Embedding Hindsight Reasoning in Separation Logic

Roland Meyer, Thomas Wies, Sebastian Wolff

[PLDI'23]

Motivation

"*Programs = Algorithms + Data Structures*"

–Niklaus Wirth, 1978

Motivation

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

–Niklaus Wirth, 1978

$\downarrow$

Verification
via Separation Logic (SL)

Motivation

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

–Niklaus Wirth, 1978

$\downarrow$

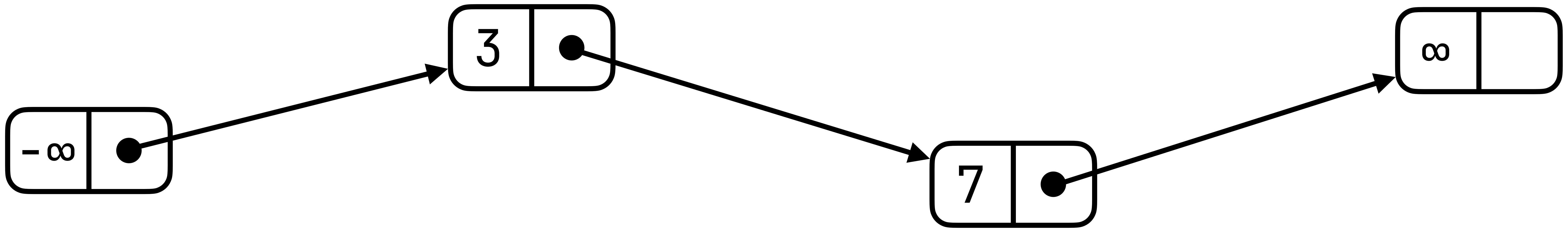
Verification via Separation Logic (SL)

$\downarrow$

Automation

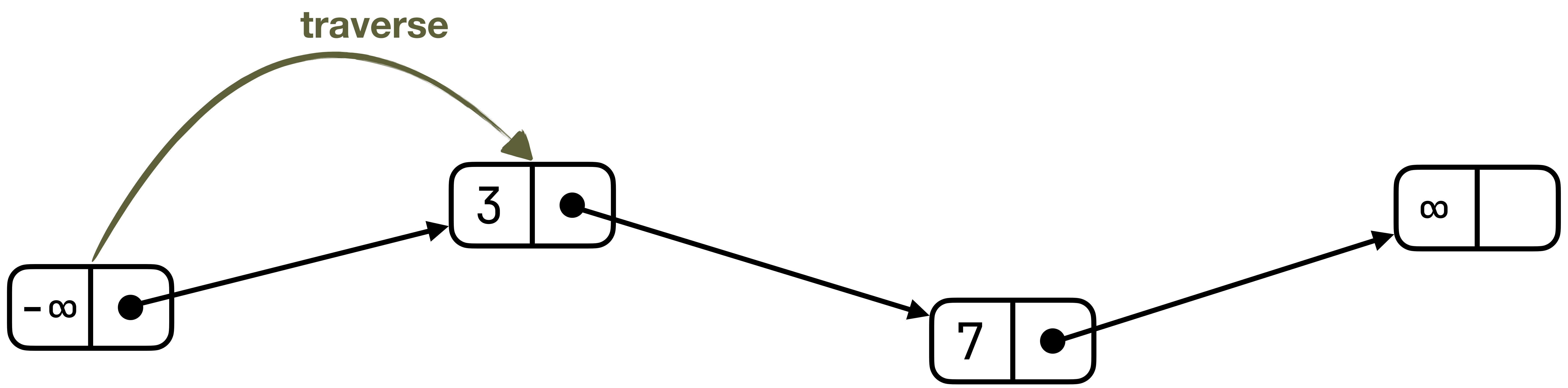
Example: `contains(5)`

```
sorted({3,7}):
```



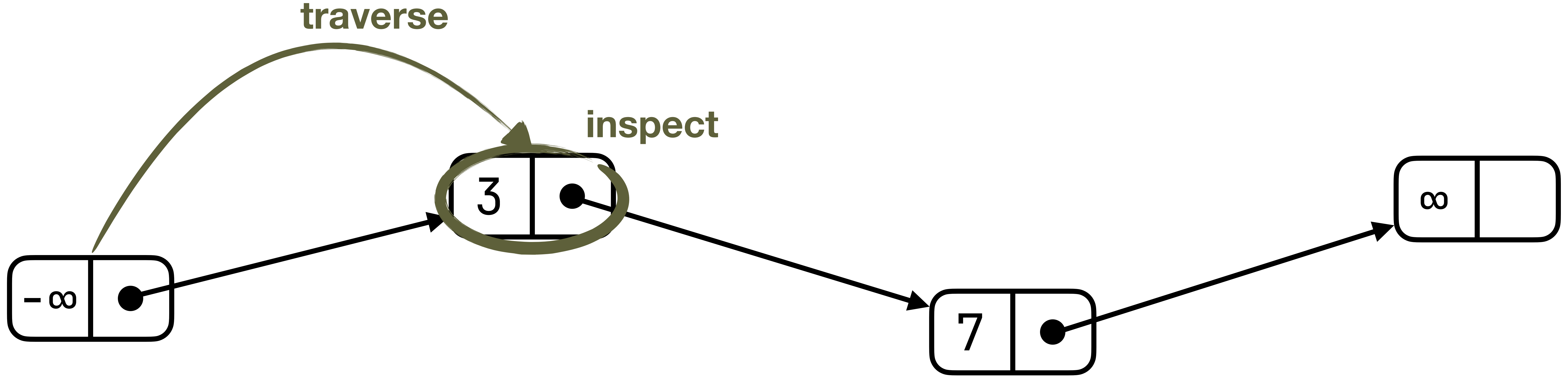
Example: `contains(5)`

```
sorted({3,7}):
```



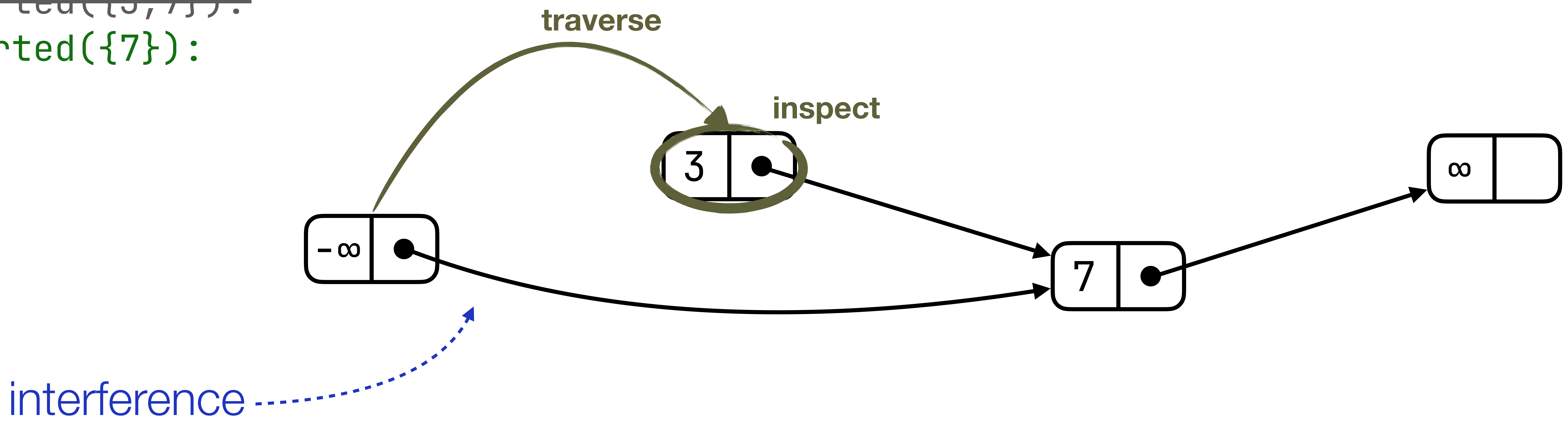
Example: `contains(5)`

`sorted({3,7}):`



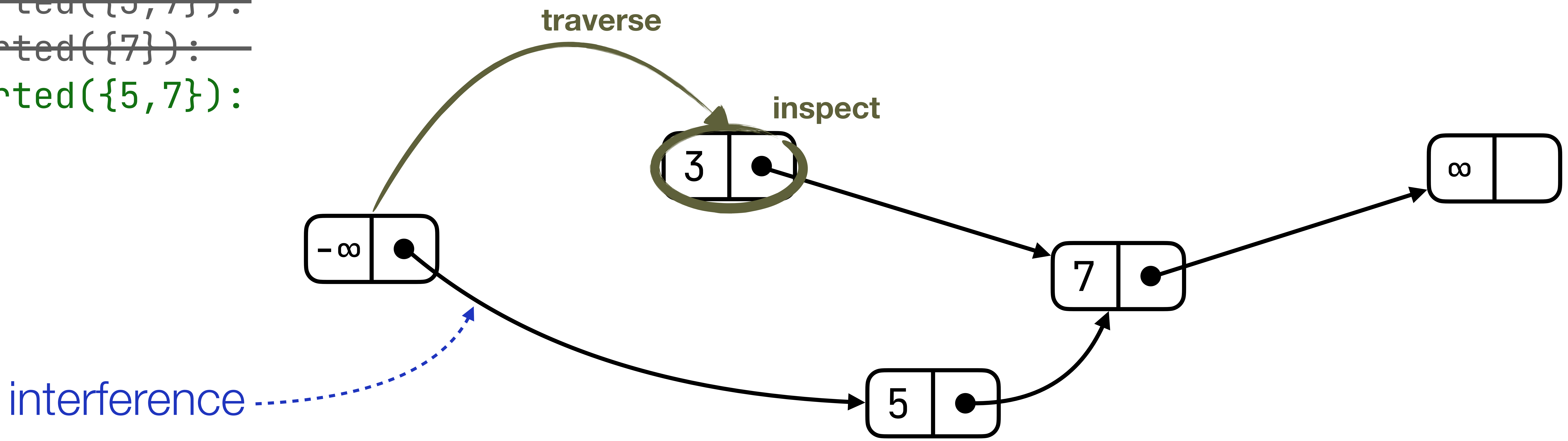
Example: `contains(5)`

~~`sorted({3,7}):`~~
`sorted({7}):`



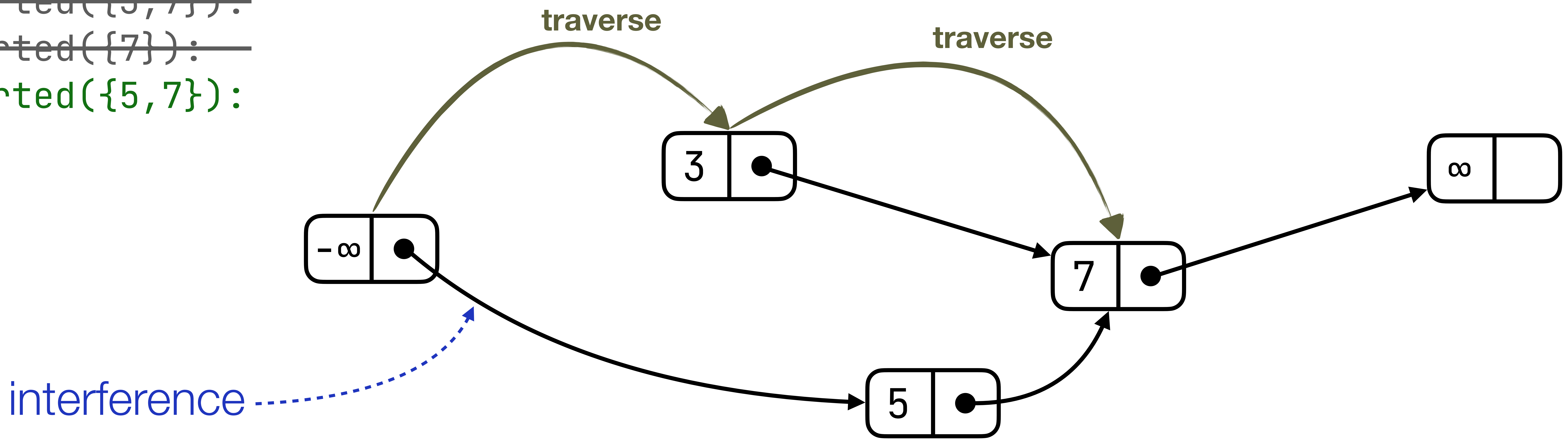
Example: `contains(5)`

```
~~sorted({3,7}):~~
~~sorted({7}):~~
sorted({5,7}):
```



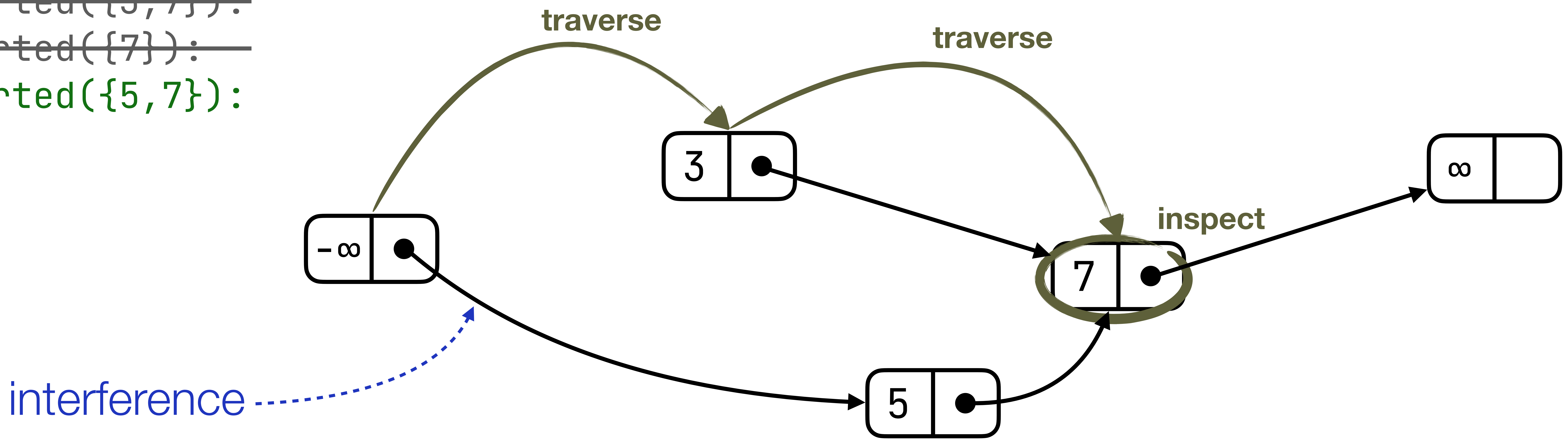
Example: `contains(5)`

~~sorted({3,7}):~~
~~sorted({7}):~~
sorted({5,7}):



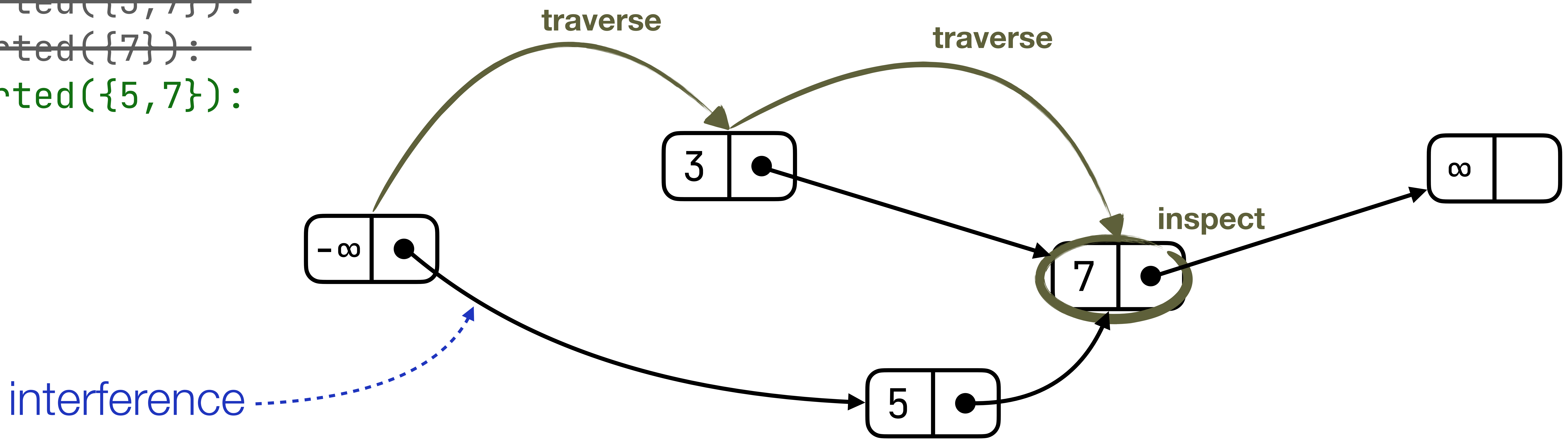
Example: `contains(5)`

```
~~sorted({3,7}):~~
~~sorted({7}):~~
sorted({5,7}):
```



Example: `contains(5)`

~~`sorted({3,7}):`~~
~~`sorted({7}):`~~
`sorted({5,7}):`



- Result: `contains(5) = false`
  - ➡ linearizable (i.e. correct); proof obligation: `traversal` ⊨ `sorted(M), 5∉M` *`at some point`*

Non-fixed Linearization Points

State of the art:

Non-fixed Linearization Points

State of the art:

- prophecies
  - ➡ predict interference upfront to "fix" linearization point

eager case analysis

Non-fixed Linearization Points

State of the art:

- prophecies
  - ➡ predict interference upfront to "fix" linearization point
- custom resource algebras
  - ➡ capture structural invariant + pass proof obligations

eager case analysis

Non-fixed Linearization Points

State of the art:

- prophecies
  - ➡ predict interference upfront to "fix" linearization point
- custom resource algebras
  - ➡ capture structural invariant + pass proof obligations

$\Longrightarrow$ complex, cumbersome proofs

eager case analysis

hard to automate

Non-fixed Linearization Points

State of the art:

- prophecies
  - ➡ predict interference upfront to "fix" linearization point
- custom resource algebras
  - ➡ capture structural invariant + pass proof obligations

$\Longrightarrow$ complex, cumbersome proofs

eager case analysis

hard to automate

Contribution: **hindsight reasoning** in SL

Hindsight Reasoning

- Record execution history P Past

  not just current state Q Now

Hindsight Reasoning

- Record execution history $\boxed{P \quad \textbf{Past}}$

  not just current state $\boxed{Q \quad \textbf{Now}}$

- Derive *inevitable* facts:

  $$\boxed{P \quad \textbf{Past}} \wedge \boxed{Q \quad \textbf{Now}} \leadsto \boxed{R \quad \textbf{Past}}$$

  if in the state transition system

  all paths from P to Q pass through R

Hindsight Reasoning

- Record execution history $\boxed{P\ \text{Past}}$

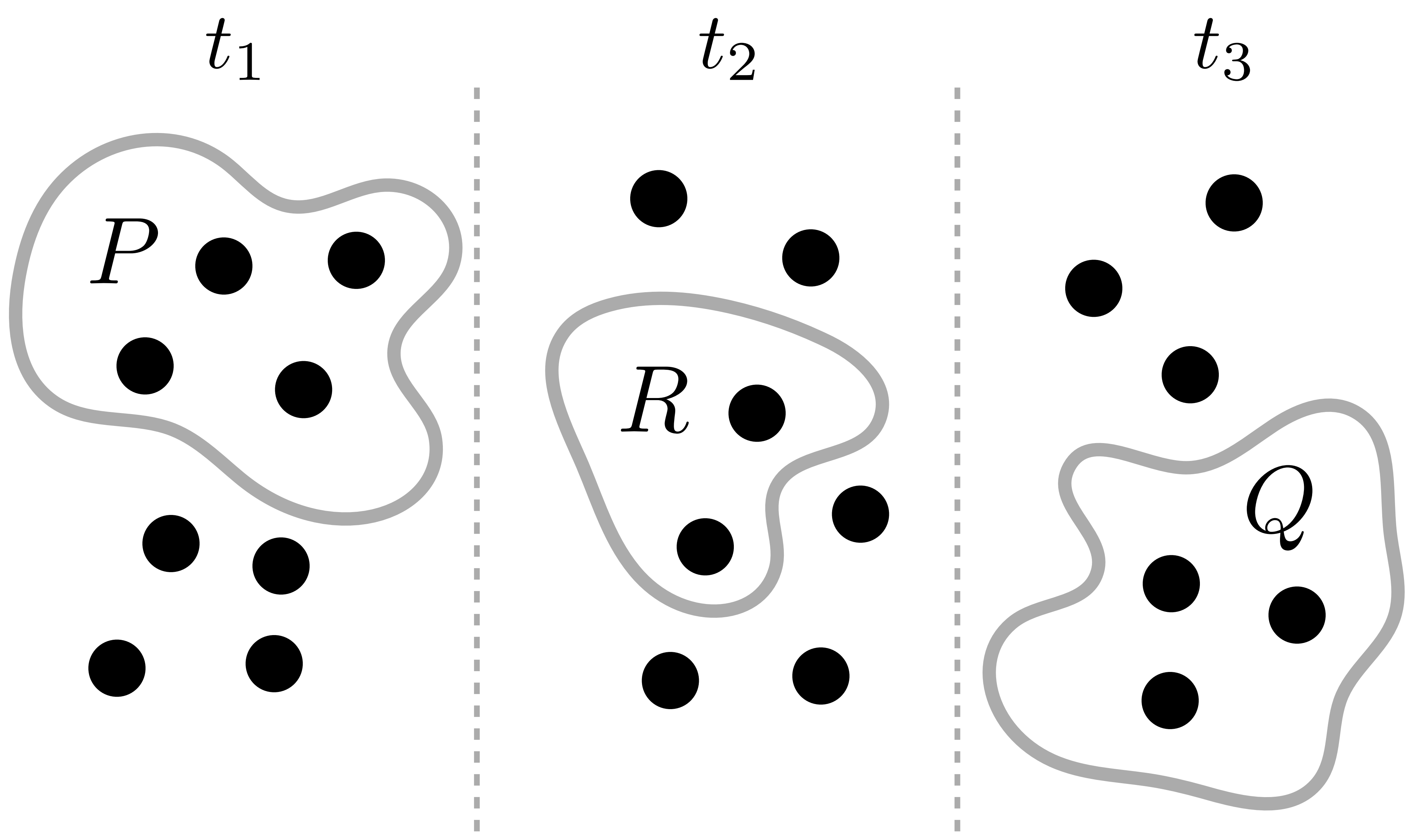
  not just current state $\boxed{Q\ \text{Now}}$

- Derive *inevitable* facts:

$$\boxed{P\ \text{Past}} \land \boxed{Q\ \text{Now}} \rightsquigarrow \boxed{R\ \text{Past}}$$

  if in the state transition system

  all paths from P to Q pass through R



Hindsight Reasoning

- Record execution history $\boxed{P \ \text{Past}}$

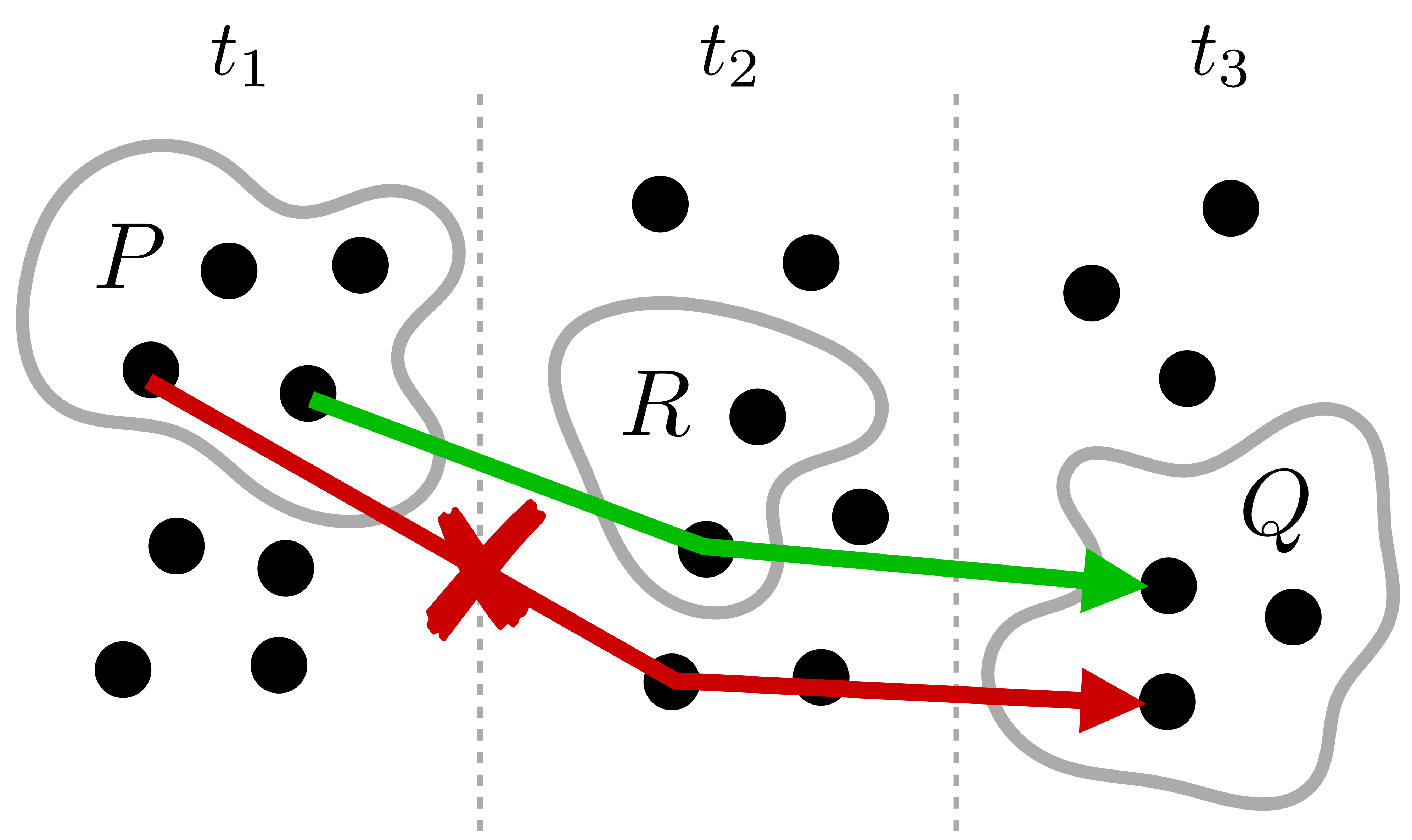
  not just current state $\boxed{Q \ \text{Now}}$

- Derive *inevitable* facts:

  $$\boxed{P \ \text{Past}} \wedge \boxed{Q \ \text{Now}} \rightsquigarrow \boxed{R \ \text{Past}}$$

  if in the state transition system

  all paths from P to Q pass through R



Hindsight Reasoning

- Record execution history P **Past**

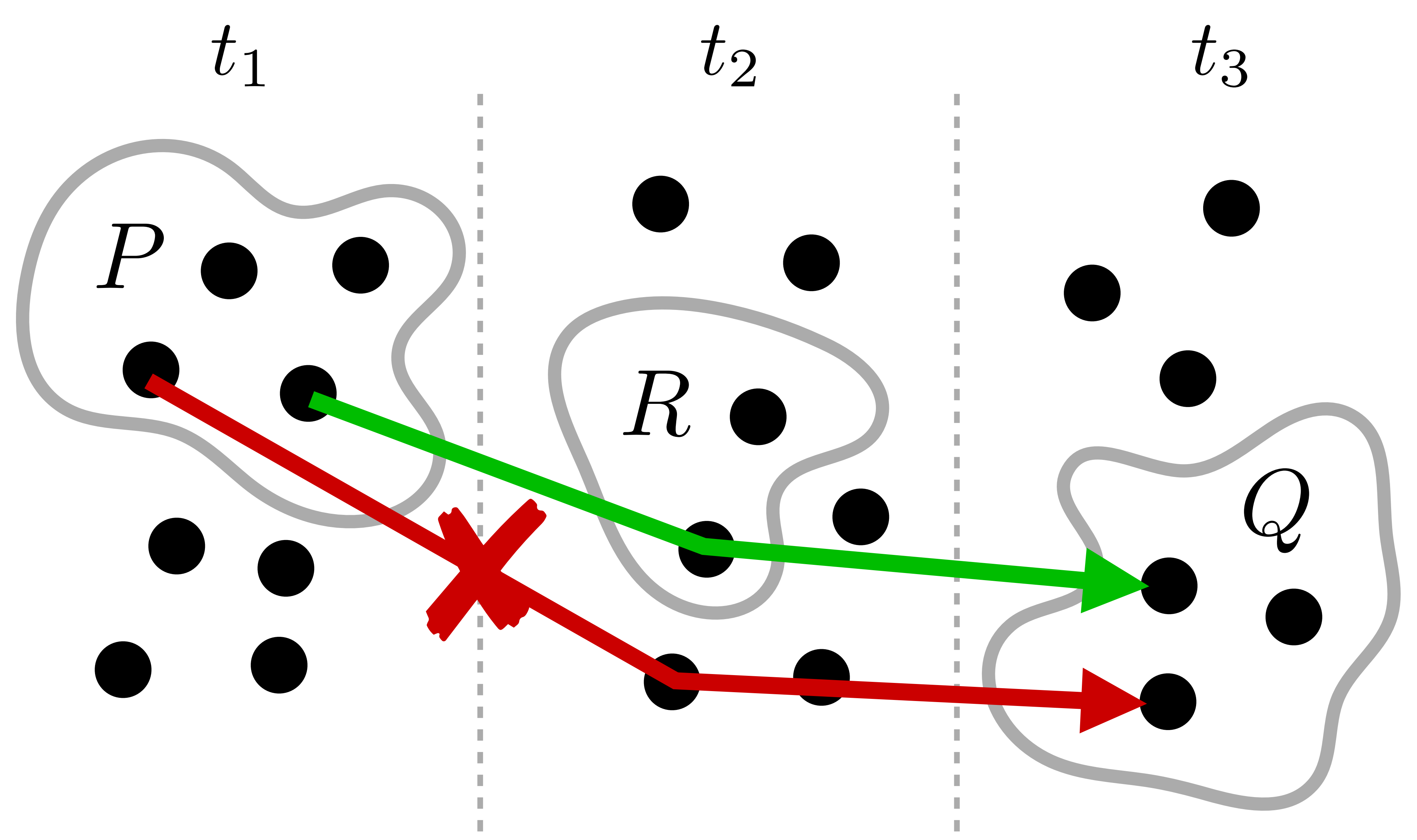
  not just current state Q **Now**

- Derive *inevitable* facts:

  P **Past** $\land$ Q **Now** $\rightsquigarrow$ R **Past**

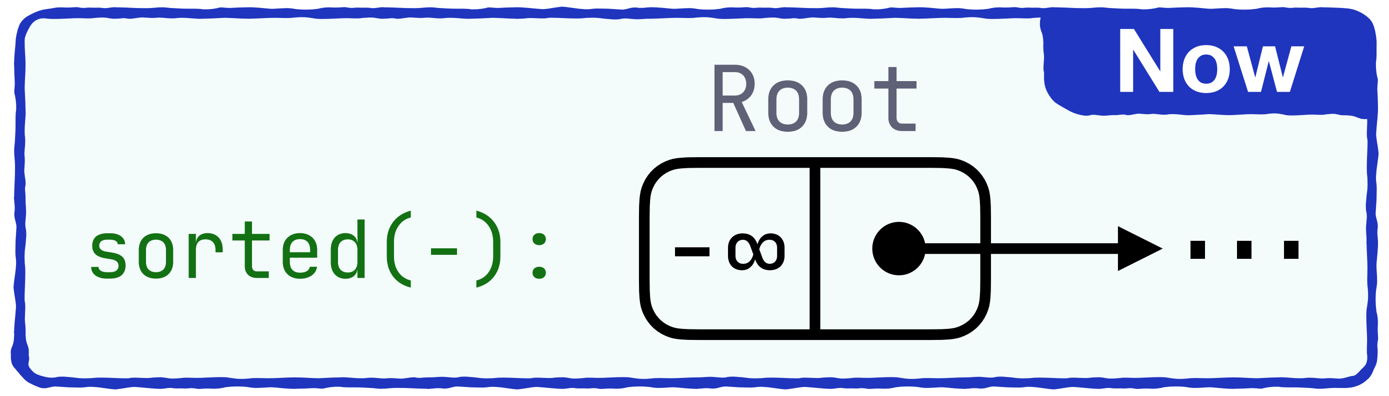
  if in the state transition system

  all paths from P to Q pass through R



derived from interference recorded by Owicki-Gries / Rely-Guarantee proof

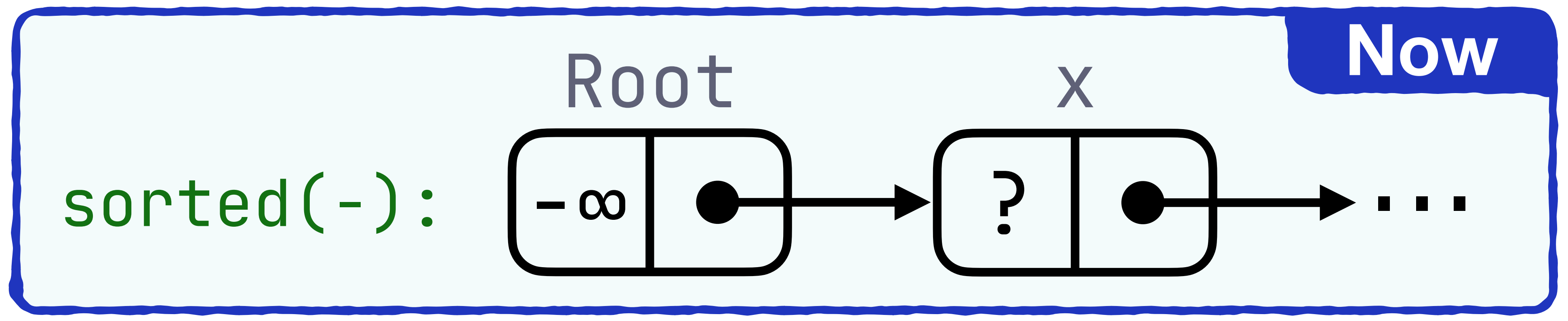
Example: `contains(5)`



```
x = Root→next;                    // traverse -∞→3
if (x→key < 5) continue;          // inspect 3
y = x→next;                       // traverse 3→7
if (y→key > 5) return false;      // inspect 7
```

Example: `contains(5)`

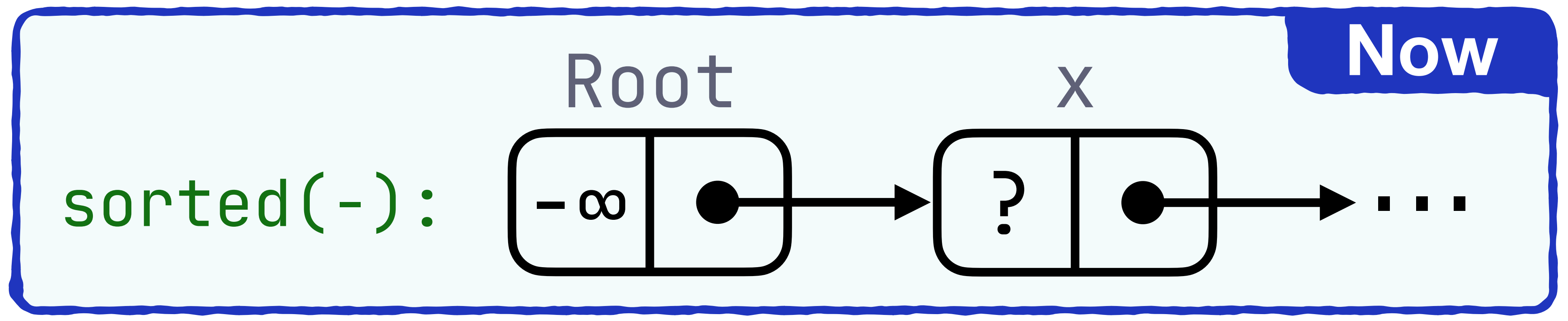
```
x = Root→next;                    // traverse -∞→3
```



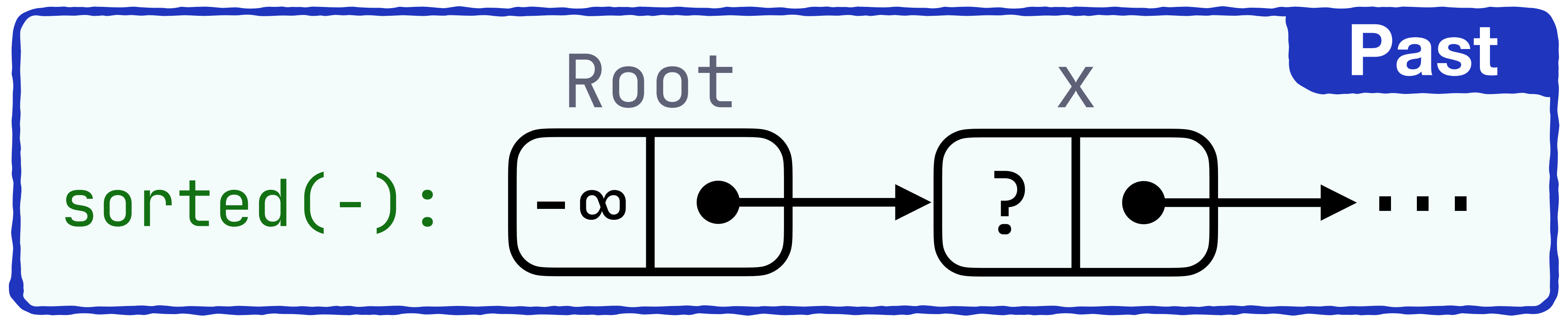
```
if (x→key < 5) continue;          // inspect 3
y = x→next;                       // traverse 3→7
if (y→key > 5) return false;      // inspect 7
```

Example: `contains(5)`

```
x = Root→next;                    // traverse -∞→3
```



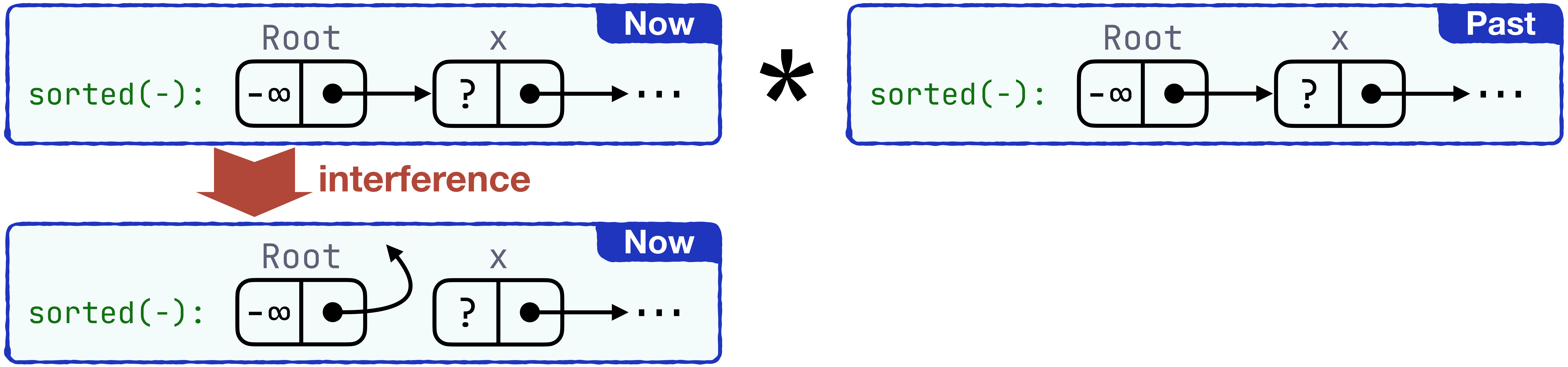
*



```
if (x→key < 5) continue;          // inspect 3

y = x→next;                       // traverse 3→7

if (y→key > 5) return false;      // inspect 7
```

Example: `contains(5)`

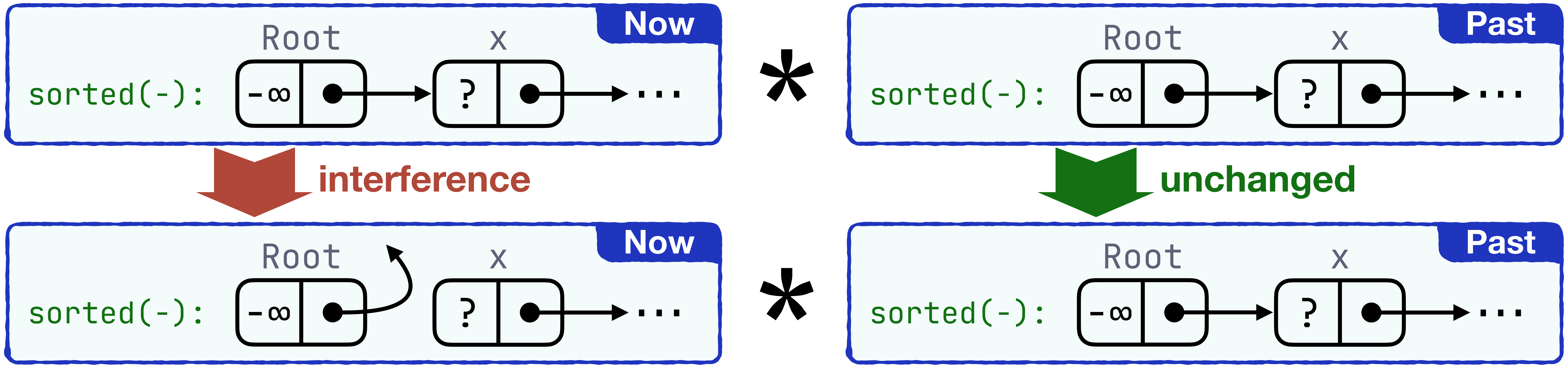
```
x = Root→next;                        // traverse -∞→3
```



```
if (x→key < 5) continue;              // inspect 3

y = x→next;                           // traverse 3→7

if (y→key > 5) return false;          // inspect 7
```

Example: `contains(5)`

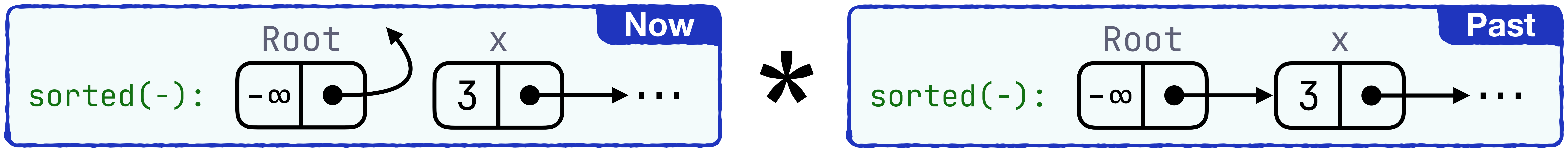
```
x = Root→next;                   // traverse -∞→3
```



```
if (x→key < 5) continue;         // inspect 3

y = x→next;                      // traverse 3→7

if (y→key > 5) return false;     // inspect 7
```

Example: `contains(5)`

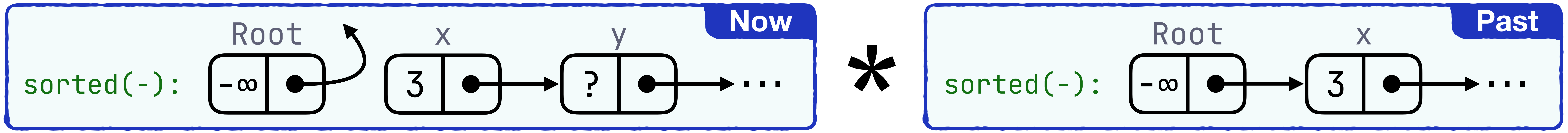
```
x = Root→next;                 // traverse -∞→3

if (x→key < 5) continue;       // inspect 3
```



```
y = x→next;                    // traverse 3→7

if (y→key > 5) return false;   // inspect 7
```

Example: `contains(5)`

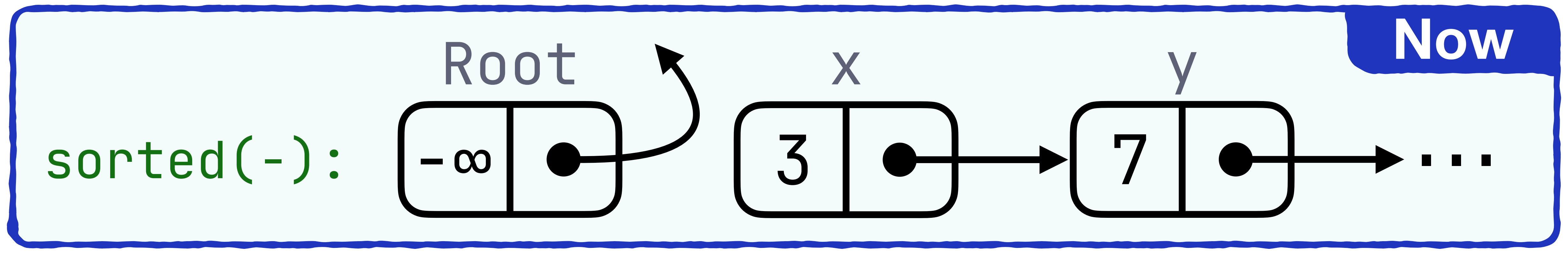
```
x = Root→next;                 // traverse -∞→3

if (x→key < 5) continue;       // inspect 3

y = x→next;                    // traverse 3→7
```



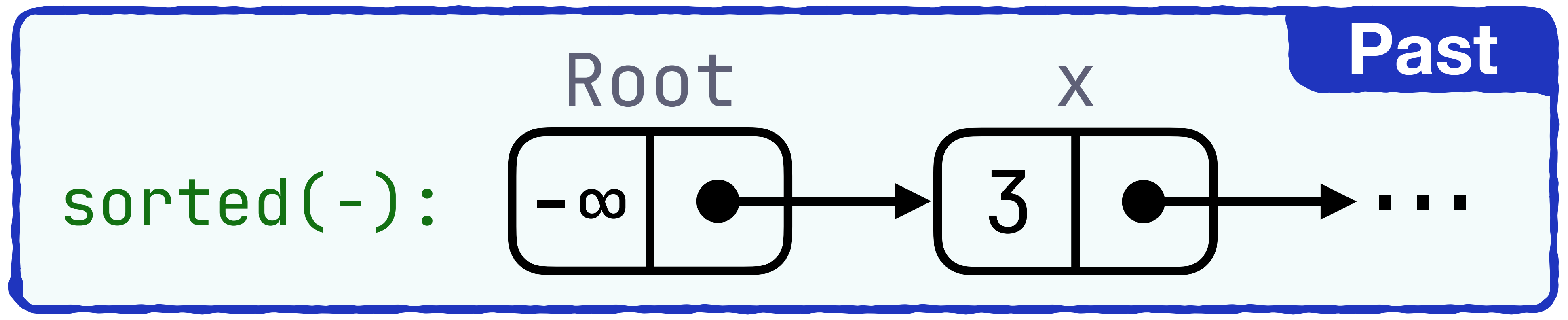
```
if (y→key > 5) return false;   // inspect 7
```

Example: `contains(5)`

```
x = Root→next;                  // traverse -∞→3
if (x→key < 5) continue;        // inspect 3
y = x→next;                     // traverse 3→7
if (y→key > 5) return false;    // inspect 7
```

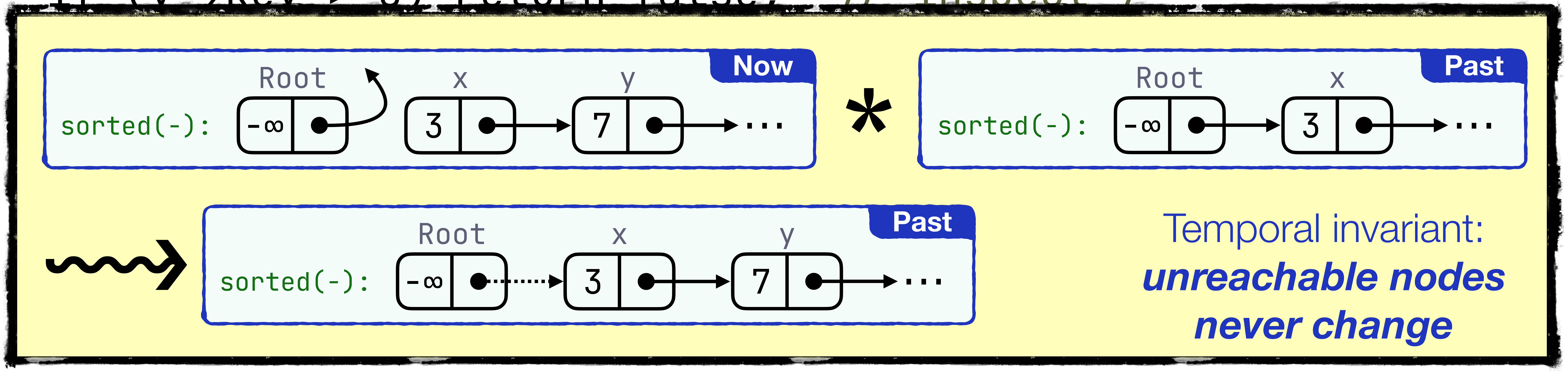


$*$



Example: `contains(5)`

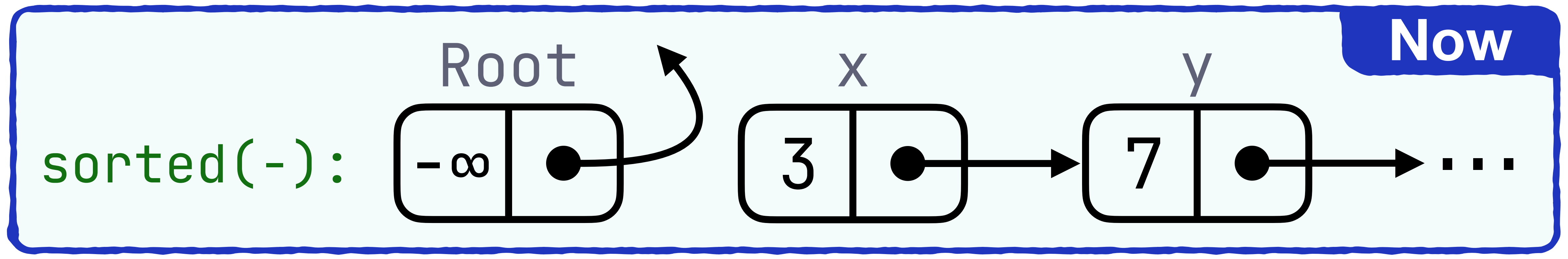
```
x = Root→next;                 // traverse -∞→3

if (x→key < 5) continue;       // inspect 3

y = x→next;                    // traverse 3→7

if (y→key > 5) return false;   // inspect 7
```



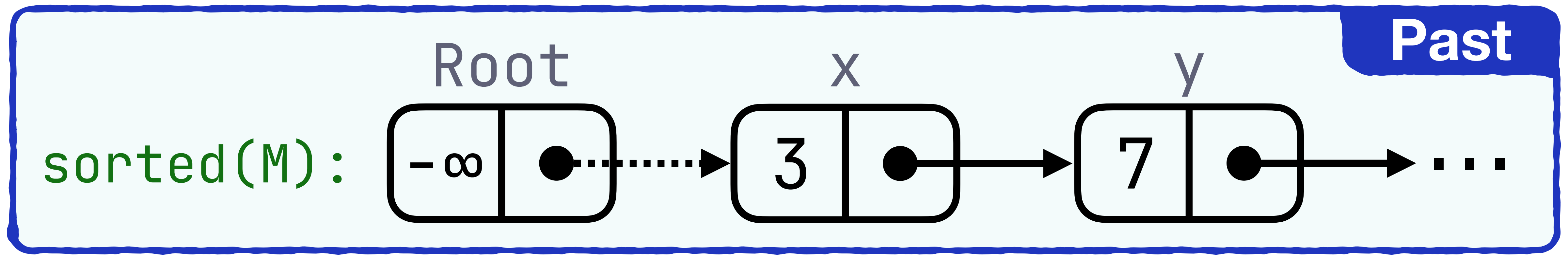
Temporal invariant:
unreachable nodes never change

Example: `contains(5)`

```
x = Root→next;                  // traverse -∞→3
if (x→key < 5) continue;        // inspect 3
y = x→next;                     // traverse 3→7
if (y→key > 5) return false;    // inspect 7
```



$*$



Example: `contains(5)`

```
x = Root→next;                 // traverse -∞→3
if (x→key < 5) continue;       // inspect 3
y = x→next;                    // traverse 3→7
if (y→key > 5) return false;   // inspect 7
```

Now
sorted(-):
Root
x
y
-∞
3
7
...
*
Past
sorted(M):
Root
x
y
-∞
3
7
...
⊨ sorted(M), 5∉M at some point

Example: `contains(5)`

```
x = Root→next;                  // traverse -∞→3

if (x→key < 5) continue;        // inspect 3

y = x→next;                     // traverse 3→7

if (y→key > 5) return false;    // inspect 7
```

Now
sorted(-):
Root
x
y
-∞
3
7
*
Past
sorted(M):
Root
x
y
-∞
3
7

⊨ sorted(M), 5∉M at some point

establishes linearizability proof obligation

Technical Contribution

- Abstract Owicki-Gries separation logic, with

Technical Contribution

- Abstract Owicki-Gries separation logic, with
  - temporal interpolation proof rule

$$\frac{\mathsf{Interpolate}(P, Q) = R}{\{S \wedge \mathsf{Past}(P) \wedge \mathsf{Now}(Q)\}\ \mathtt{skip}\ \{S \wedge \mathsf{Past}(R)\}}$$

Technical Contribution

- Abstract Owicki-Gries separation logic, with
    - temporal interpolation proof rule

$$\frac{\mathsf{Interpolate}(P, Q) = R}{\{S \land \mathsf{Past}(P) \land \mathsf{Now}(Q)\}\ \texttt{skip}\ \{S \land \mathsf{Past}(R)\}}$$

    - interpolation strategy: temporal invariant

$$\frac{\exists I. \quad \mathsf{Now}(P) \implies I \qquad I \text{ interference-free} \qquad I \land \mathsf{Now}(Q) \implies \mathsf{Past}(R)}{\mathsf{Interpolate}(P, Q) = R}$$

Technical Contribution

- Abstract Owicki-Gries separation logic, with
  - temporal interpolation proof rule

$$\frac{\mathsf{Interpolate}(P, Q) = R}{\{S \wedge \mathsf{Past}(P) \wedge \mathsf{Now}(Q)\}\ \texttt{skip}\ \{S \wedge \mathsf{Past}(R)\}}$$

  - interpolation strategy: temporal invariant

$$\frac{\exists I. \quad \mathsf{Now}(P) \implies I \qquad I \text{ interference-free} \qquad I \wedge \mathsf{Now}(Q) \implies \mathsf{Past}(R)}{\mathsf{Interpolate}(P, Q) = R}$$

- Soundness: by elimination of interpolation rule

temporal interpolation
= proof structuring

Implementation

- Automatic linearizability checker using temporal interpolation
- *Logical Ordering Tree*
  - found bug in implementation
  - found bug in previous proof
  - first proof of (fixed) version

Benchmark	Verification Time
Fine-Grained set	45*s* ✓
Lazy set	2*m* 13*s* ✓
FEMRS tree (no maintenance)	3*m* 50*s* ✓
Vechev&Yahav 2CAS set	1*m* 15*s* ✓
Vechev&Yahav CAS set	2*m* 20*s* ✓
ORVYY set	1*m* 36*s* ✓
Michael set	6*m* 53*s* ✓
Michael set (wait-free search)	6*m* 53*s* ✓
Harris set	57*m* 20*s* ✓
Harris set (wait-free search)	43*m* 00*s* ✓
LO-tree (generalized maintenance)	16*m* 43*s* ✓

Thanks

Implementation

- Automatic linearizability checker using temporal interpolation
- *Logical Ordering Tree*
  - found bug in implementation
  - found bug in previous proof
  - first proof of (fixed) version

Benchmark	Verification Time
Fine-Grained set	45*s* ✓
Lazy set	2*m* 13*s* ✓
FEMRS tree (no maintenance)	3*m* 50*s* ✓
Vechev&Yahav 2CAS set	1*m* 15*s* ✓
Vechev&Yahav CAS set	2*m* 20*s* ✓
ORVYY set	1*m* 36*s* ✓
Michael set	6*m* 53*s* ✓
Michael set (wait-free search)	6*m* 53*s* ✓
Harris set	57*m* 20*s* ✓
Harris set (wait-free search)	43*m* 00*s* ✓
LO-tree (generalized maintenance)	16*m* 43*s* ✓