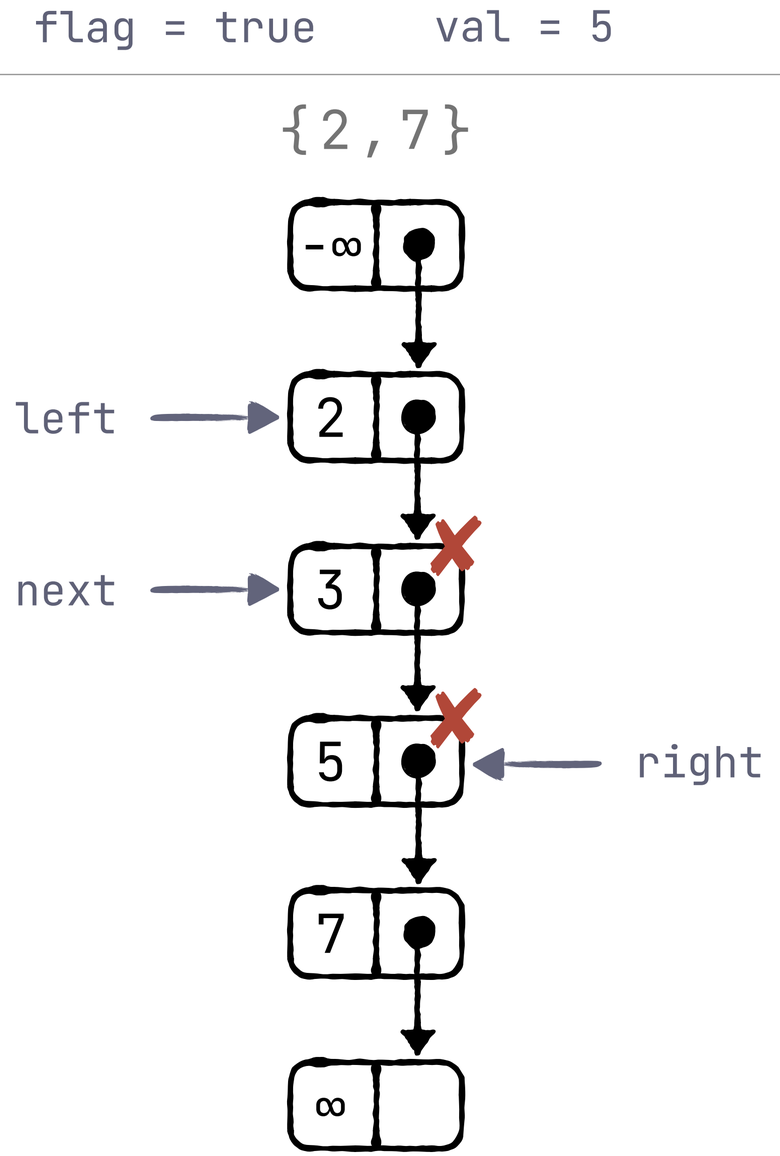
  ⇝ no automation

Contribution: *SL + Futures*

- compose complex updates from simpler ones
- discover&compose simpler updates during traversal
- automated (in a tool)

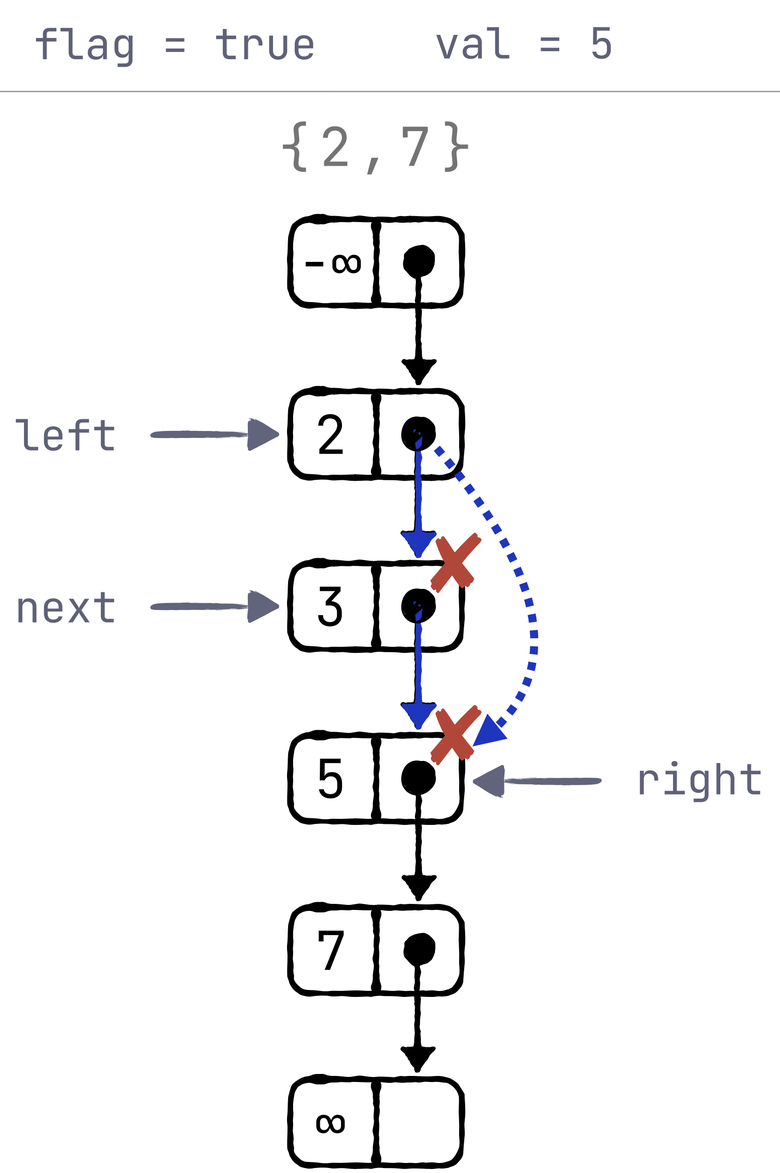
Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
if ( left→next == next && !left→mark )
   left→next := right;
return val == k;
```



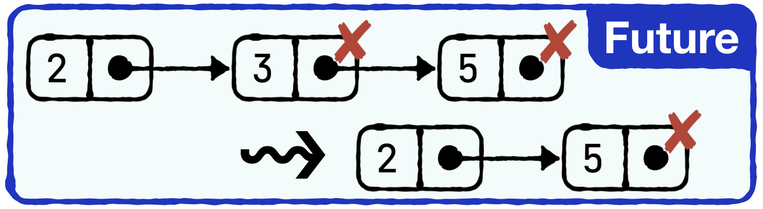
Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
} while ( flag || val < k );
if ( left→next == next && !left→mark )
   left→next := right;
return val == k;
```



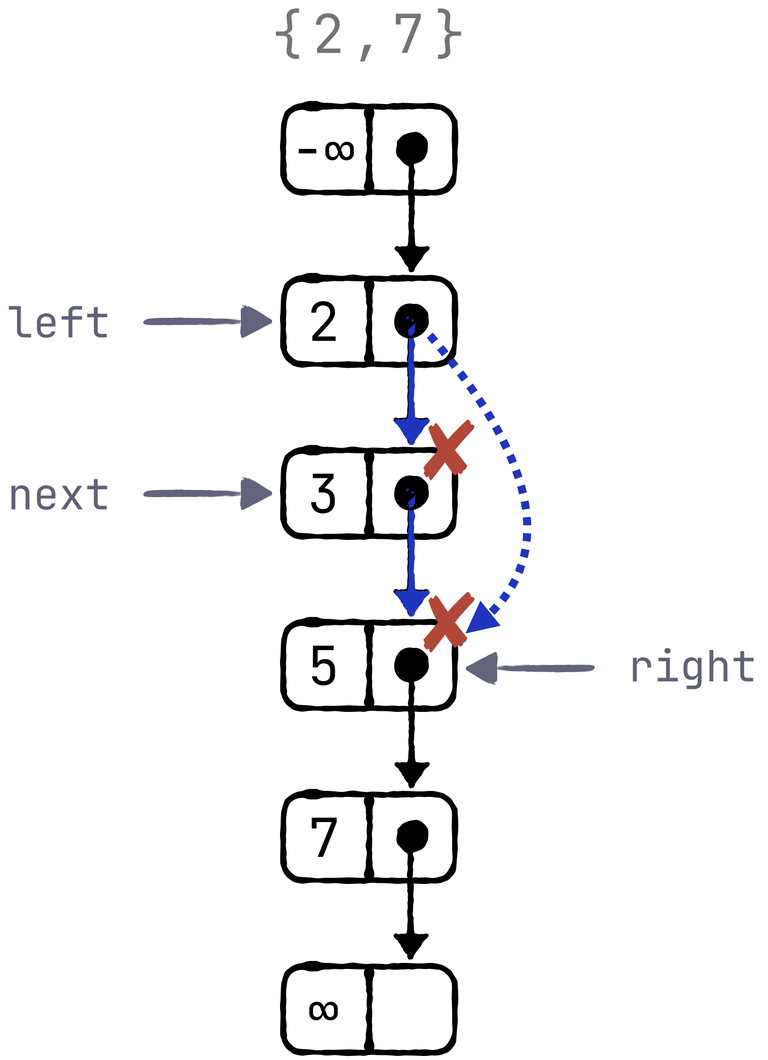
Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
```



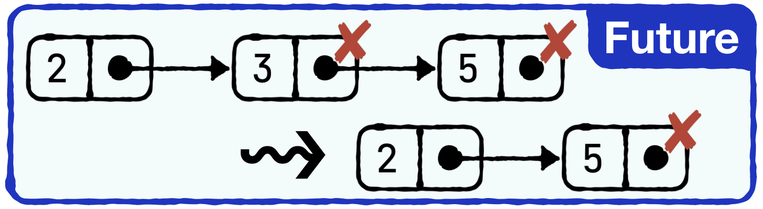

```
} while ( flag || val < k );
if ( left→next == next && !left→mark )
   left→next := right;
```

flag = true val = 5



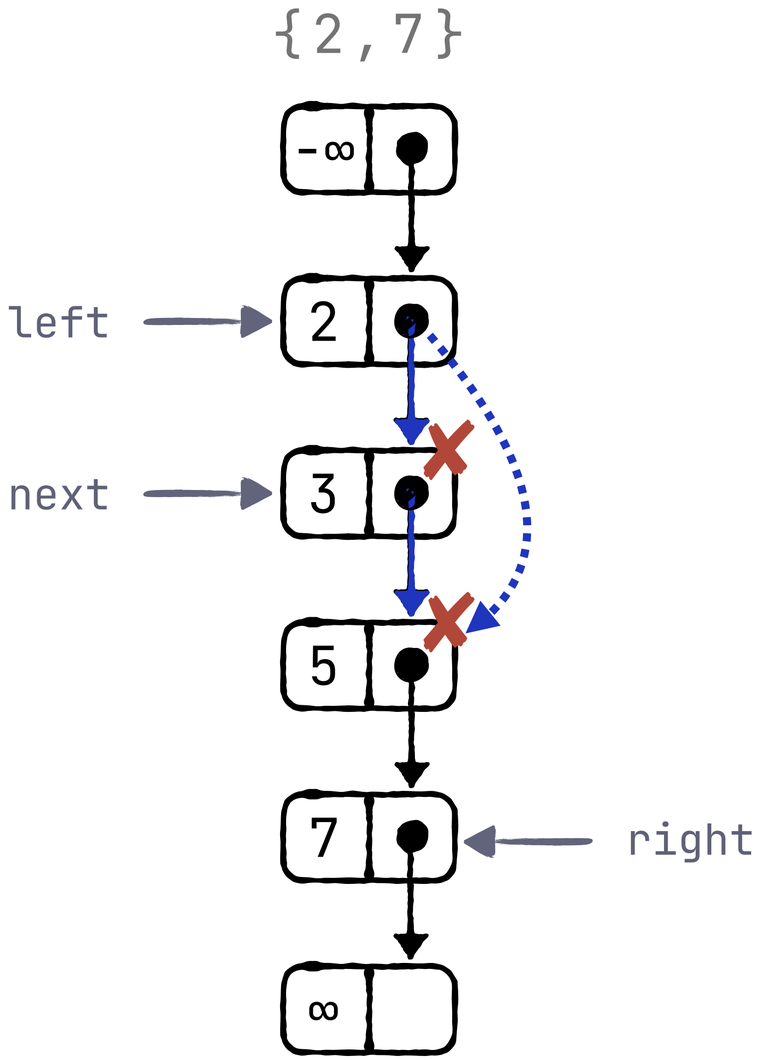
Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
```



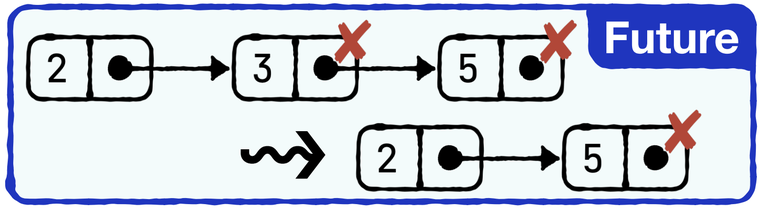

```
} while ( flag || val < k );
if ( left→next == next && !left→mark )
   left→next := right;
```

flag = false val = 7



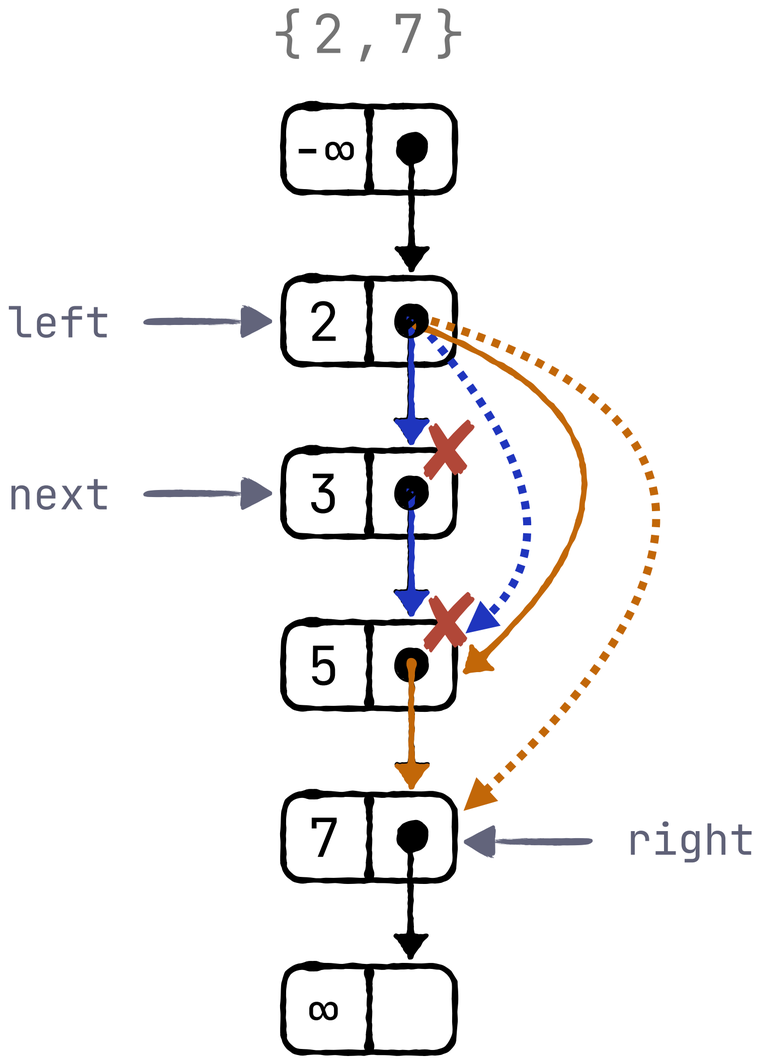
Harris' List: *contains*(7)

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
```



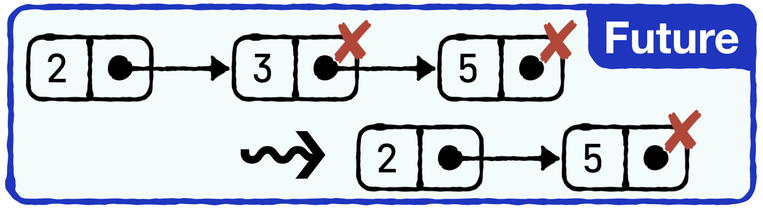

```
} while ( flag || val < k );
if ( left→next == next && !left→mark )
   left→next := right;
```

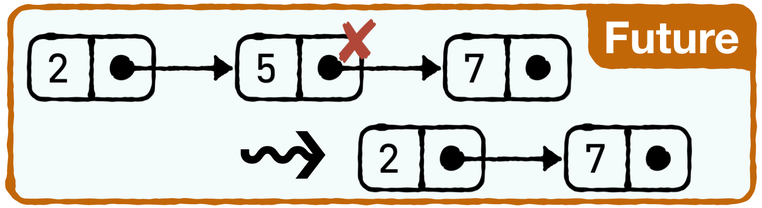
flag = false val = 7



Harris' List: *contains*(7)

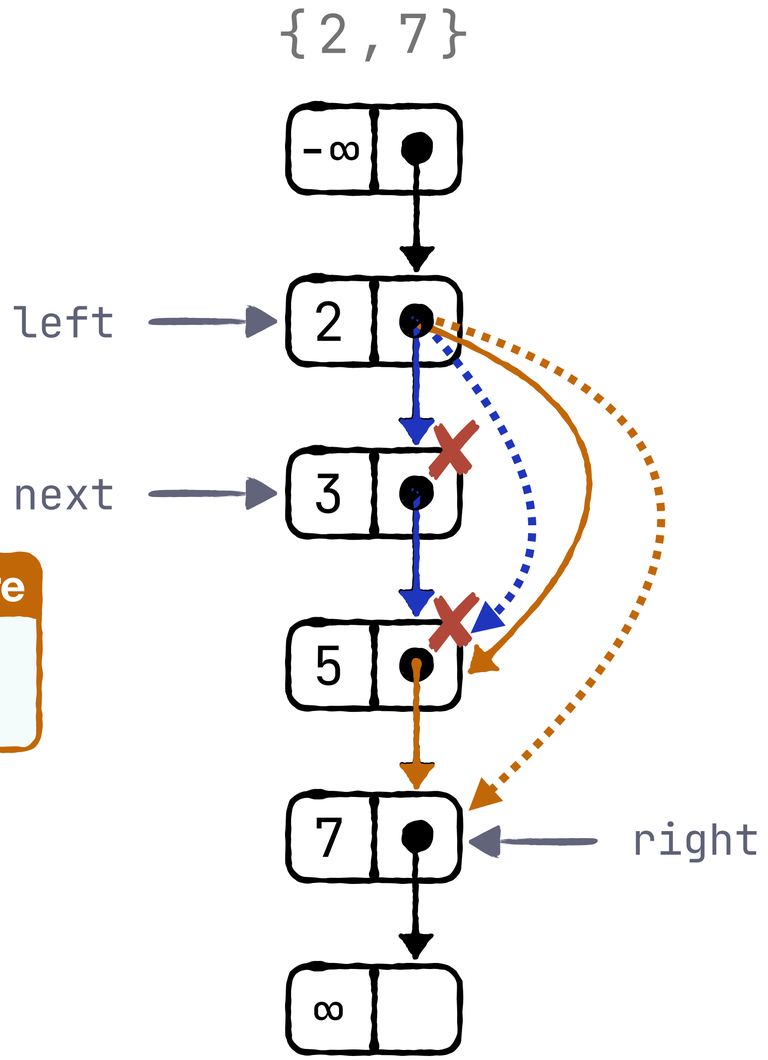
```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
```






```
} while ( flag || val < k );
if ( left→next == next && !left→mark )
   left→next := right;
```

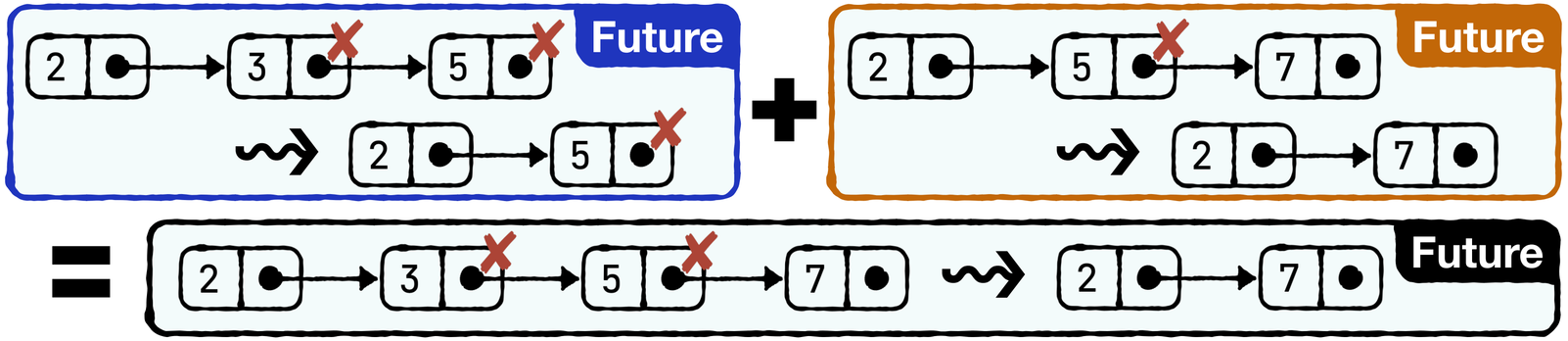
flag = false val = 7



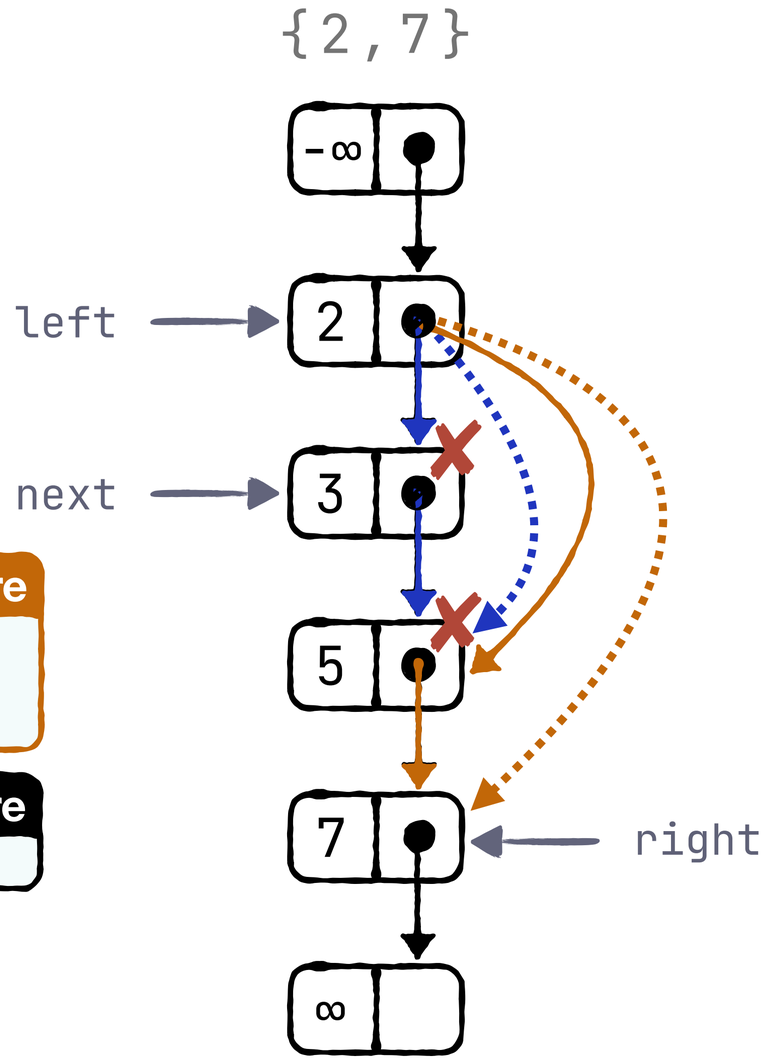
Harris' List: *contains*(7)

`flag = false  val = 7`

```
contains(k):

// ...
do {
   if (!mark) left, next := right, right→next;
   right := right→next;
   flag, val := right→mark, right→key;
```

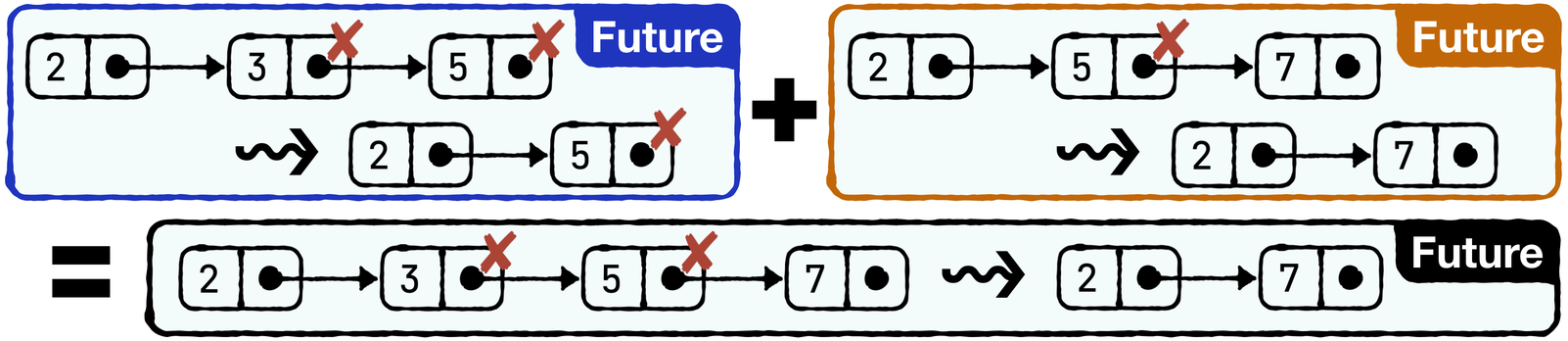



```
} while ( flag || val < k );
if ( left→next == next && !left→mark )
   left→next := right;
```



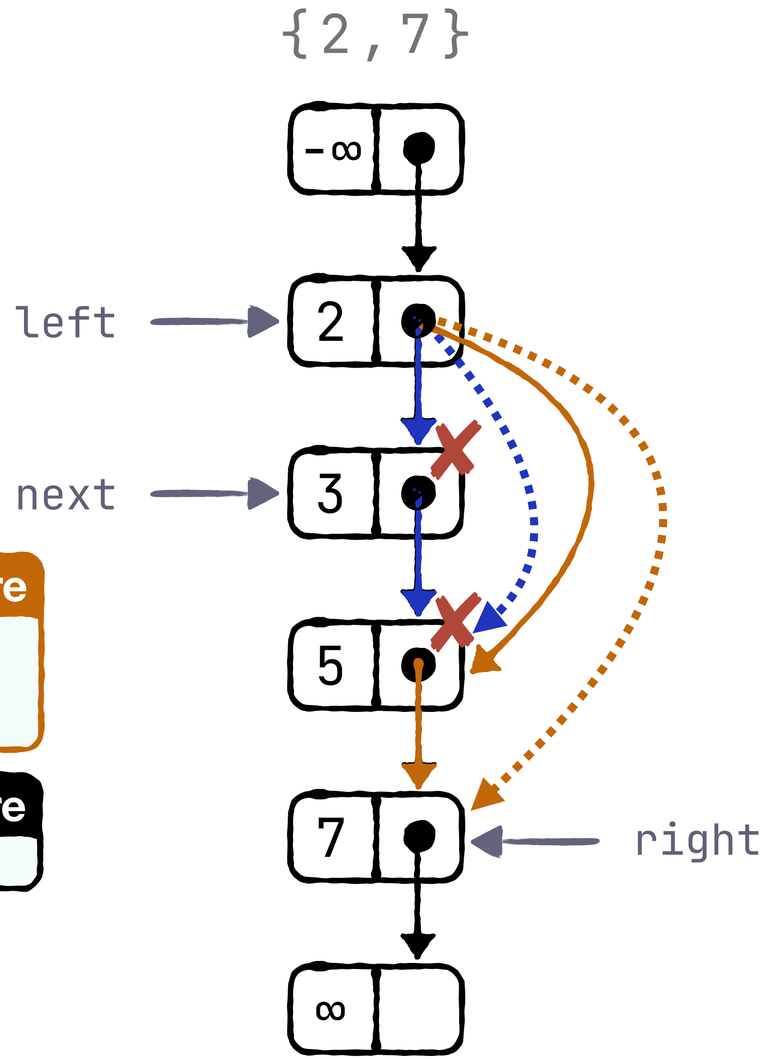
Harris' List: *contains*(7)

```
contains(k):

// ...
do {
    if (!mark) left, next := right, right→next;
    right := right→next;
    flag, val := right→mark, right→key;
```




```
} while ( flag || val < k );

if ( left→next == next && !left→mark )
    left→next := right;
```

flag = false val = 7



Recap:
A Concurrent Program Logic with a Future and History

- program logic with
  - history ⇝ hindsight reasoning in SL
  - future ⇝ complex unbounded updates via simple ones
  - tool ⇝ first to automate general hindsight reasoning
    first to verify challenging examples like Harris' list

Benchmark	#Iter	$\#\mathbb{I}_{lfp}$	#Cand	Com.	Fut.	Hist.	Join	Inter.	Lineariz.
Fine-Grained set	2	5	2	11%	15%	43%	15%	8%	46*s* ✓
Lazy set	2	6	2	10%	13%	54%	11%	5%	77*s* ✓
FEMRS tree (no maintenance)	2	5	2	19%	0%	49%	1%	9%	130*s* ✓
Vechev&Yahav 2CAS set	2	3	1	14%	0%	33%	31%	9%	125*s* ✓
Vechev&Yahav CAS set	2	4	1	15%	7%	39%	23%	6%	54*s* ✓
ORVYY set	2	3	0	17%	0%	40%	26%	6%	47*s* ✓
Michael set	2	4	2	11%	29%	30%	15%	6%	306*s* ✓
Michael set (wait-free search)	2	4	2	11%	28%	30%	15%	6%	246*s* ✓
Harris set	2	4	2	7%	8%	19%	32%	4%	1378*s* ✓
Harris set (wait-free search)	2	4	2	8%	10%	17%	34%	3%	1066*s* ✓

Recap:
A Concurrent Program Logic with a Future and History

Thanks

- program logic with
  - history ⇝ hindsight reasoning in SL
  - future ⇝ complex unbounded updates via simple ones
  - tool ⇝ first to automate general hindsight reasoning
    first to verify challenging examples like Harris' list

Benchmark	#Iter	$\#\mathbb{I}_{lfp}$	#Cand	Com.	Fut.	Hist.	Join	Inter.	Lineariz.
Fine-Grained set	2	5	2	11%	15%	43%	15%	8%	46*s* ✓
Lazy set	2	6	2	10%	13%	54%	11%	5%	77*s* ✓
FEMRS tree (no maintenance)	2	5	2	19%	0%	49%	1%	9%	130*s* ✓
Vechev&Yahav 2CAS set	2	3	1	14%	0%	33%	31%	9%	125*s* ✓
Vechev&Yahav CAS set	2	4	1	15%	7%	39%	23%	6%	54*s* ✓
ORVYY set	2	3	0	17%	0%	40%	26%	6%	47*s* ✓
Michael set	2	4	2	11%	29%	30%	15%	6%	306*s* ✓
Michael set (wait-free search)	2	4	2	11%	28%	30%	15%	6%	246*s* ✓
Harris set	2	4	2	7%	8%	19%	32%	4%	1378*s* ✓
Harris set (wait-free search)	2	4	2	8%	10%	17%	34%	3%	1066*s* ✓