Statically Checking API Protocol Conformance with Mined Multi-Object Specifications

Michael Pradel[1], Ciera Jaspan[2], Jonathan Aldrich[3], and Thomas R. Gross[1]

[1] ETH Zurich

[2] Cal Poly Pomona

[3] Carnegie Mellon University

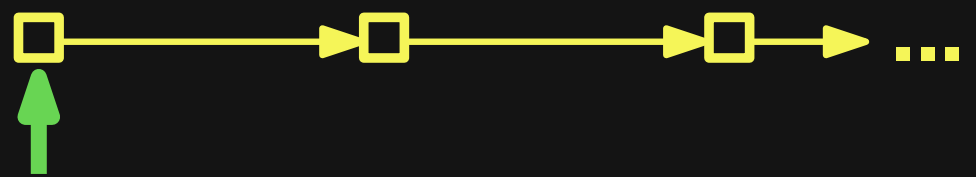
Motivating Example

```
LinkedList pinConnections = ...
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```

(from DaCapo benchmarks)

Motivating Example

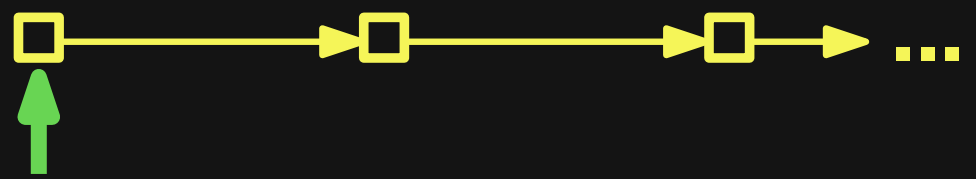
```
LinkedList pinConnections = ...
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



(from DaCapo benchmarks)

Motivating Example

```
LinkedList pinConnections = ...
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



(from DaCapo benchmarks)

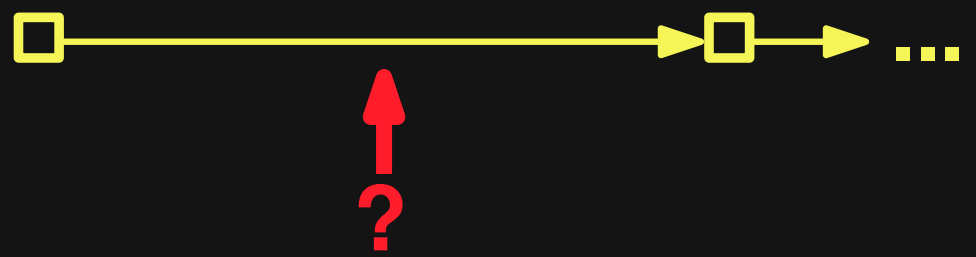
Motivating Example

```
LinkedList pinConnections = ...
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```

(from DaCapo benchmarks)

Motivating Example

```
LinkedList pinConnections = ...
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



(from DaCapo benchmarks)

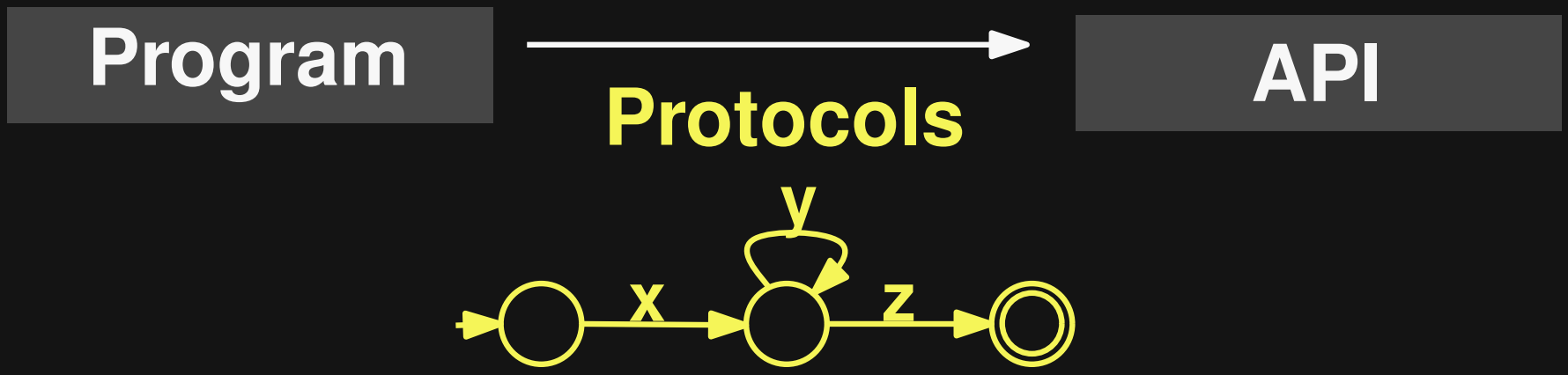
Motivating Example

```
LinkedList pinConnections = ...
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```

Don't modify a collection while iterating over it!

(from DaCapo benchmarks)

The Problem



FSMs that capture constraints like:

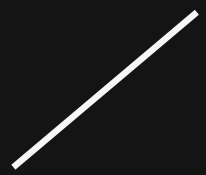
- Call x before y
- Don’t call y after calling x
- Don’t call x while calling y and z

Multi-Object Protocols

Many protocols involve multiple objects

Multi-Object Protocols

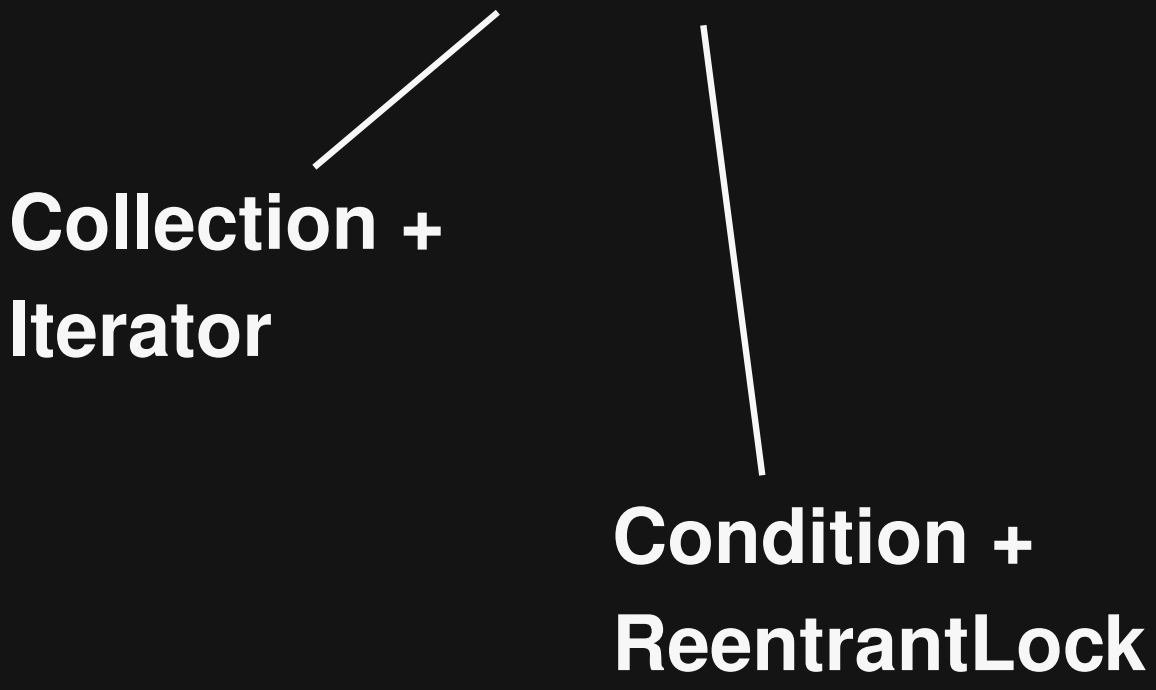
Many protocols involve multiple objects



Collection +
Iterator

Multi-Object Protocols

Many protocols involve multiple objects



Multi-Object Protocols

Many protocols involve multiple objects

Multi-Object Protocols

Many protocols involve multiple objects

How to find incorrect API usages that involve multiple objects?

Collection + Iterator

Condition + ReentrantLock

FileReader + BufferedReader

This Talk

Automatic detection of multi-object protocol bugs

This Talk

Don’t require API specs

Automatic detection of multi-object protocol bugs

This Talk

Automatic detection of multi-object protocol bugs

Warnings about:

- **Missing calls**
- **Incorrect calls**

State of the Art

DeLine + Fähndrich ’04

Bierhoff + Aldrich ’07

Type state checking

Fink et al., ’08

Naeem + Lhotak, ’08

Bodden ’10

Whaley et al. ’02

Wasylkowski + Zeller ’09

Anomaly Detection

Gabel + Su ’10

Nguyen et al. ’09

Open Issues

Type state checking:

- Needs API specs
- Sound but incomplete

Anomaly detection:

- Single-object
- Missing calls
  (not: incorrect calls)

Open Issues

Type state checking:

- Needs API specs
- Sound but incomplete

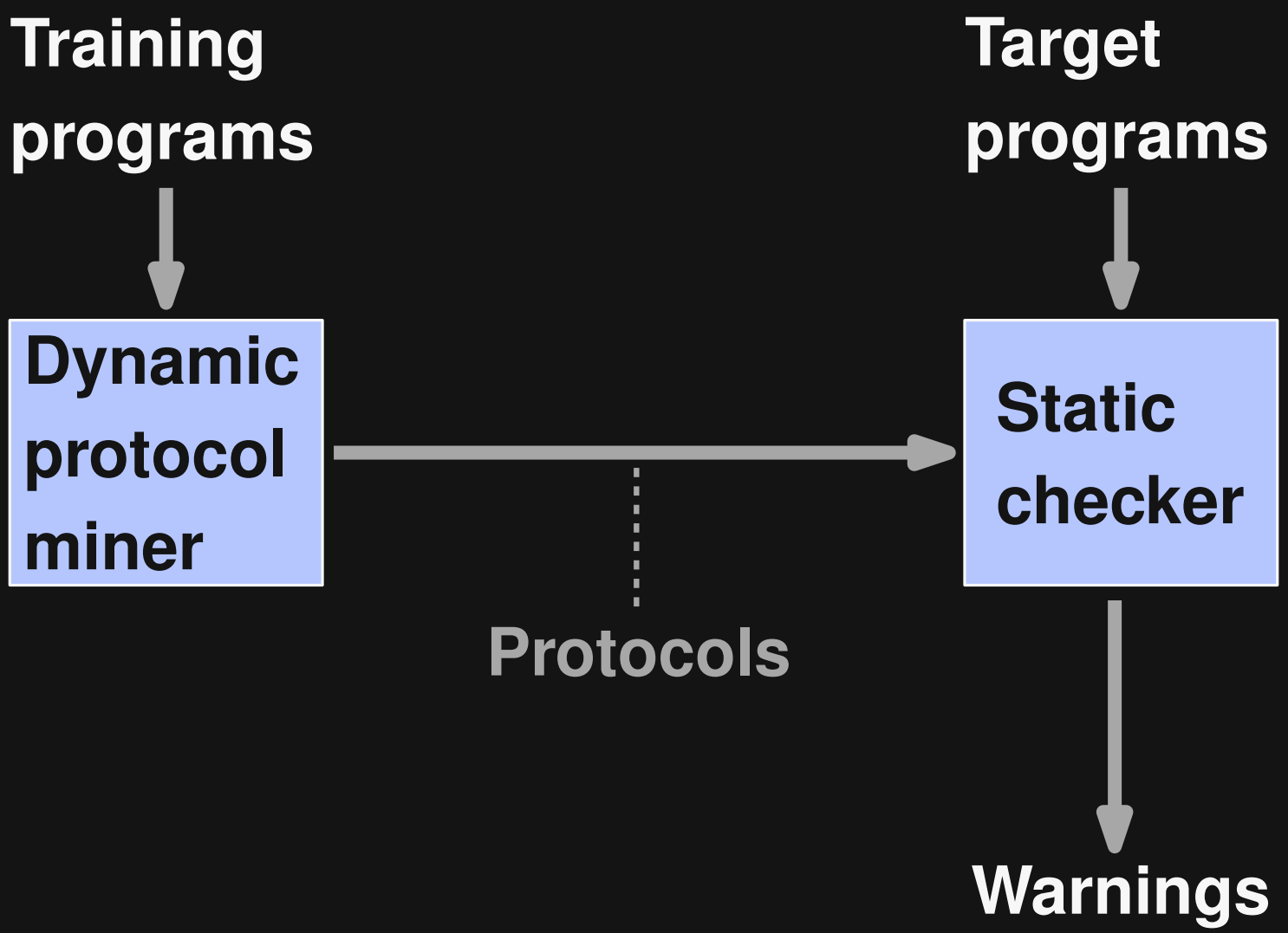
Anomaly detection:

- Single-object
- Missing calls
  (not: incorrect calls)

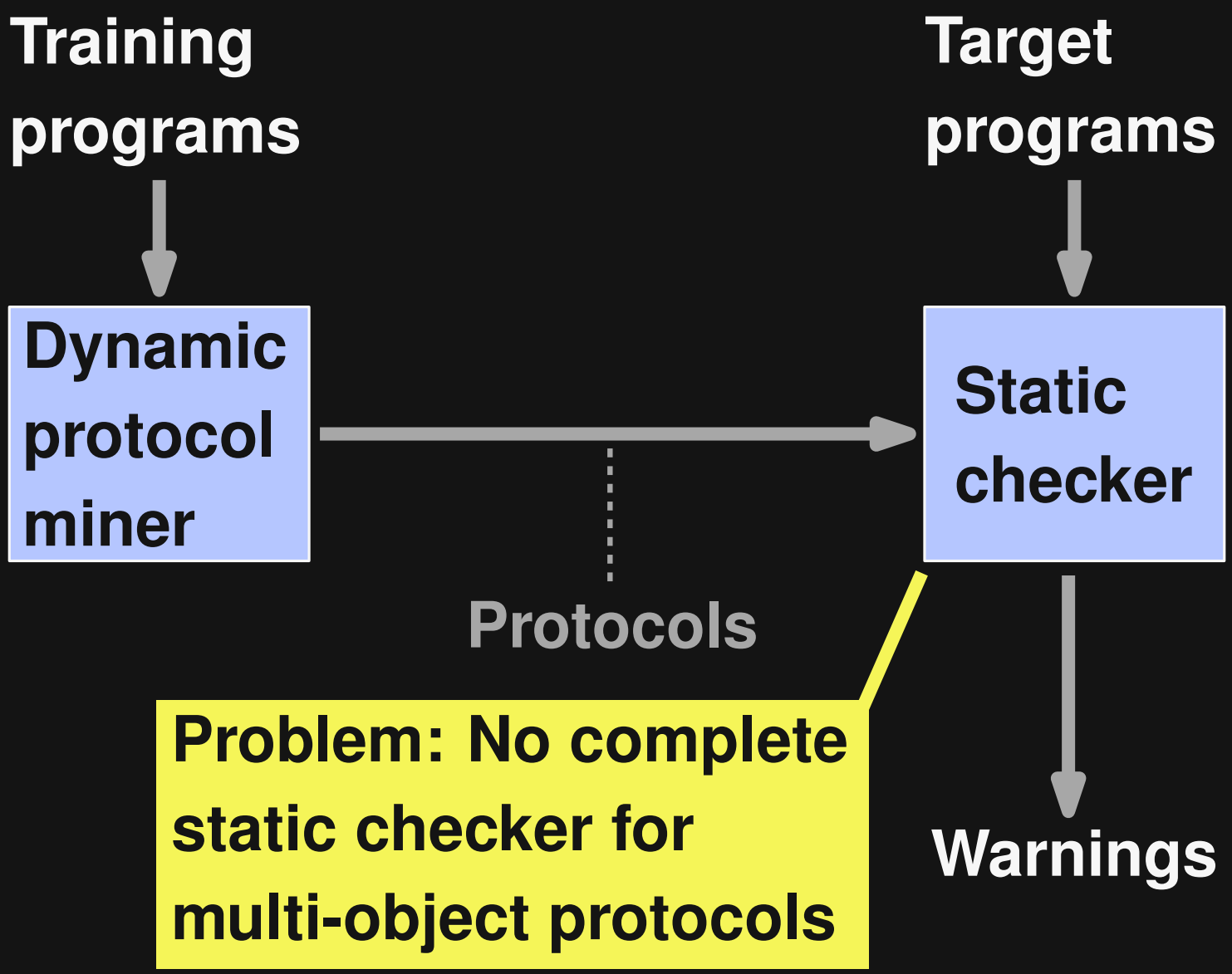
Our contributions:

- Multi-object
- Missing calls and incorrect calls
- Complete checking

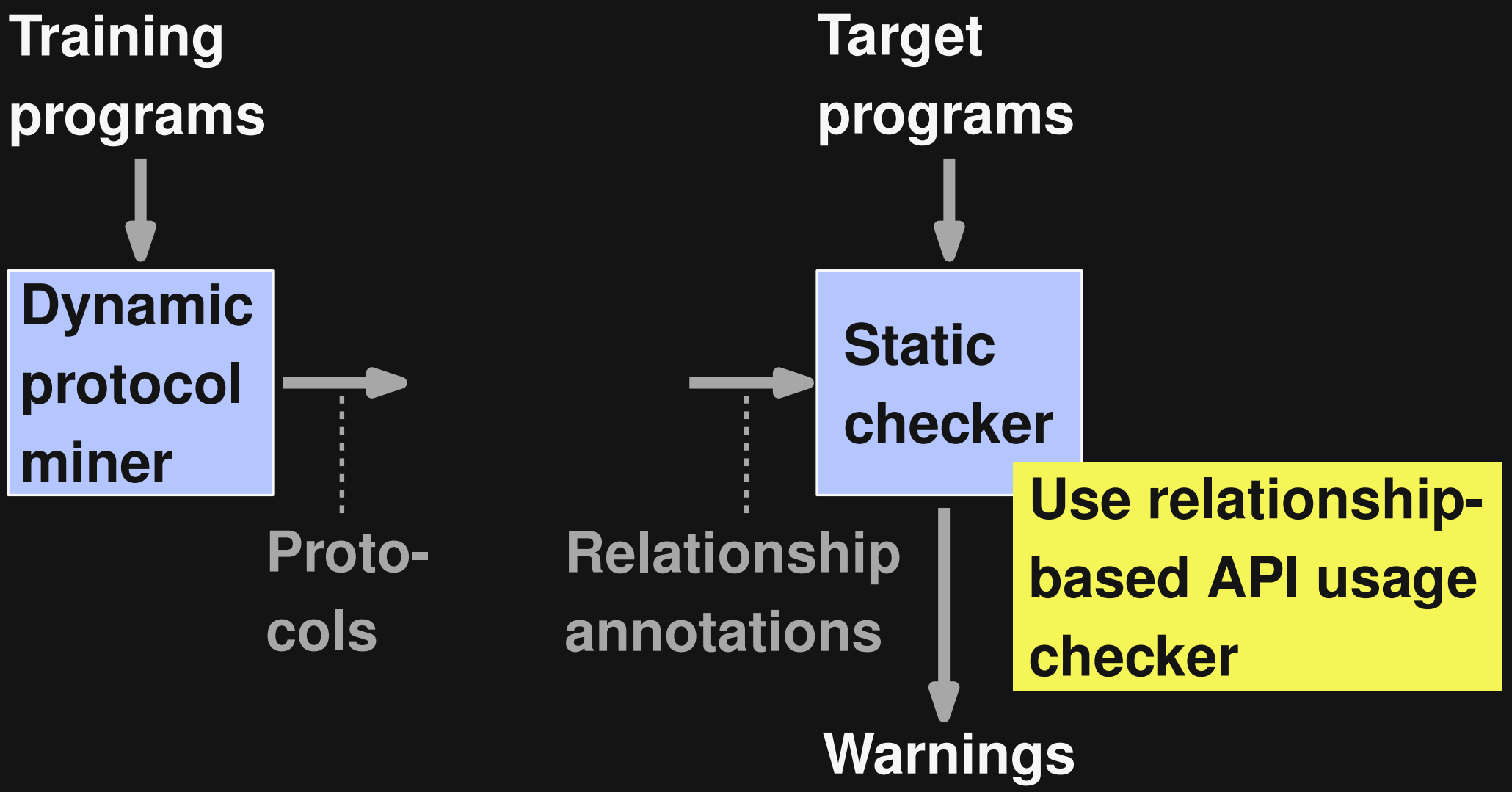
Approach Overview



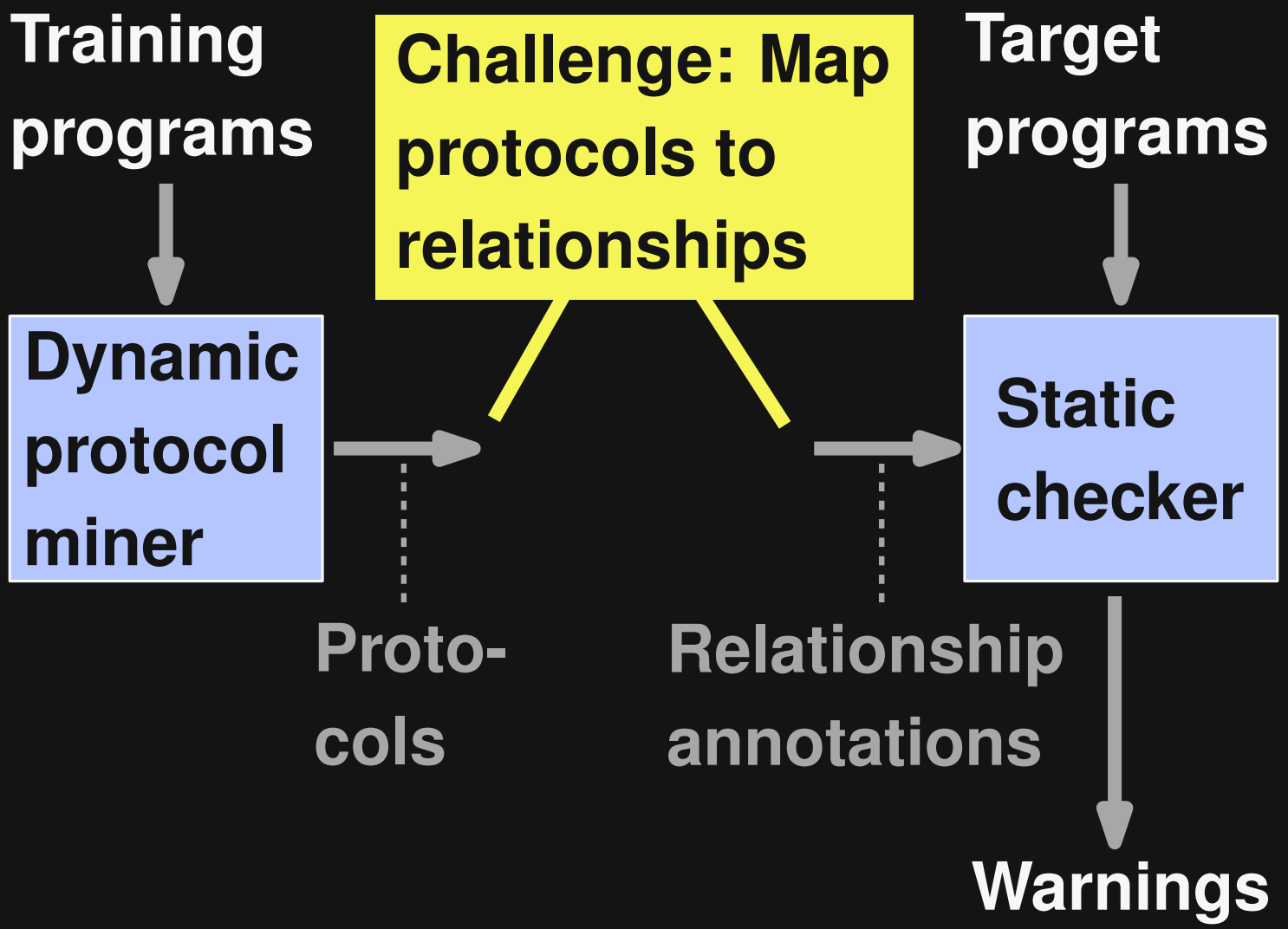
Approach Overview



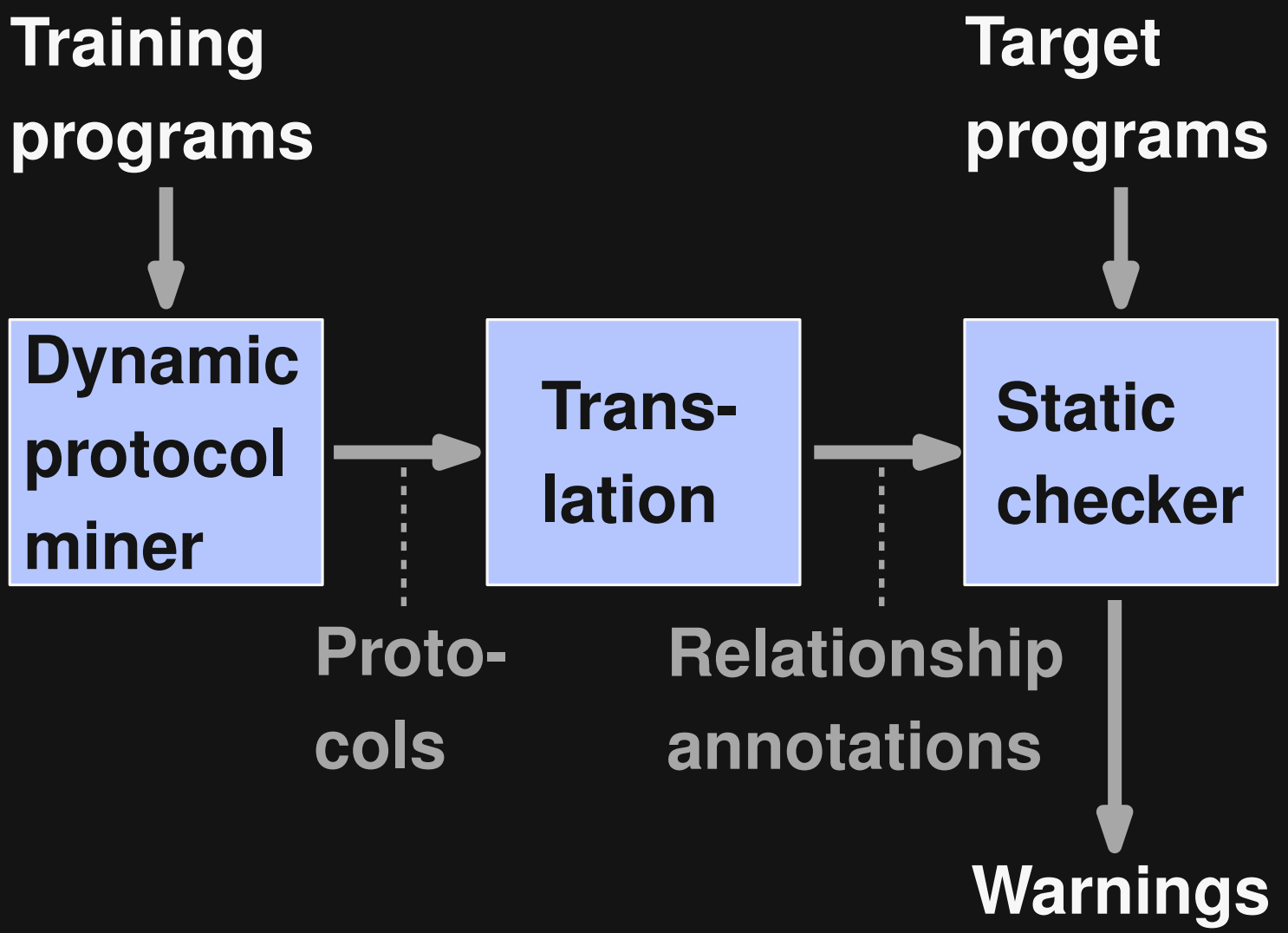
Approach Overview



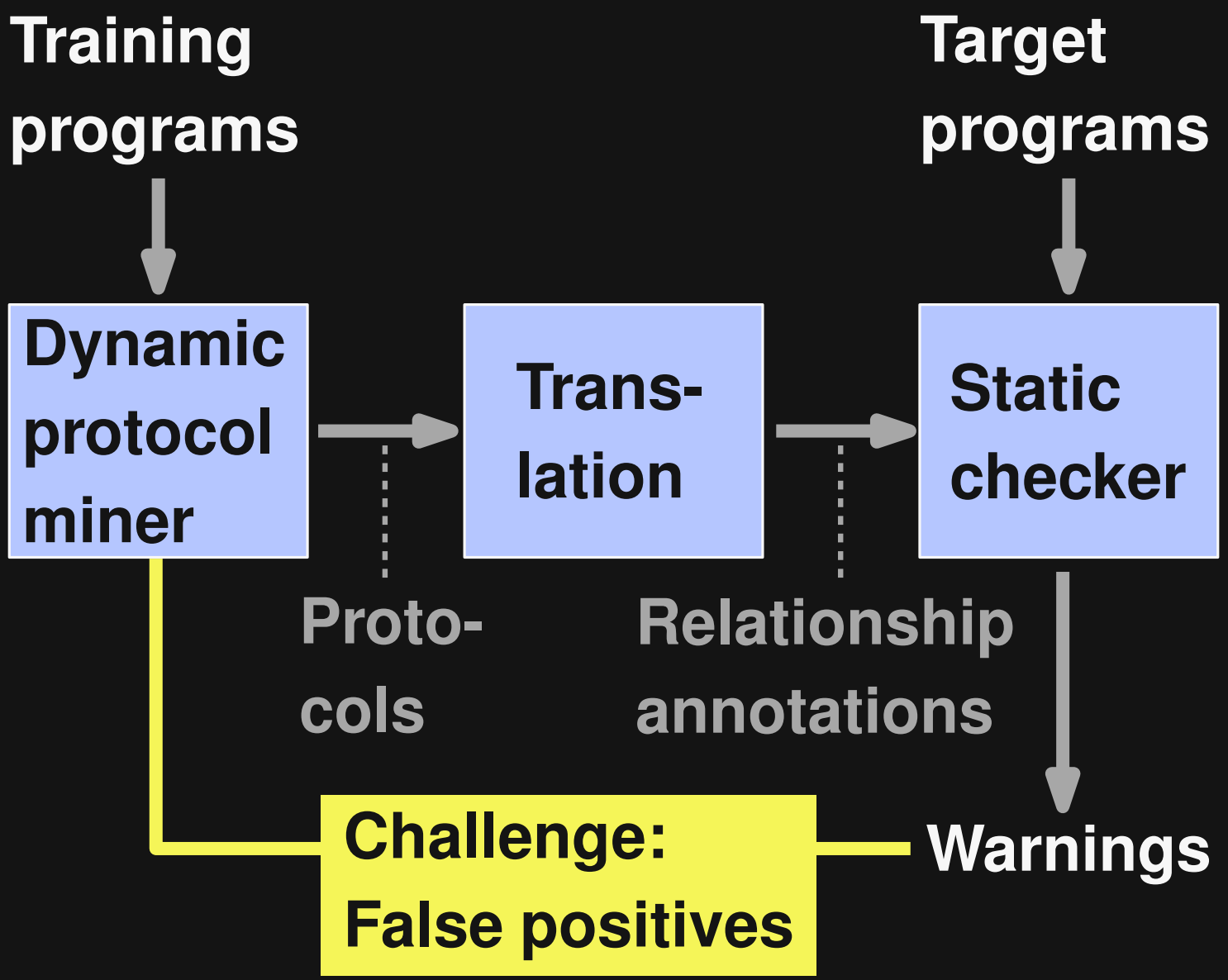
Approach Overview



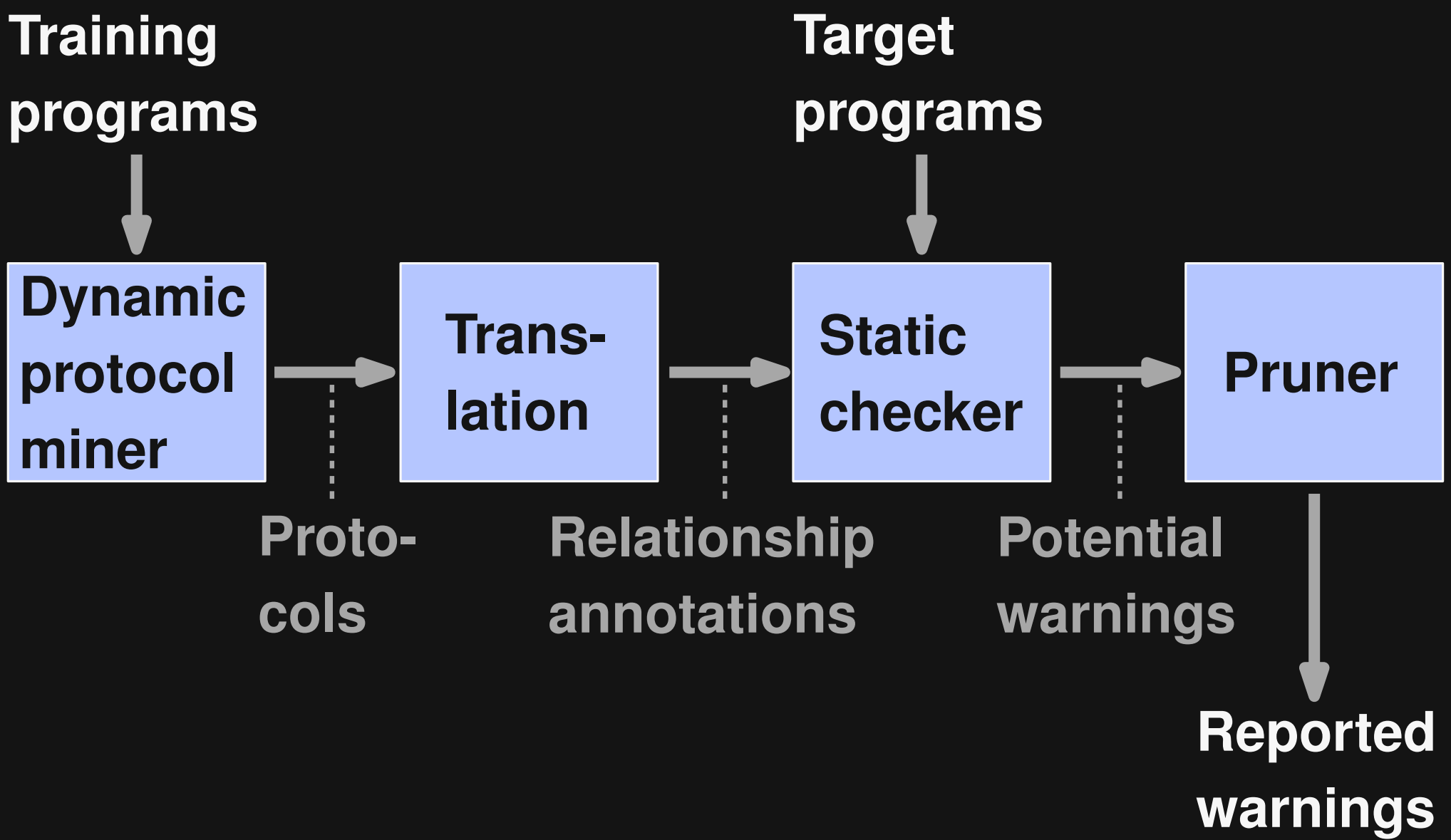
Approach Overview



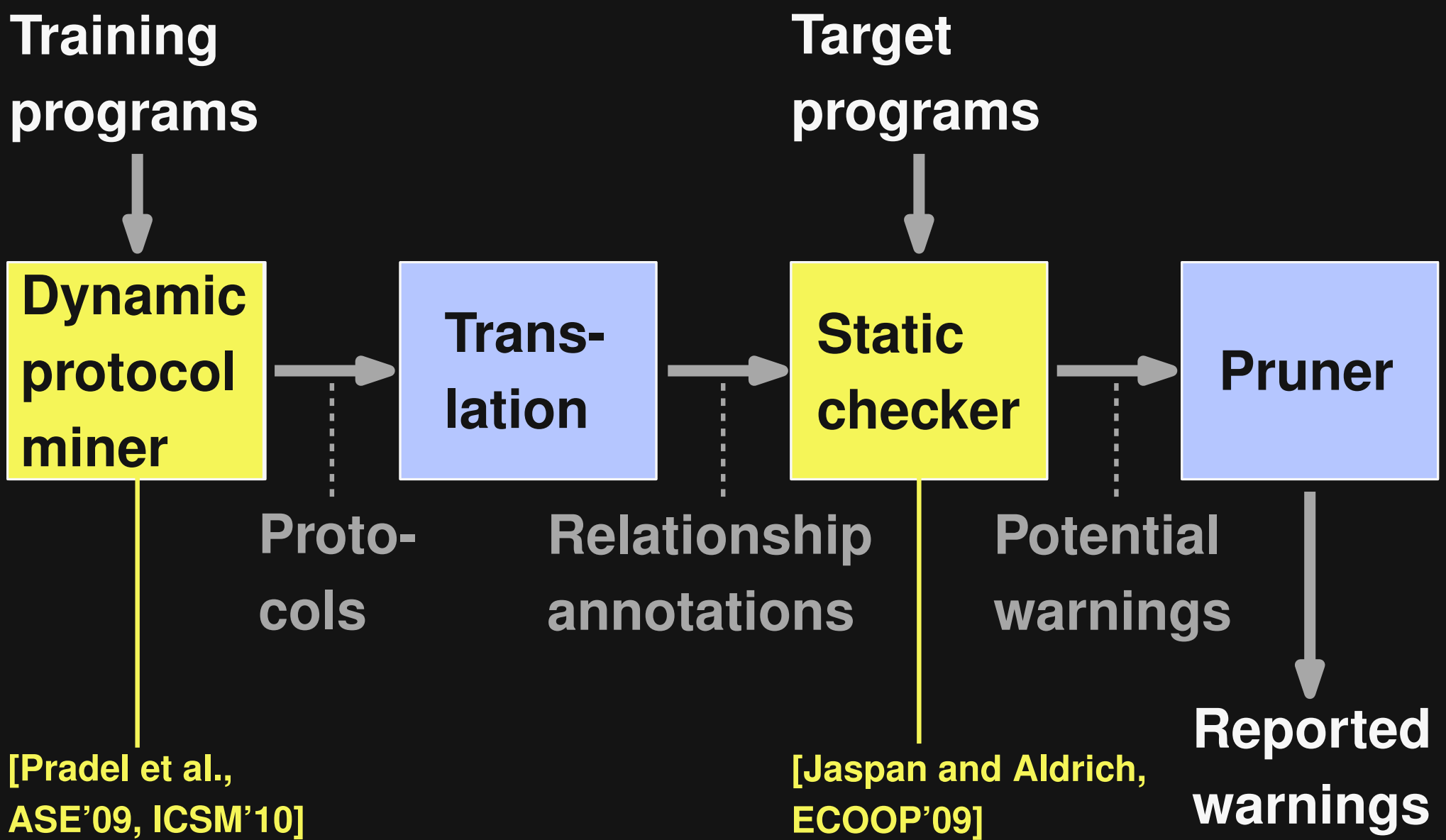
Approach Overview



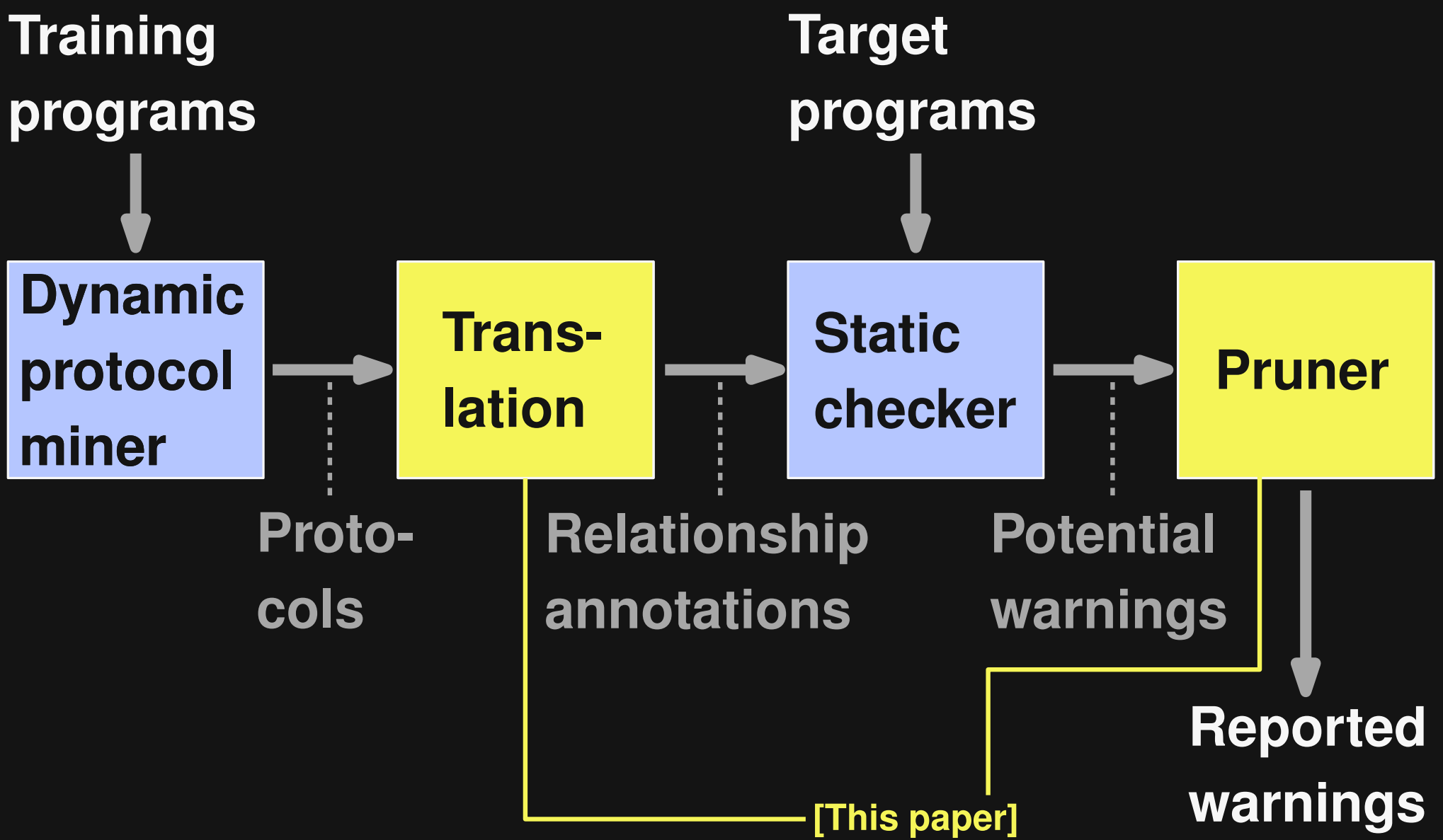
Approach Overview



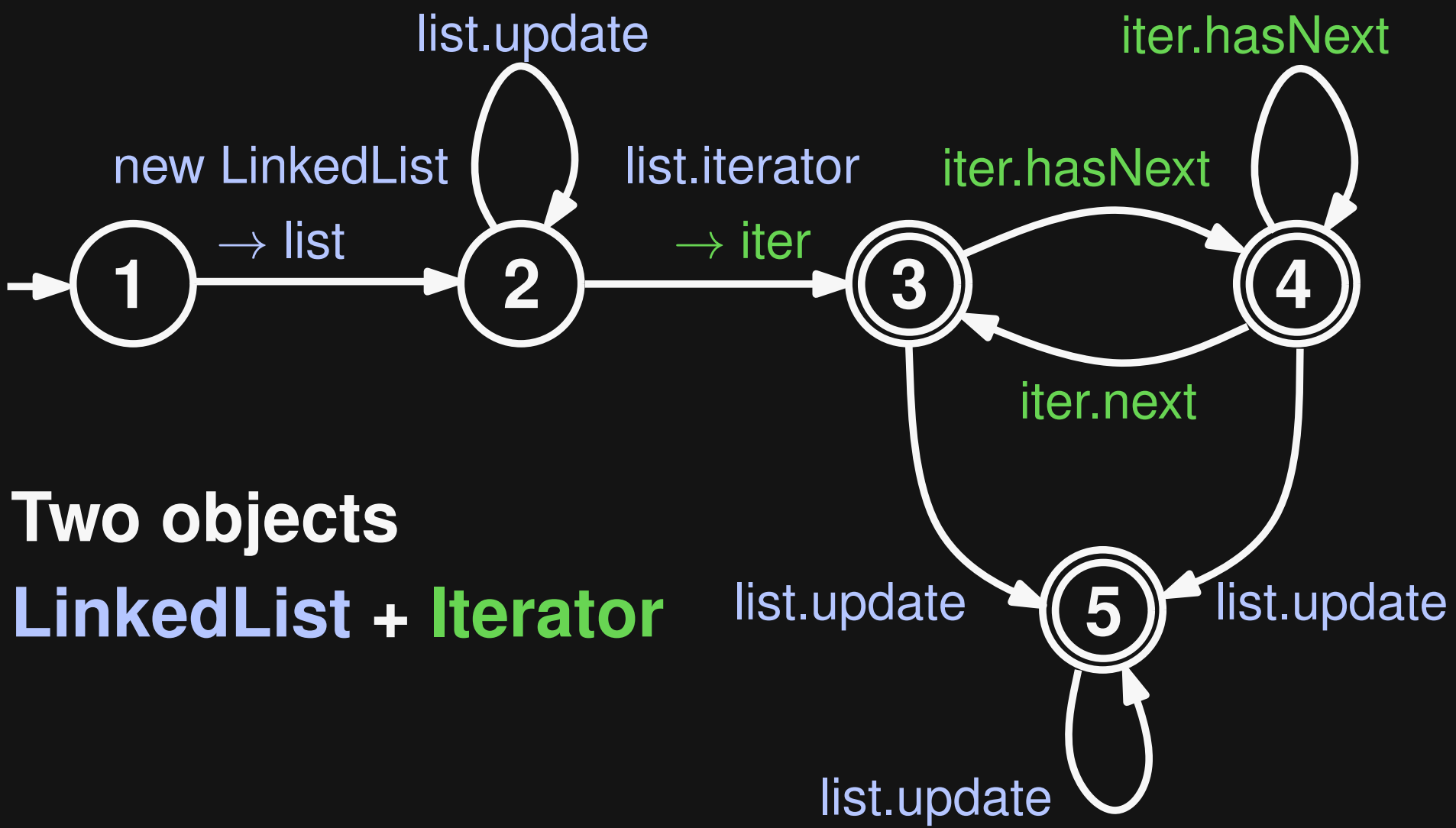
Approach Overview



Approach Overview



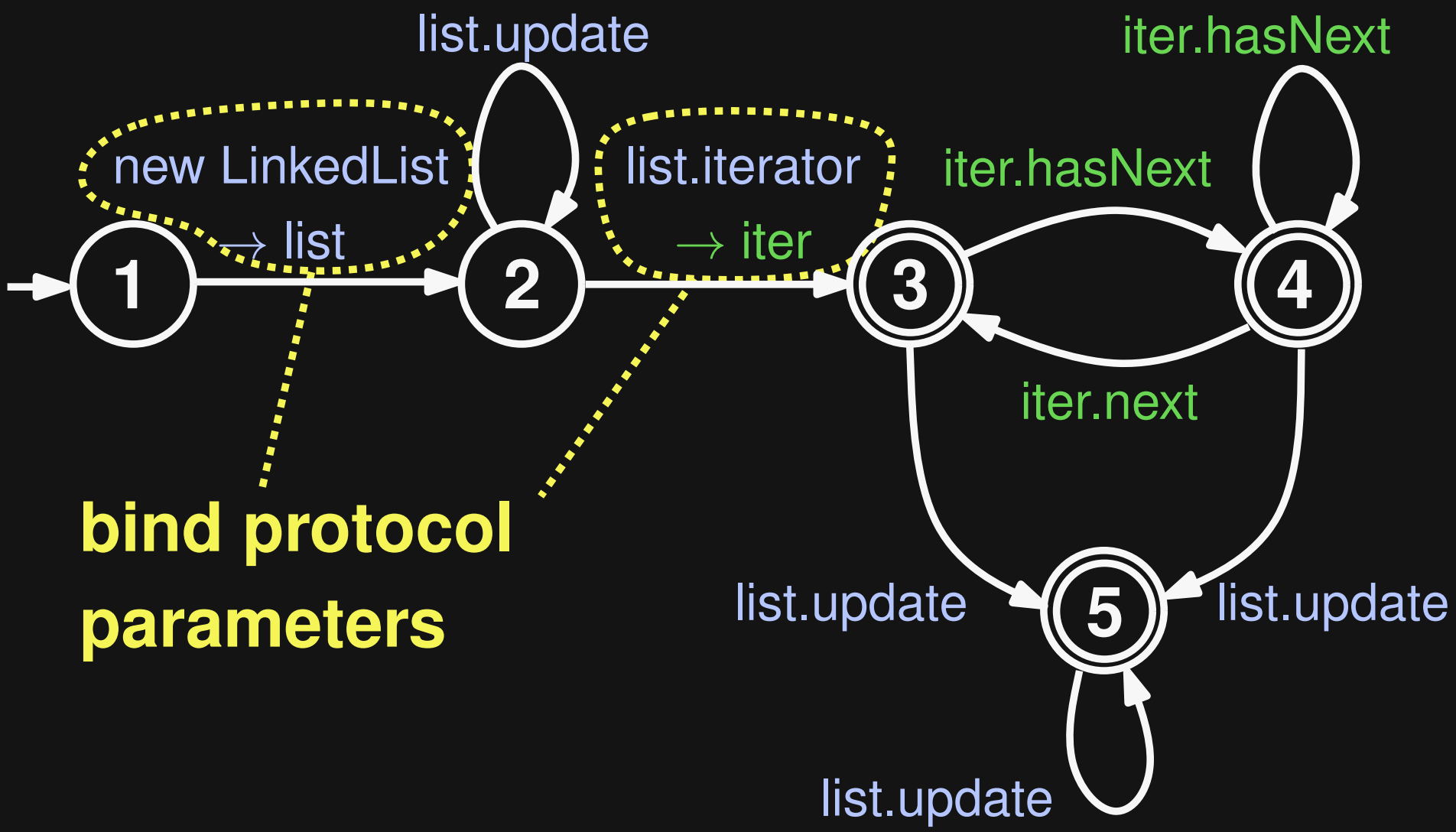
Example: Inferred Protocol



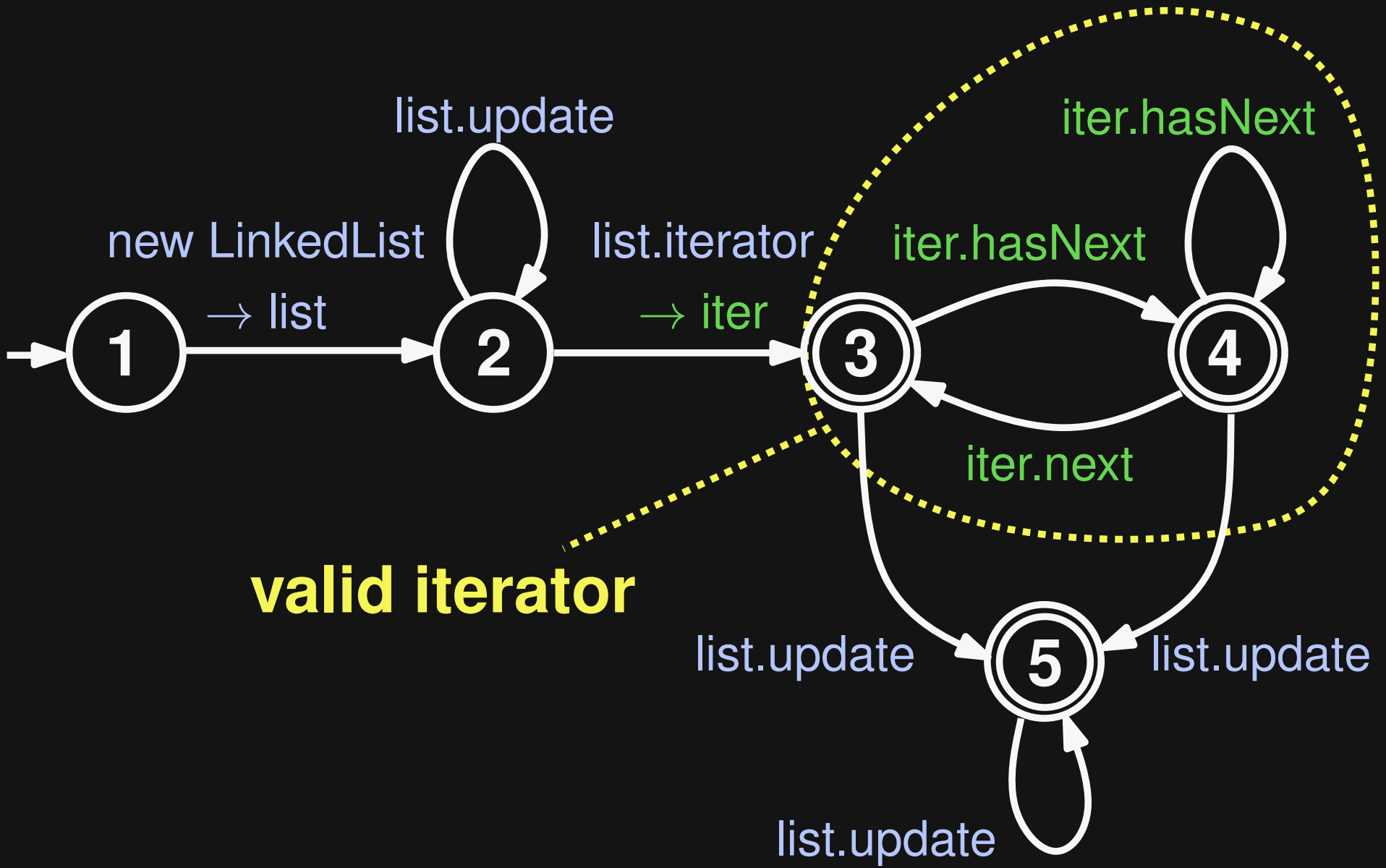
Two objects

LinkedList + Iterator

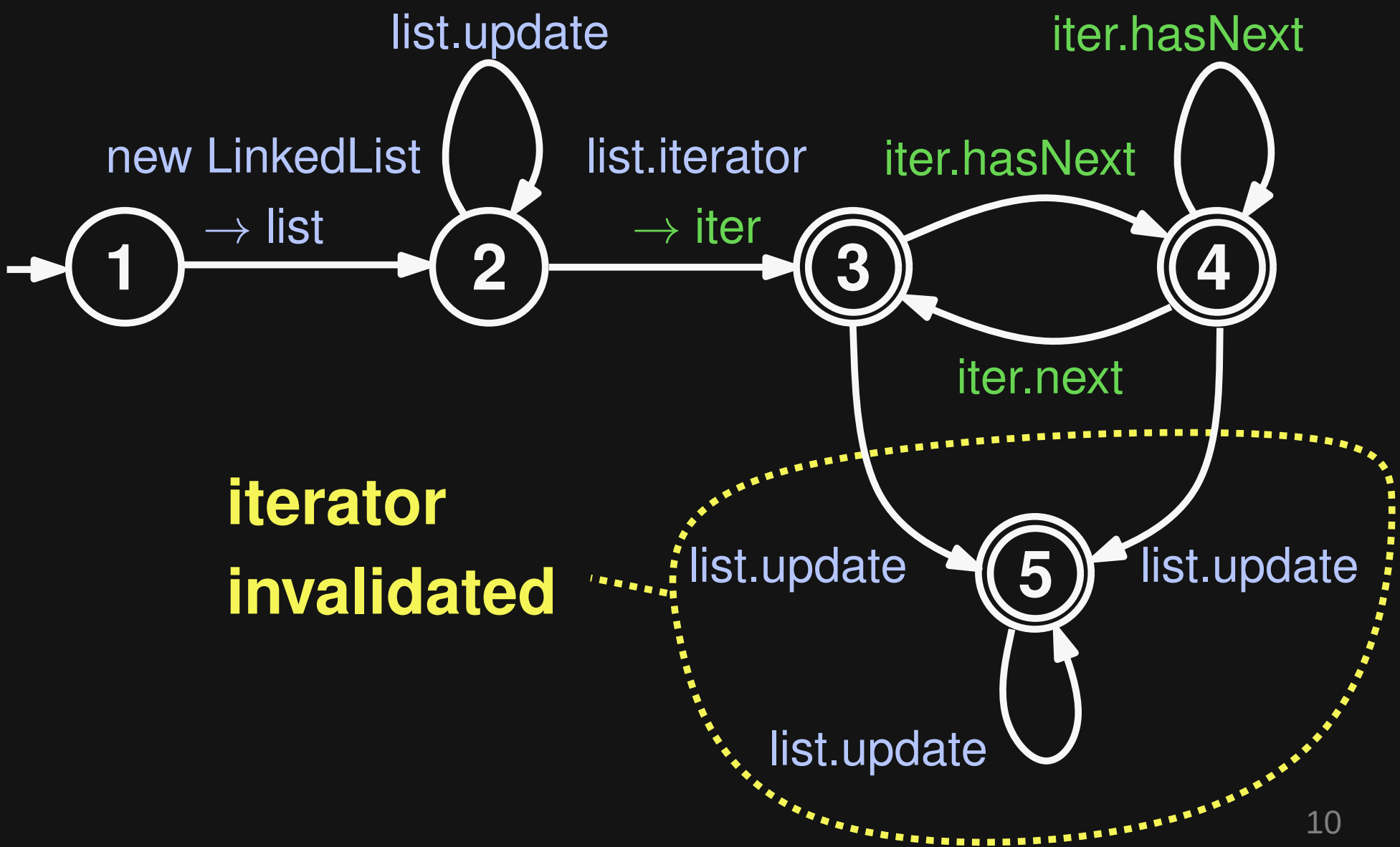
Example: Inferred Protocol



Example: Inferred Protocol

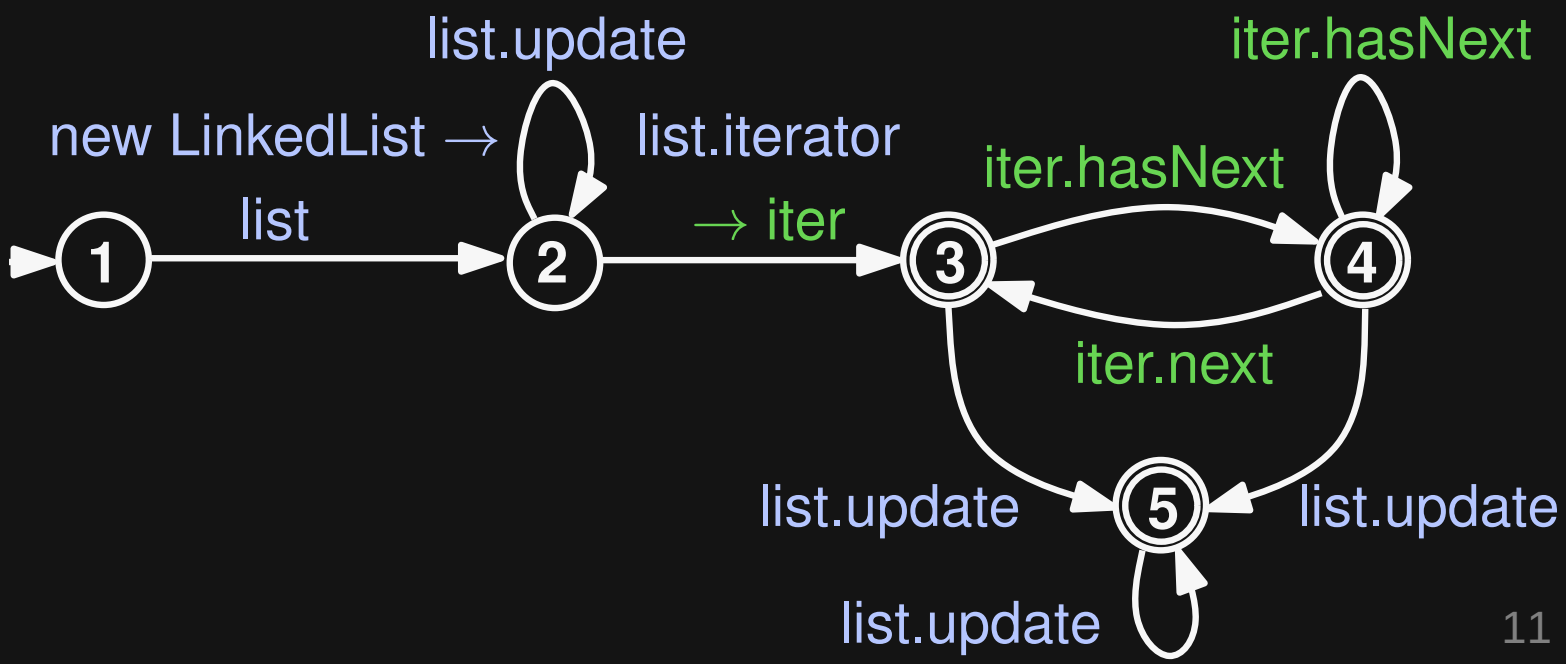


Example: Inferred Protocol



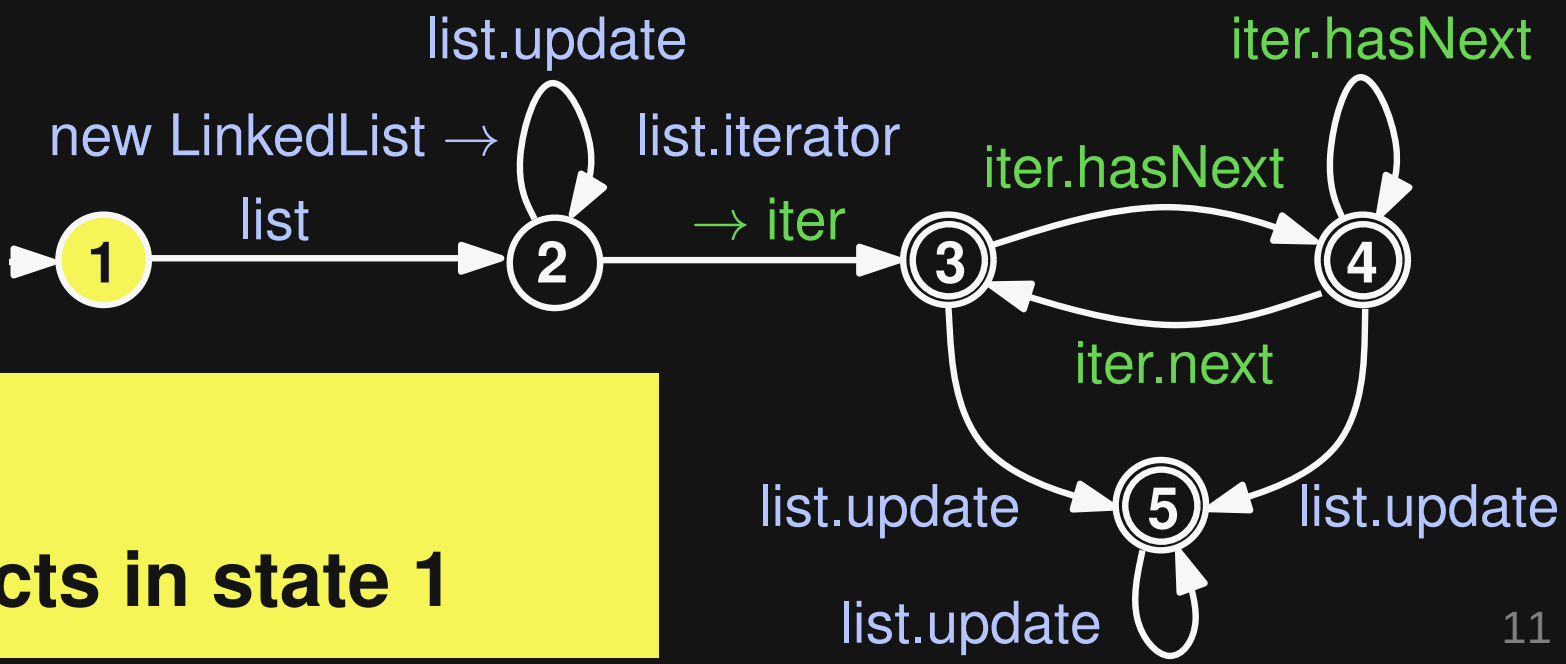
Example: Checking

```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



Example: Checking

```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```

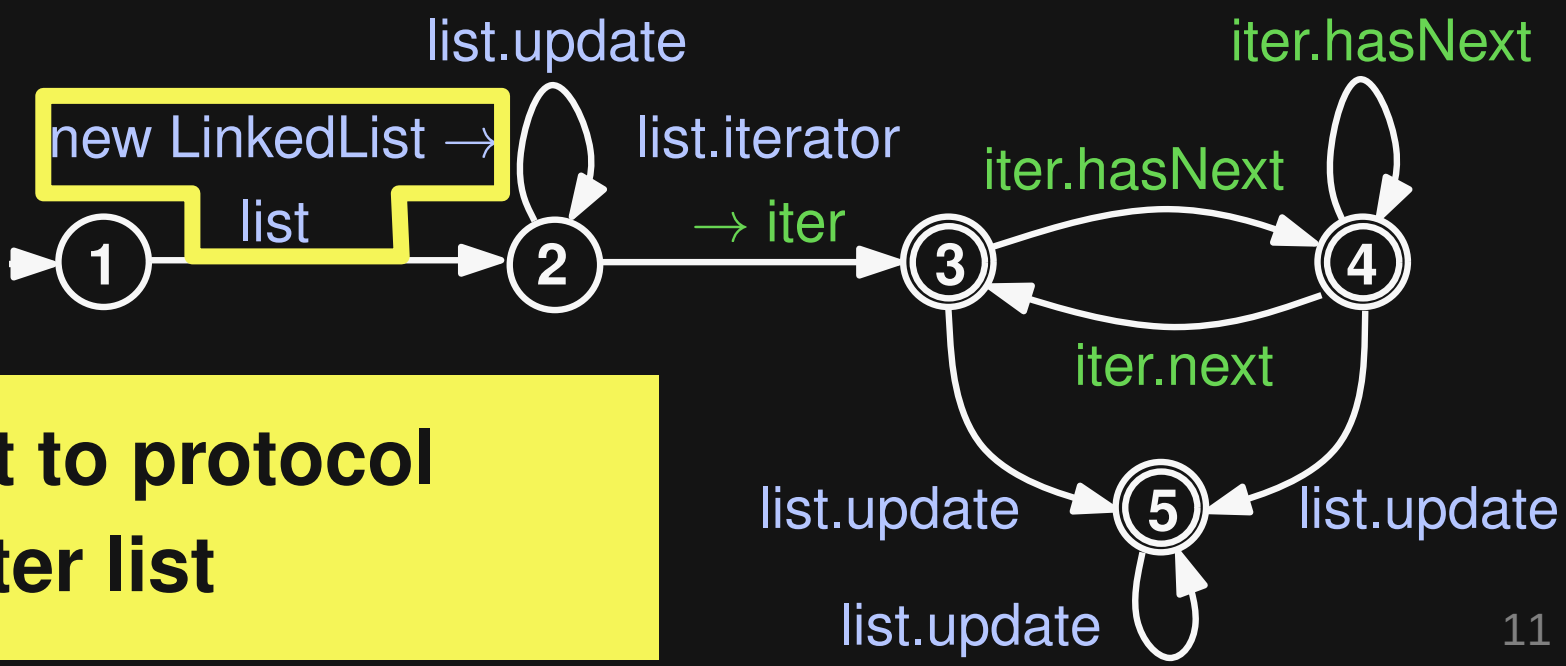


Initially:

All objects in state 1

Example: Checking

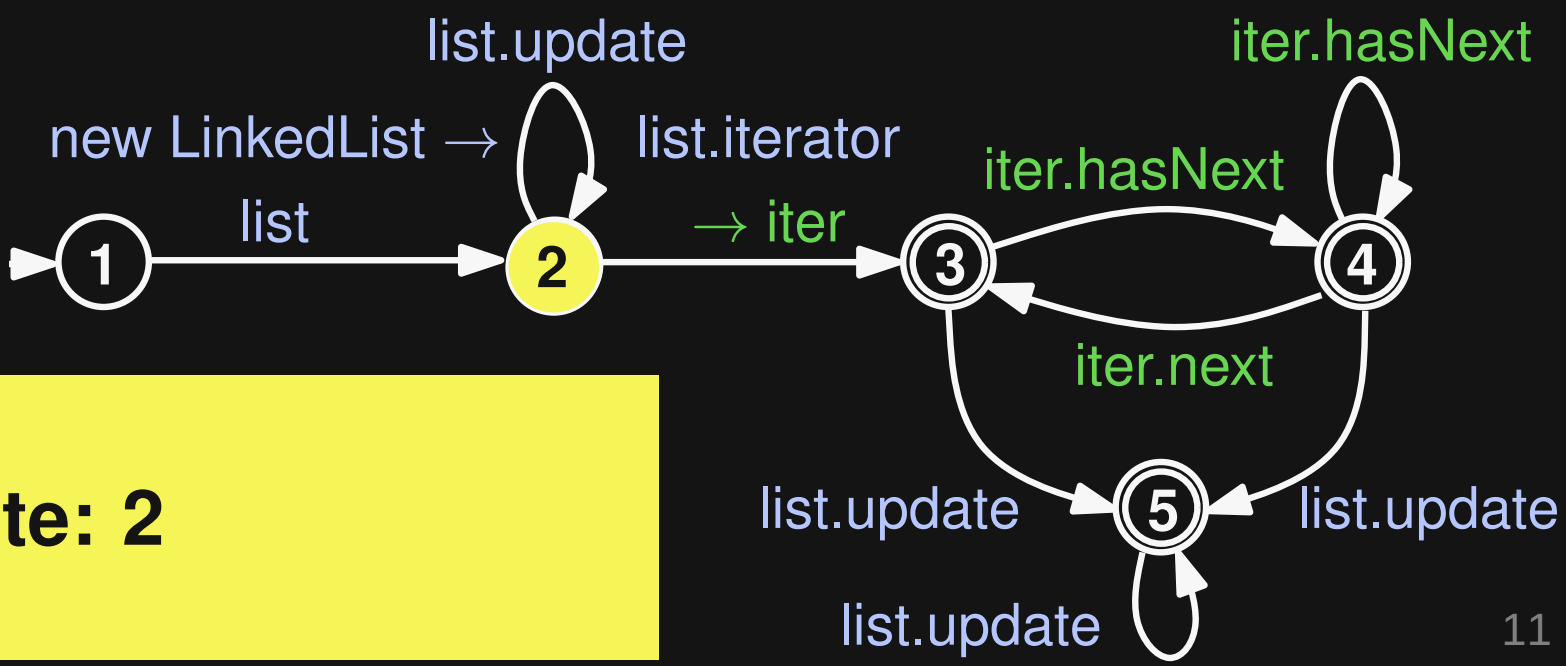
```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



Bind list to protocol parameter list

Example: Checking

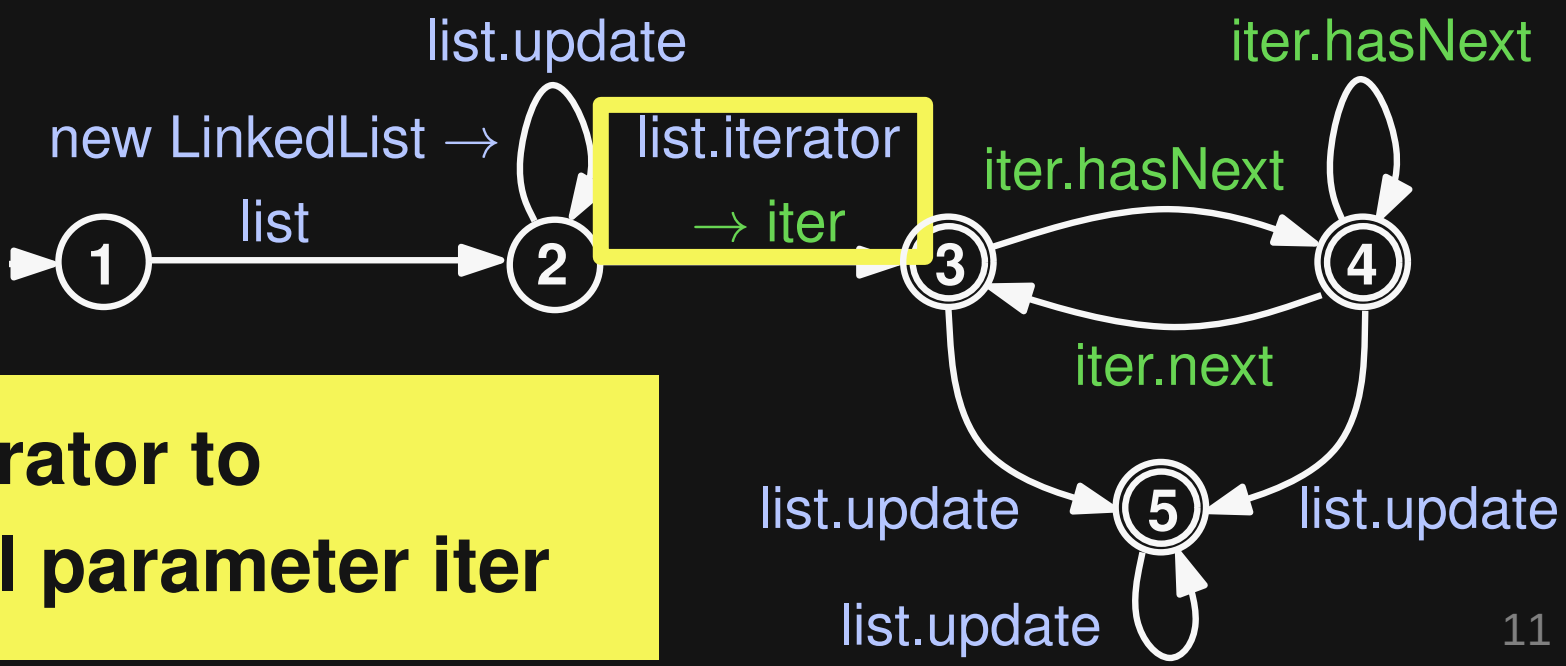
```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



New state: 2

Example: Checking

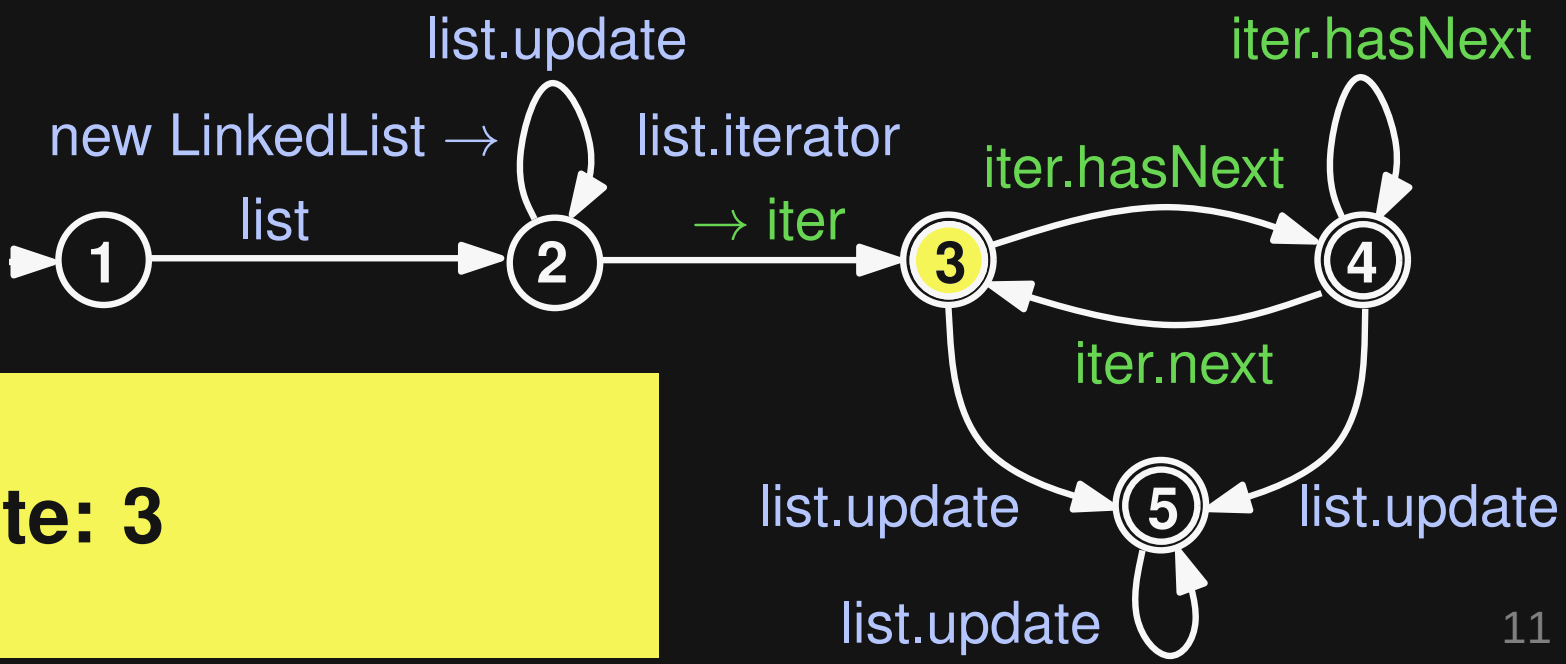
```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



Bind iterator to protocol parameter iter

Example: Checking

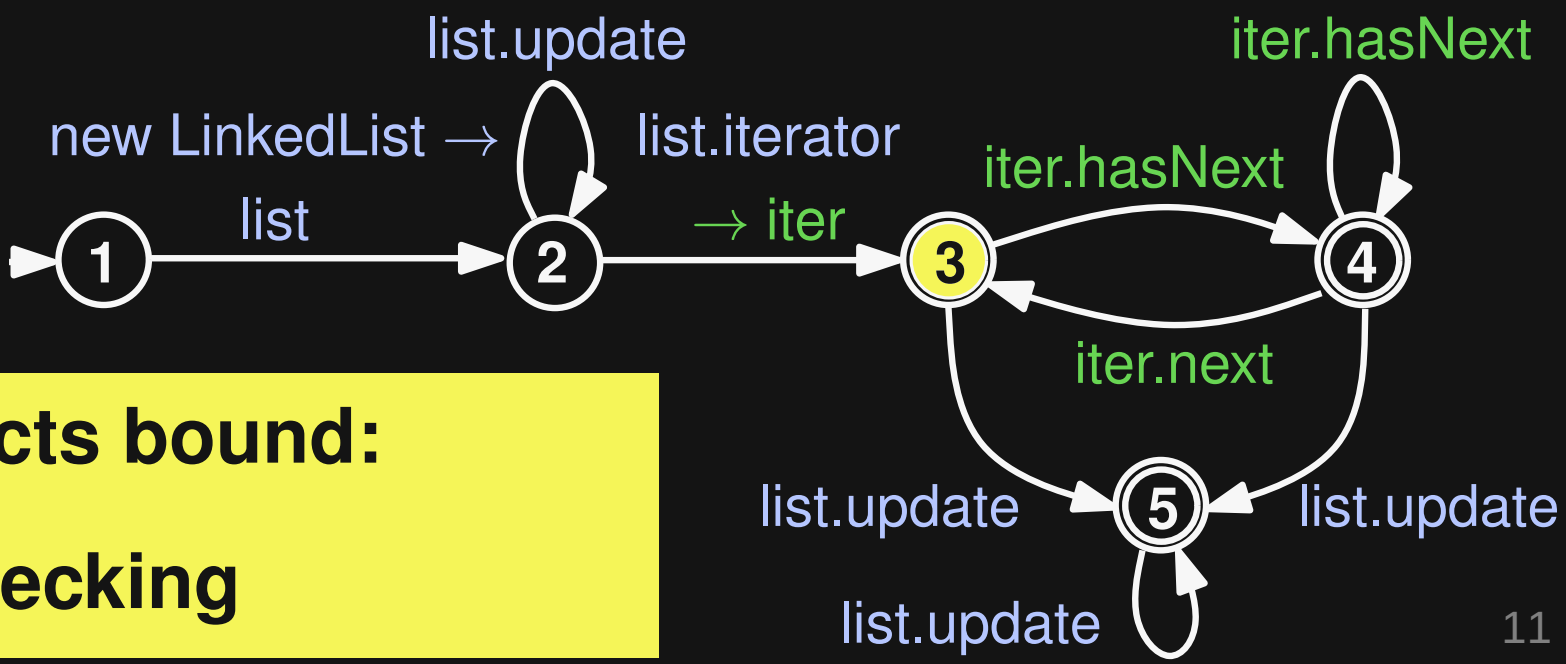
```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



New state: 3

Example: Checking

```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```

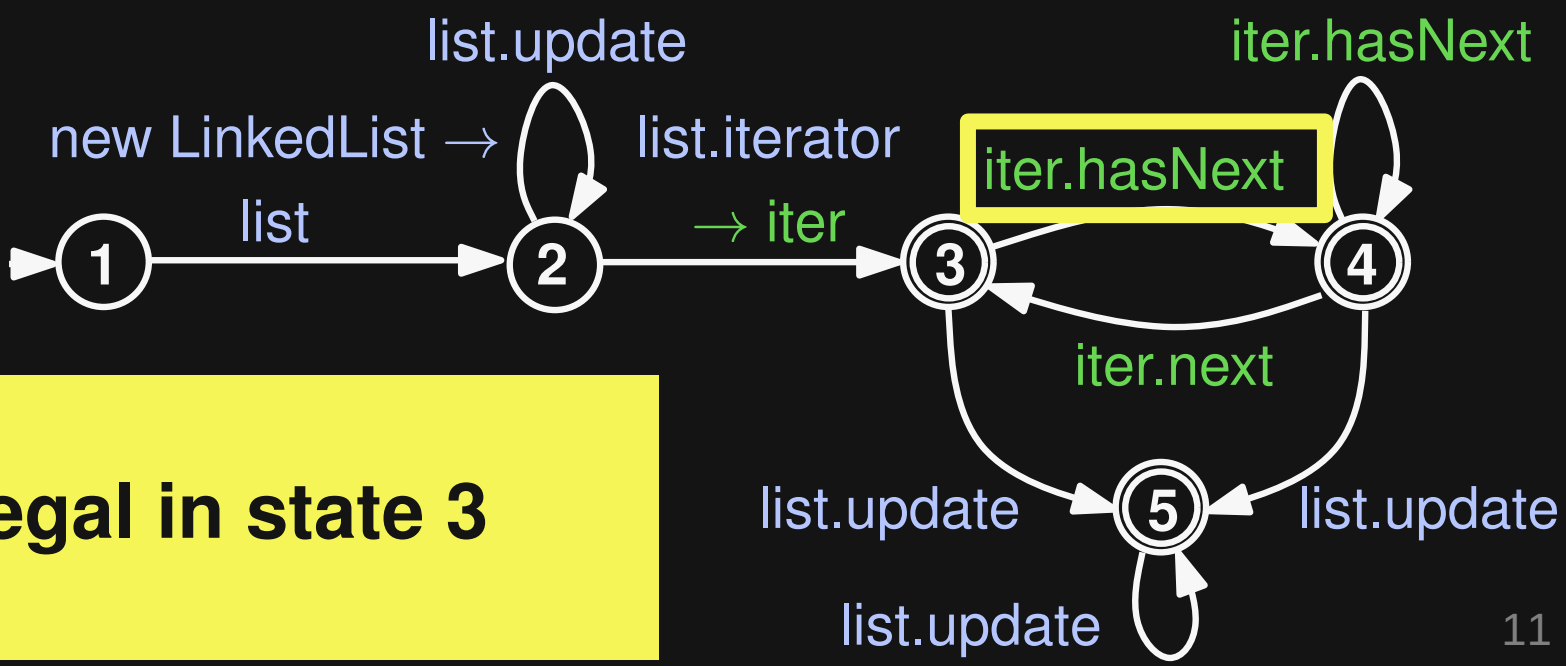


All objects bound:

Start checking

Example: Checking

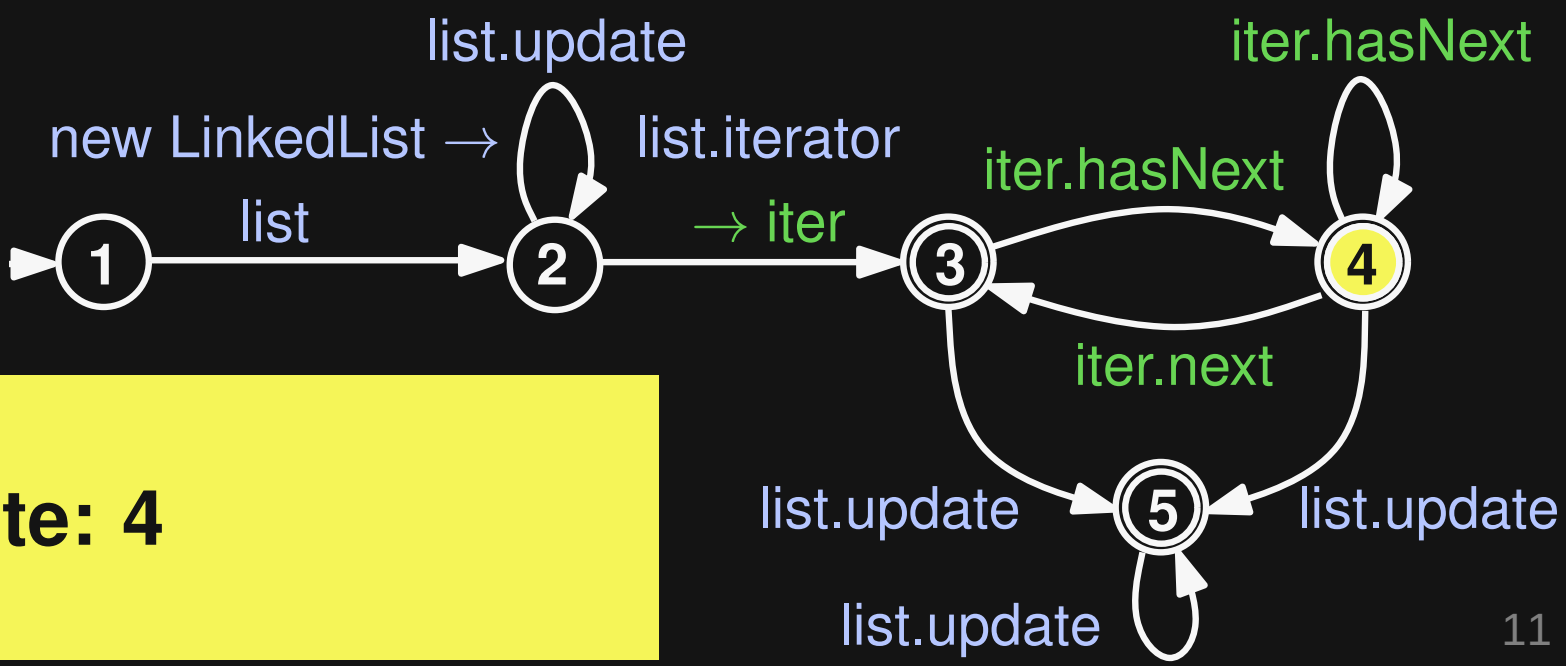
```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



Call is legal in state 3

Example: Checking

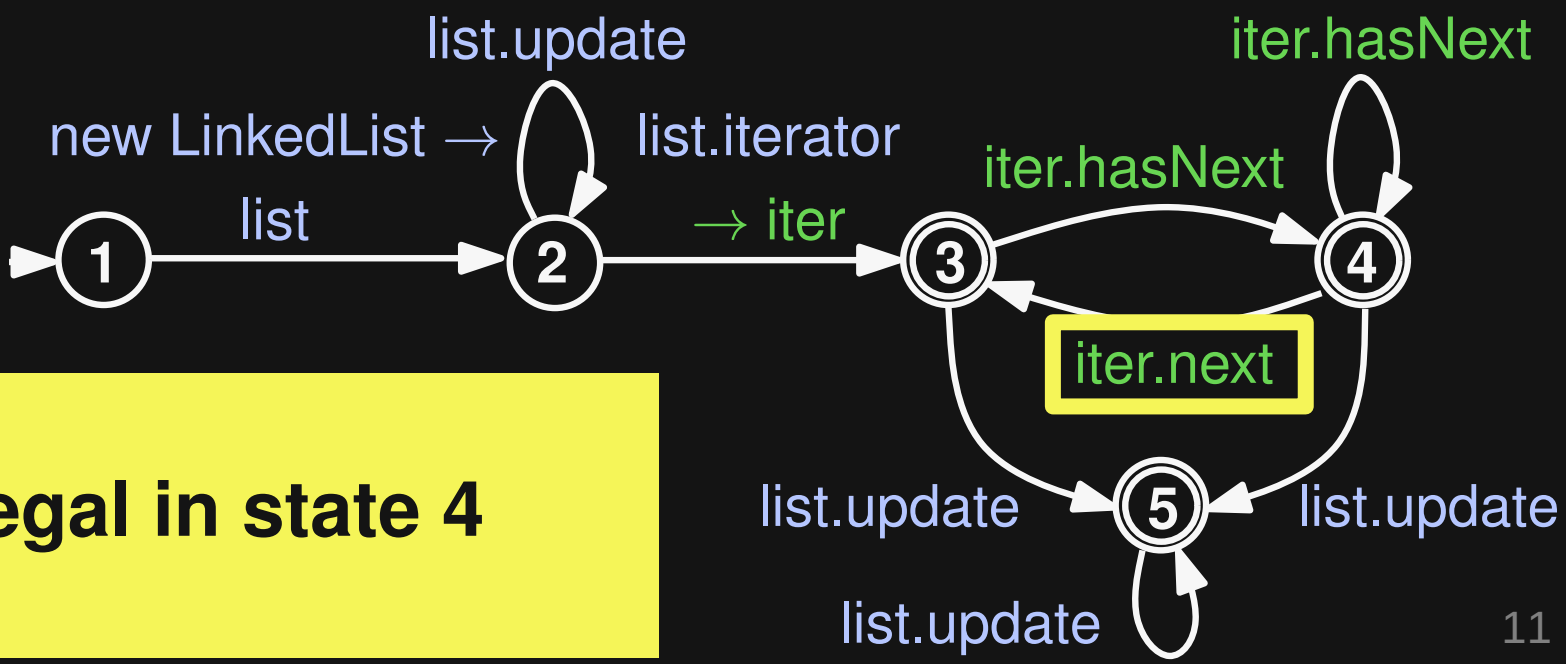
```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



New state: 4

Example: Checking

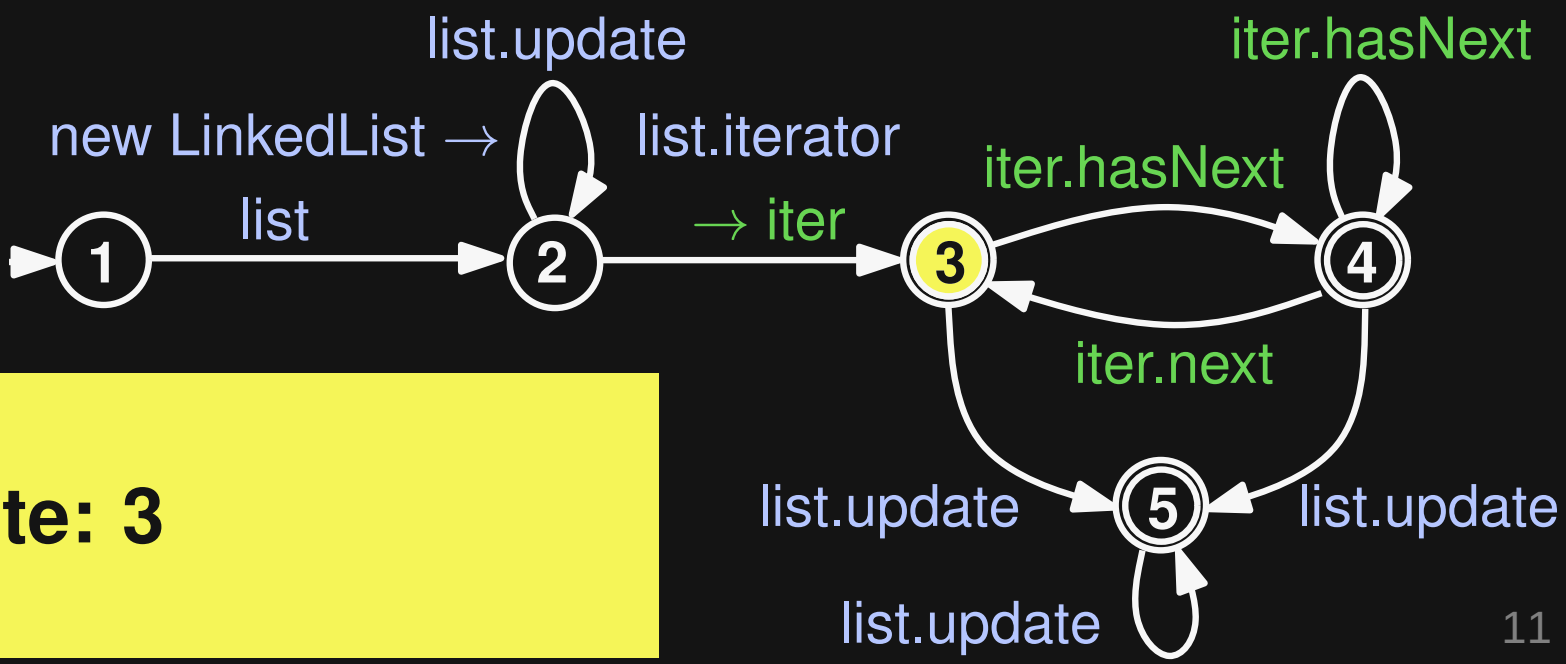
```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



Call is legal in state 4

Example: Checking

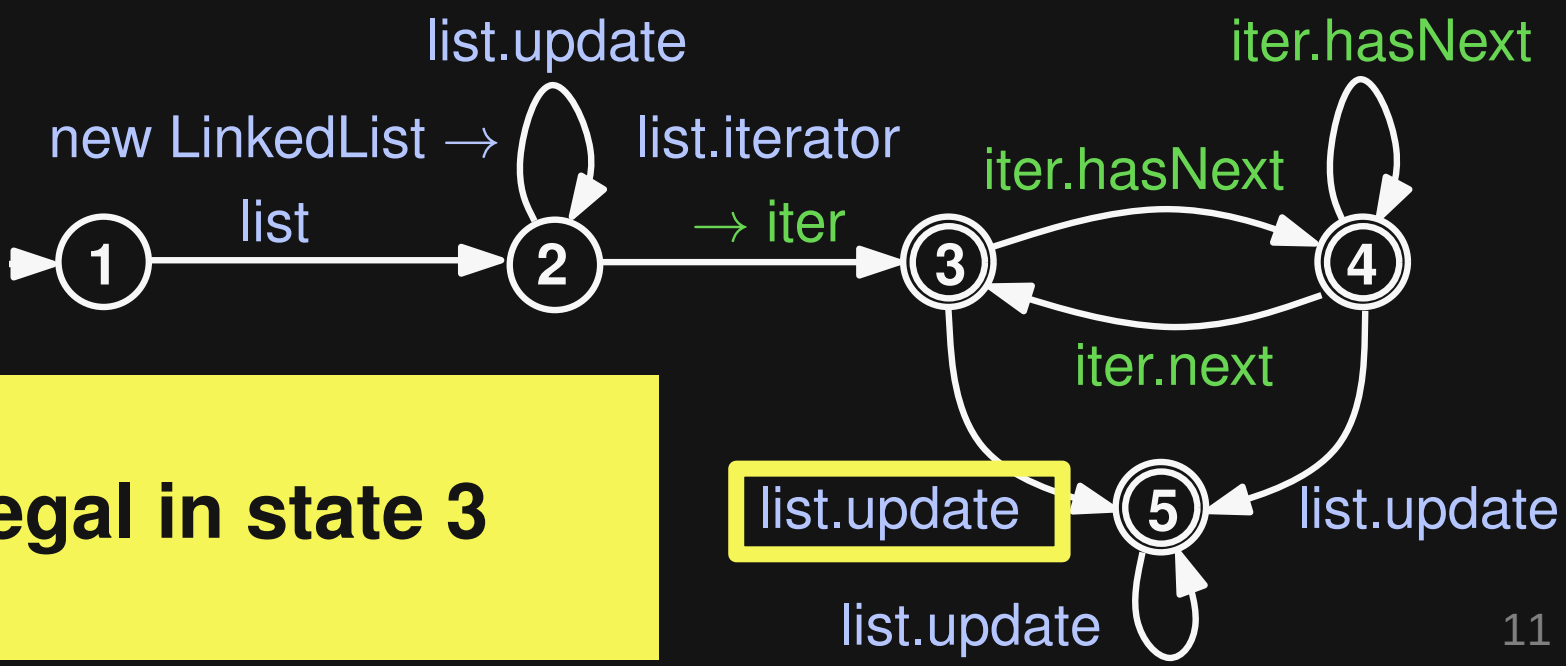
```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



New state: 3

Example: Checking

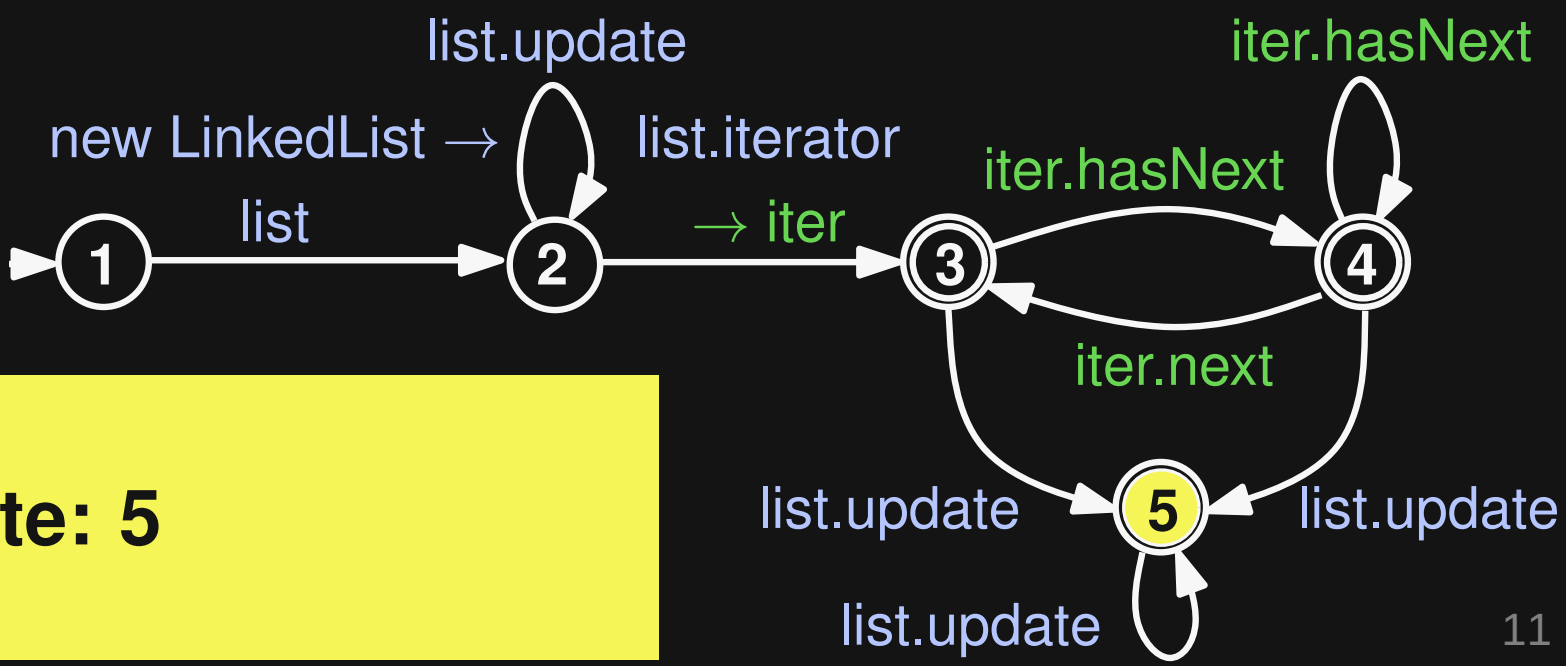
```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



Call is legal in state 3

Example: Checking

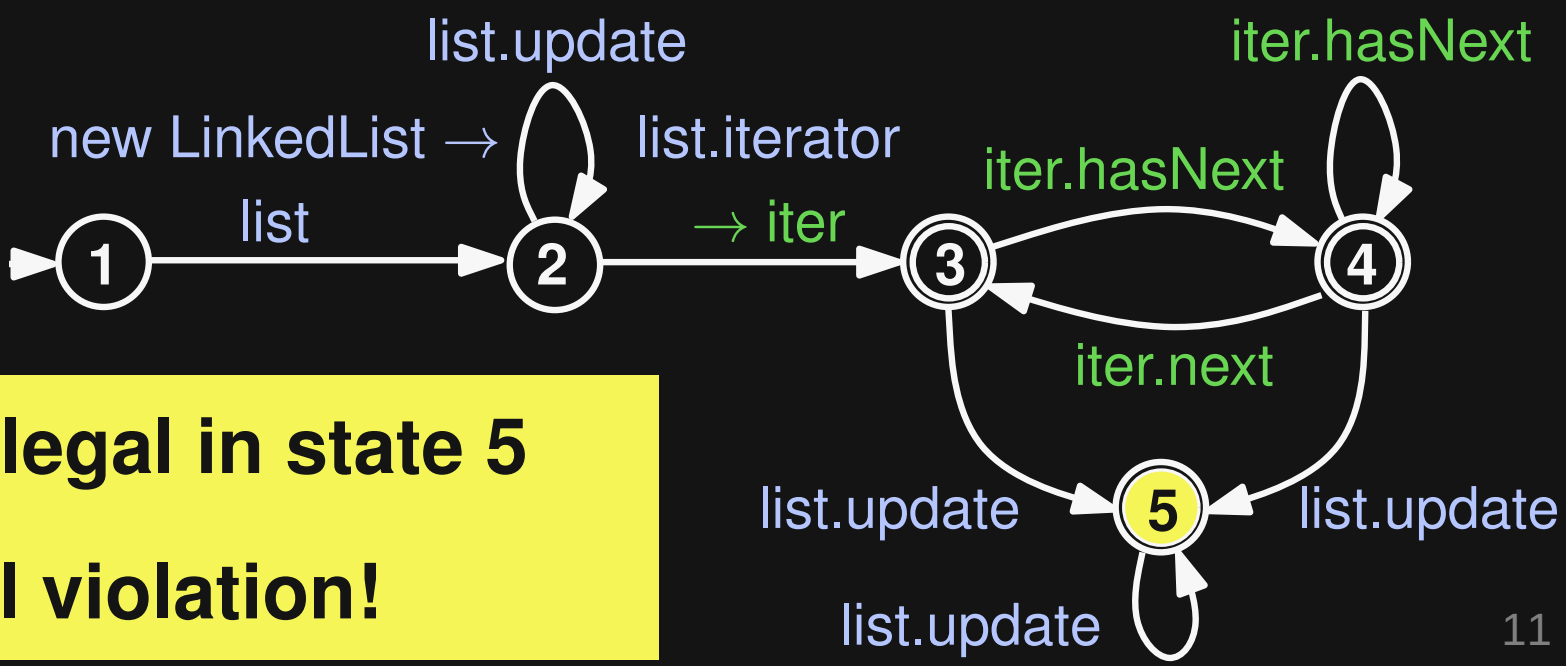
```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



New state: 5

Example: Checking

```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```



Call is illegal in state 5

Protocol violation!

Multi-Object: Essential (1)

```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```

Multi-Object: Essential (1)

```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```

Only LinkedList: No warning

Multi-Object: Essential (1)

```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```

Only Iterator: No warning

Multi-Object: Essential (1)

```
LinkedList pinConnections = new LinkedList(...);
Iterator i = pinConnections.iterator();
while (i.hasNext()) {
  PinLink curr = (PinLink) i.next();
  if (...) {
    pinConnections.remove(curr);
  }
}
```

Need LinkedList and Iterator to find bug

Multi-Object: Essential (2)

```
LinkedList l = new LinkedList(...);
Iterator i = l.iterator();
if (l.size() > 0) {
   .. = i.next();
}
```

Multi-Object: Essential (2)

```
LinkedList l = new LinkedList(...);
Iterator i = l.iterator();
if (l.size() > 0) {
   .. = i.next();
}
```

Only Iterator: False warning

Multi-Object: Essential (2)

```
LinkedList l = new LinkedList(...);
Iterator i = l.iterator();
if (l.size() > 0) {
   .. = i.next();
}
```

LinkedList and Iterator: No warning

Multi-Object Protocols

Benefits of multi-object protocols

1. **Find bugs missed by single-object protocols**

2. **Reduce false positives**

Implementation

Automatic tool to check Java programs

- Currently, two object-protocols only
- Prototype: Not tuned for performance
- 2,500 LoC (Scala)
  + existing miner and checker

Evaluation

Effective in finding protocol violations

Importance of multi-object protocols

Comparison with existing approaches

Setup

APIs: `java.*` and `javax.*`

Protocol mining

- 223 automatically mined protocols
- 146 types, 26 packages

Checking

- DaCapo benchmarks (1.6 MLoC)
- Warning classification:
  Bug — code smell — false positive

Setup

APIs: `java.*` and `javax.*`

Protocol mining

- 223 automatically mined protocols
- 146 types, 26 packages

Checking

- DaCapo benchmarks (1.6 MLoC)
- Warning classification:
  Bug — code smell — false positive

Setup

APIs: `java.*` and `javax.*`

Protocol mining

- 223 automatically mined protocols
- 146 types, 26 packages

Checking

- DaCapo benchmarks (1.6 MLoC)
- Warning classification:
  Bug — code smell — false positive

Setup

APIs: `java.*` and `javax.*`

Protocol mining

- 223 automatically mined protocols
- 146 types, 26 packages

Checking

- DaCapo benchmarks (1.6 MLoC)
- Warning classification:

  Bug — code smell — false positive

Example: Bug

```
Map comparators = ...
Iterator i = comparators.values().iterator();
for (Comparator c = (Comparator) i.next();
     c != null;
     c = (Comparator) i.next()) { ... }
```

(from Avrora)

Example: Bug

```
Map comparators = ...
Iterator i = comparators.values().iterator();
for (Comparator c = (Comparator) i.next();
     c != null;
     c = (Comparator) i.next()) { ... }
```

Illegal iterator usage

(from Avrora)

Example: Code Smell

```
BufferedReader in = null;
try {
  in = new BufferedReader(...);
  ...
  in.close();
} finally {
  if (in != null) {
    try { in.close(); }
    catch (IOException e) { ... }
  }
}
```

(from PMD)

Example: Code Smell

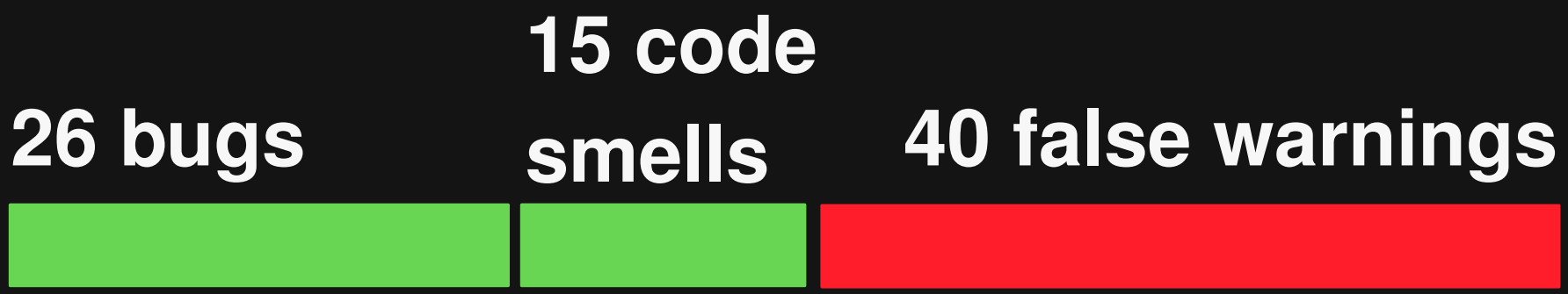
```
BufferedReader in = null;
try {
   in = new BufferedReader(...);
   ...
   in.close();
} finally {
   if (in != null) {
      try { in.close(); }
      catch (IOException e) { ... }
   }
}
```

Duplicate close

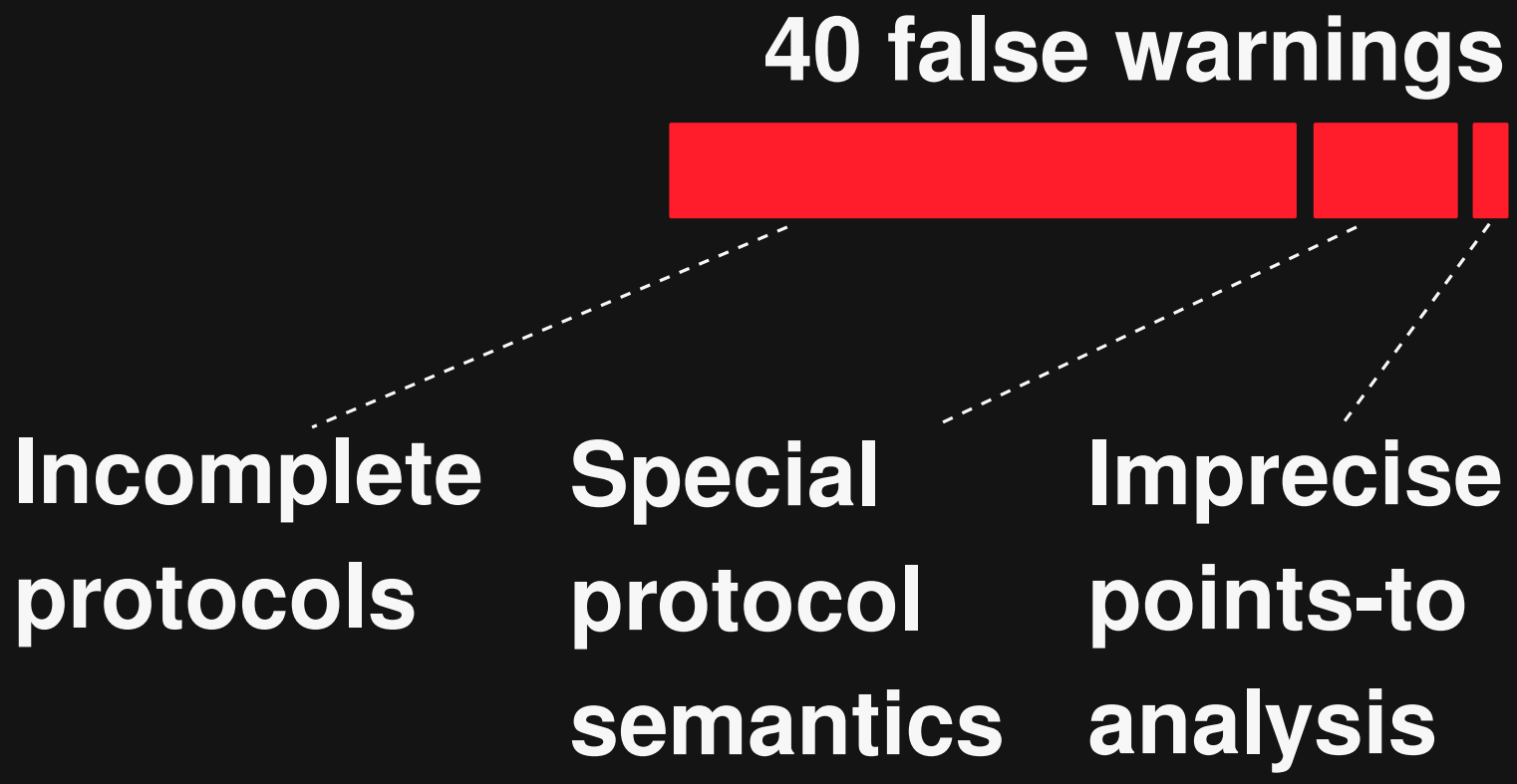
(from PMD)

Precision

81 reported warnings, 51% true positives



Precision


40 false warnings
Incomplete protocols
Special protocol semantics
Imprecise points-to analysis

Which Bugs Do We Miss?

50 randomly seeded protocol bugs

Find 35 of them (70%)

15 missed bugs:

- Conservatism of static checker (6)
- Protocol too permissive (5)
- Protocol not expressive enough (4)

Multiple Objects

Are multi-object protocols important?

- **Protocols:**
  61% involve multiple objects
- **Violations:**
  44% only found with multi-object protocol

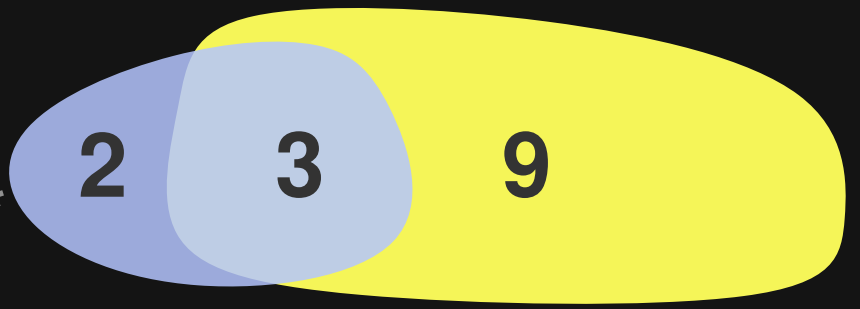
Comparison to Prior Work

Bugs found:

Nguyen et al., ESEC/FSE'09 *

Our work

Wasylkowski +Zeller, ASE'09 *



Our work

* `java.*` and `javax.*` bugs only

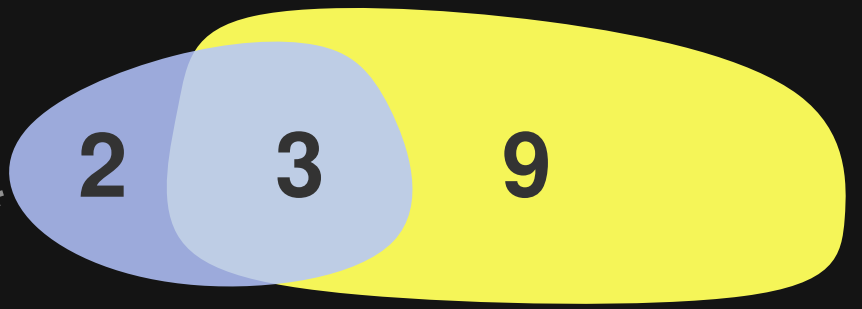
Comparison to Prior Work

Bugs found:

Nguyen et al., ESEC/FSE'09 *

Our work

Wasylkowski +Zeller, ASE'09 *



Our work

* `java.*` and `javax.*` bugs only

Why?

- Missing calls vs. incorrect calls
- Must infer the "right" specification

Conclusion

Mined protocols:
Good basis for automatic bug finding

Multi-object: Important in practice

API usage: (Still) a difficult problem

Thank you!

Statically Checking API Protocol Conformance with Mined Multi-Object Specifications

**Michael Pradel, Ciera Jaspan,
Jonathan Aldrich, and Thomas R. Gross**

**Bugs and protocols for download:
http://mp.binaervarianz.de/icse2012-statically/**