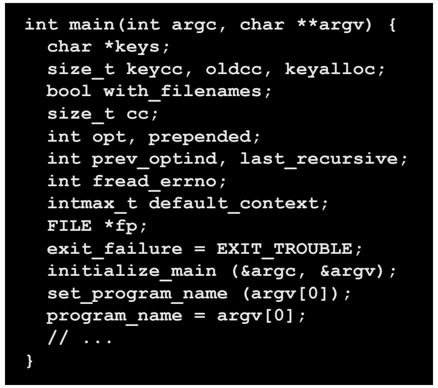
Active Learning of Points-To Specifications

Osbert Bastani, Rahul Sharma, Alex Aiken, Percy Liang



```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  bool with_filenames;
  size_t cc;
  int opt, prepended;
  int prev_optind, last_recursive;
  int fread_errno;
  intmax_t default_context;
  FILE *fp;
  exit_failure = EXIT_TROUBLE;
  initialize_main (&argc, &argv);
  set_program_name (argv[0]);
  program_name = argv[0];
  // ...
}
```

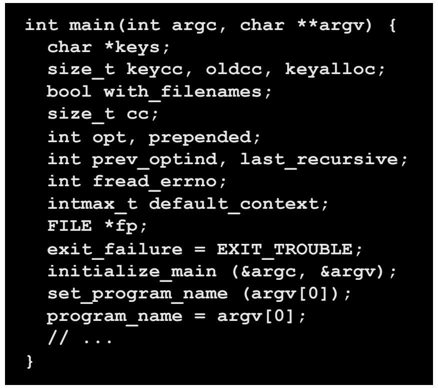
Android app

security analyst

malware?

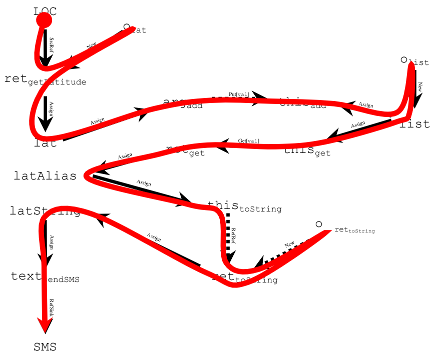
Find **malicious behaviors** using source to sink **taint flows**

Information leak:	location	flows to	Internet
SMS Fraud:	phone #	used in	SMS send
Ransomware:	network packets	encrypt	files



```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  bool with_filenames;
  size_t cc;
  int opt, prepended;
  int prev_optind, last_recursive;
  int fread_errno;
  intmax_t default_context;
  FILE *fp;
  exit_failure = EXIT_TROUBLE;
  initialize_main (&argc, &argv);
  set_program_name (argv[0]);
  program_name = argv[0];
  // ...
}
```

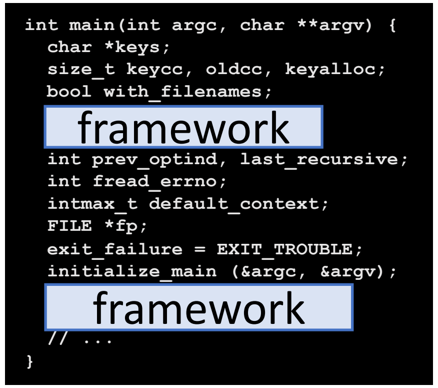
Android app



taint analysis

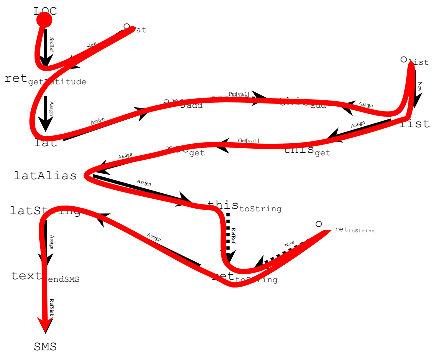
location → Internet
SMS → Internet

malicious behaviors



```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  bool with_filenames;
  framework
  int prev_optind, last_recursive;
  int fread_errno;
  intmax_t default_context;
  FILE *fp;
  exit_failure = EXIT_TROUBLE;
  initialize_main (&argc, &argv);
  framework
  // ...
}
```

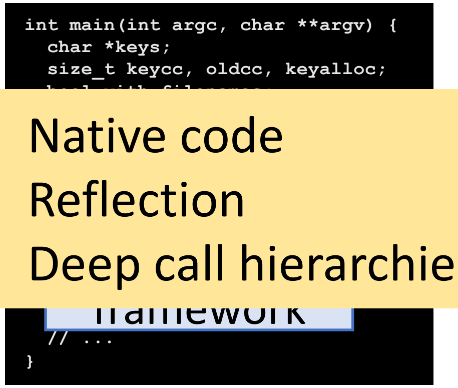
Android app



taint analysis

location → Internet
SMS → Internet

malicious behaviors



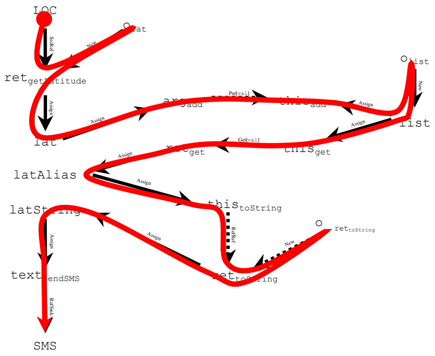
```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
```

framework

```
  // ...
}
```

- Native code
- Reflection
- Deep call hierarchies

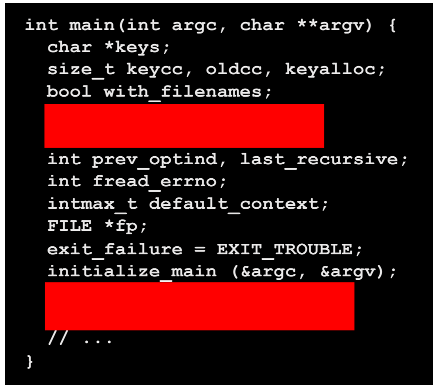
Android app



taint analysis

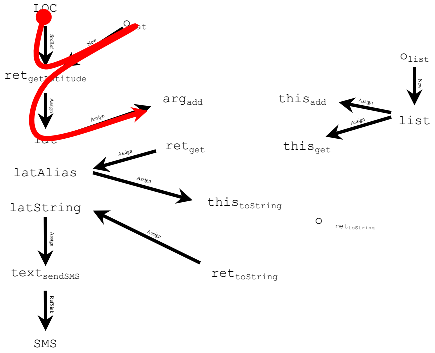
location → Internet
SMS → Internet

malicious behaviors



```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  bool with_filenames;
  [redacted]
  int prev_optind, last_recursive;
  int fread_errno;
  intmax_t default_context;
  FILE *fp;
  exit_failure = EXIT_TROUBLE;
  initialize_main (&argc, &argv);
  [redacted]
  // ...
}
```

Android app

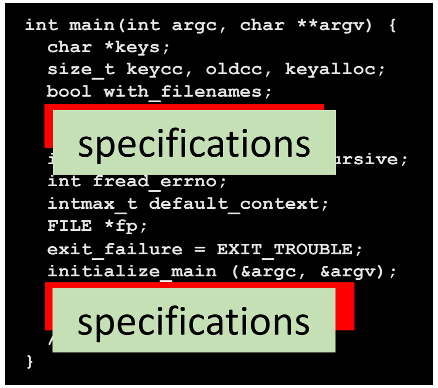


LOC
o_{lat}
SrcRef
New
$\mathrm{ret}_{\mathrm{getLatitude}}$
Assign
lat
Assign
$\mathrm{arg}_{\mathrm{add}}$
$\mathrm{ret}_{\mathrm{get}}$
Assign
latAlias
Assign
$\mathrm{this}_{\mathrm{toString}}$
latString
Assign
$\mathrm{ret}_{\mathrm{toString}}$
Assign
$\mathrm{text}_{\mathrm{sendSMS}}$
RefSink
SMS
o_{list}
New
list
Assign
$\mathrm{this}_{\mathrm{add}}$
Assign
$\mathrm{this}_{\mathrm{get}}$
o
$\mathrm{ret}_{\mathrm{toString}}$

taint analysis

?

malicious behaviors

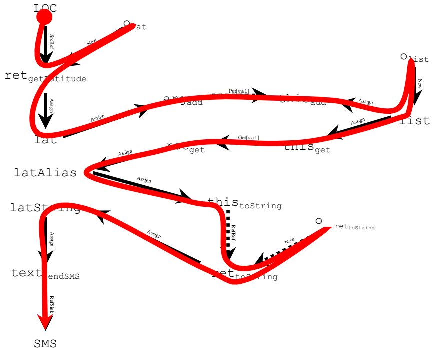


```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  bool with_filenames;
  
  i                               ursive;
  int fread_errno;
  intmax_t default_context;
  FILE *fp;
  exit_failure = EXIT_TROUBLE;
  initialize_main (&argc, &argv);
  
  /
}
```

specifications

specifications

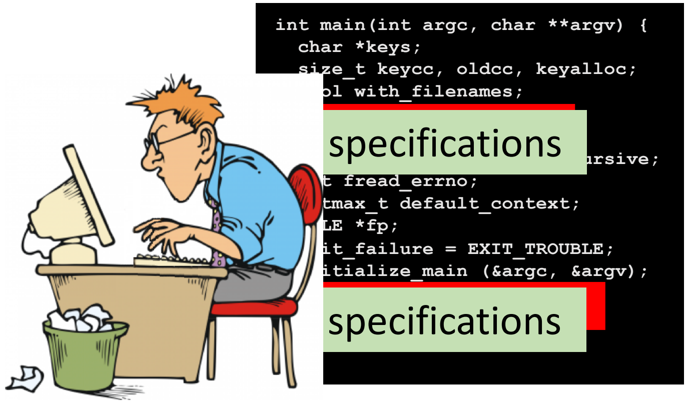
Android app



taint analysis

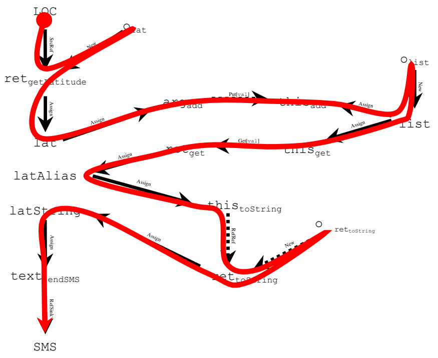
location → Internet
SMS → Internet

malicious behaviors



```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  ol with_filenames;
                                  rsive;
  t fread_errno;
  tmax_t default_context;
  LE *fp;
  it_failure = EXIT_TROUBLE;
  itialize_main (&argc, &argv);
```

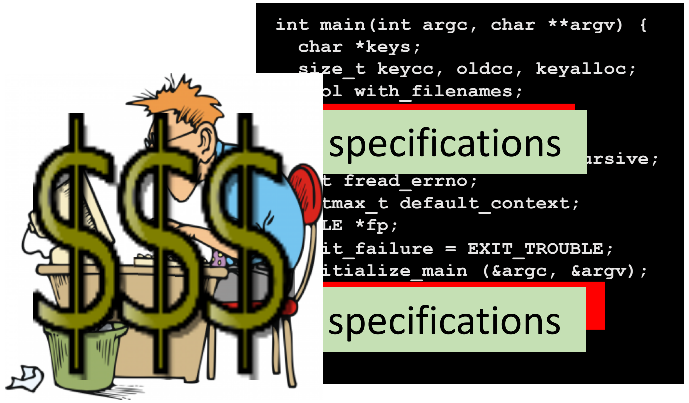
Android app



taint analysis

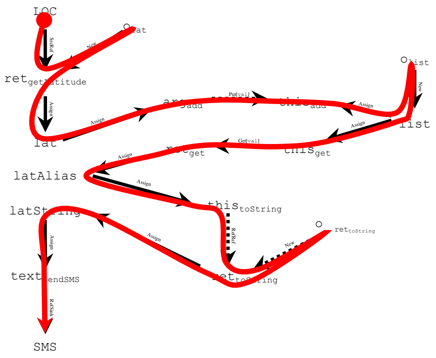
location → Internet
SMS → Internet

malicious behaviors



```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  ol with_filenames;
                              ursive;
t fread_errno;
tmax_t default_context;
LE *fp;
it_failure = EXIT_TROUBLE;
itialize_main (&argc, &argv);
```

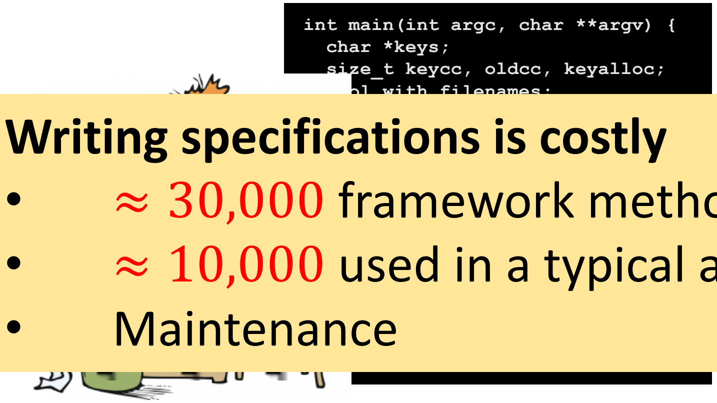
Android app



taint analysis

location → Internet
SMS → Internet

malicious behaviors

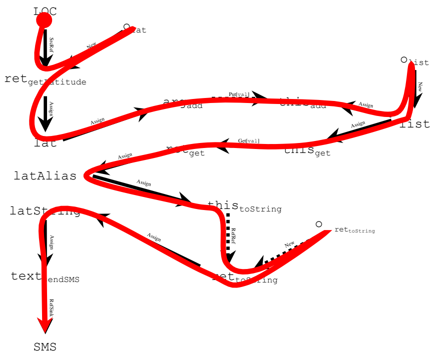


```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
   ol with filenames;
```

Writing specifications is costly

- ≈ 30,000 framework methods
- ≈ 10,000 used in a typical app
- Maintenance

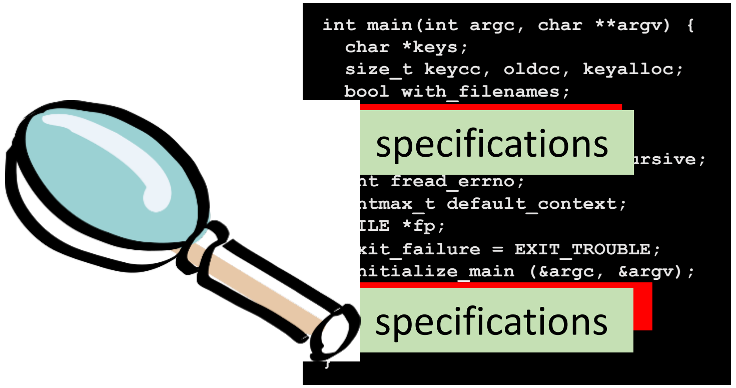
Android app



taint analysis

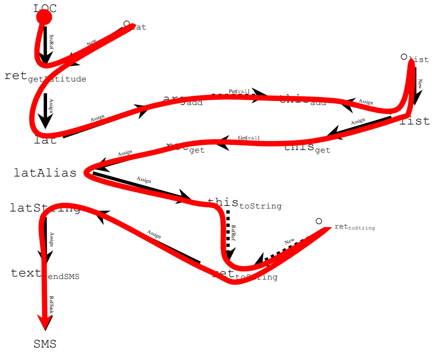
location → Internet
SMS → Internet

malicious behaviors



```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  bool with_filenames;
                                 ursive;
 nt fread_errno;
 ntmax_t default_context;
 ILE *fp;
 xit_failure = EXIT_TROUBLE;
 nitialize_main (&argc, &argv);
}
```

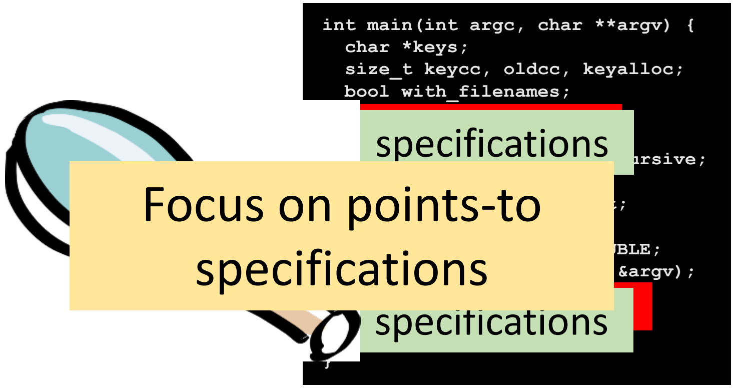
Android app



taint analysis

location → Internet
SMS → Internet

malicious behaviors



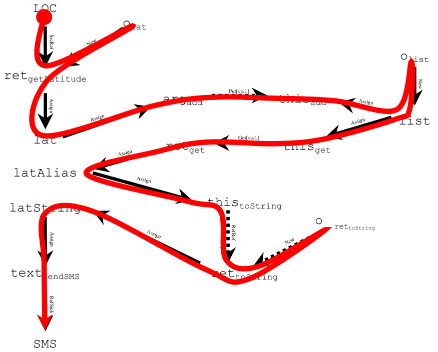
```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  bool with_filenames;
```

specifications

Focus on points-to specifications

specifications

Android app



taint analysis

location → Internet
SMS → Internet

malicious behaviors

Roadmap

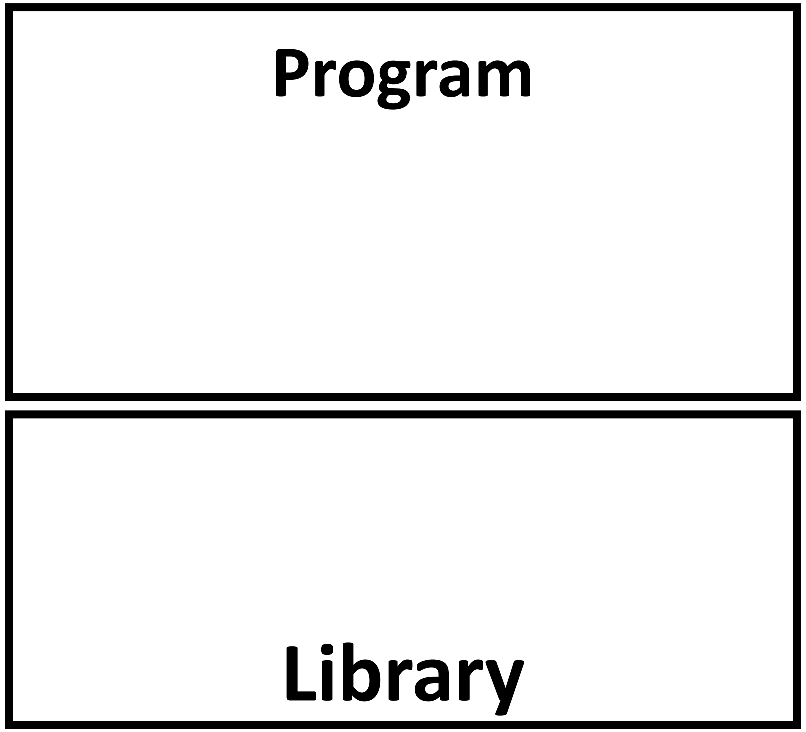
- Points-to analysis
- Path specifications
- Inference algorithm
- Evaluation

Roadmap

- **Points-to analysis**
- Path specifications
- Inference algorithm
- Evaluation

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```

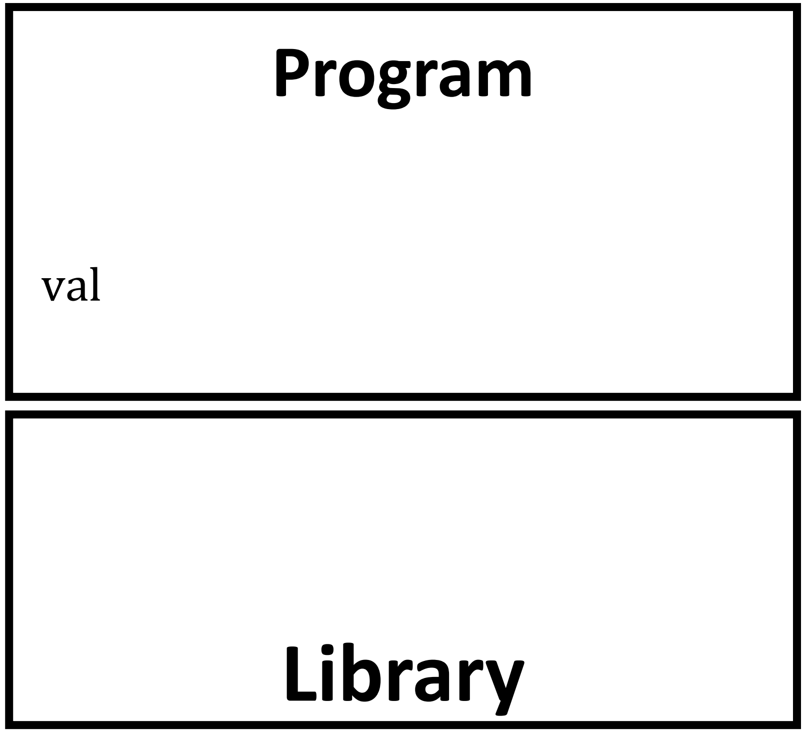


Program

Library

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```



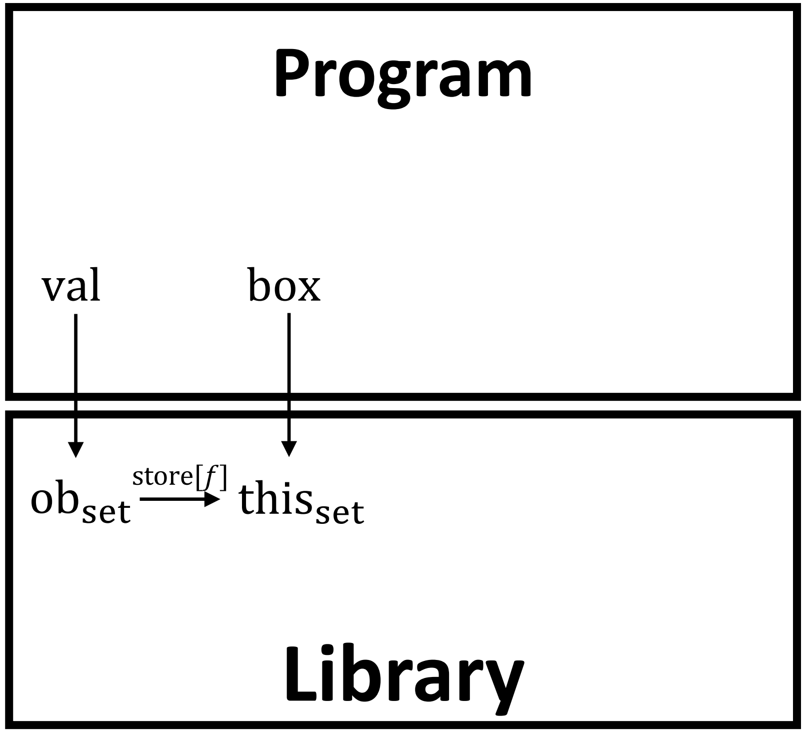
Program

val

Library

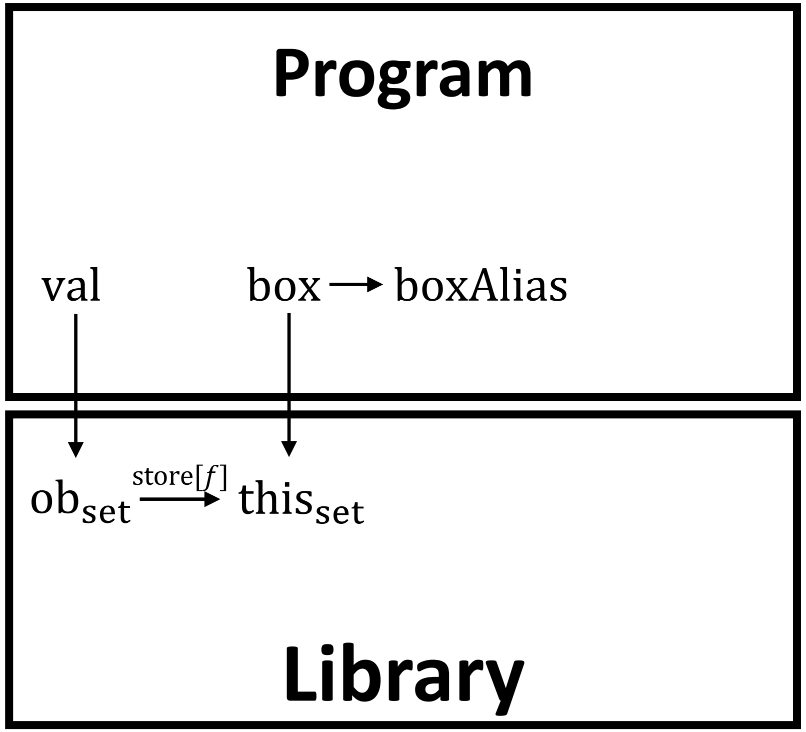
Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```


Program
val
box
$\mathrm{ob}_{\mathrm{set}} \xrightarrow{\mathrm{store}[f]} \mathrm{this}_{\mathrm{set}}$
Library

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();


class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```



Program

val

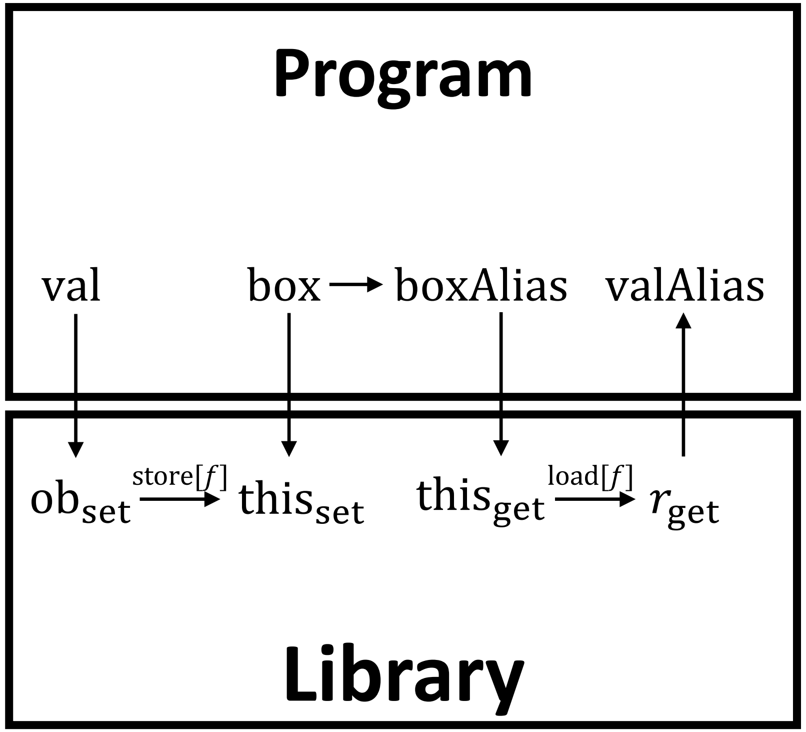
box → boxAlias

$\mathrm{ob}_{\mathrm{set}} \xrightarrow{\mathrm{store}[f]} \mathrm{this}_{\mathrm{set}}$

Library

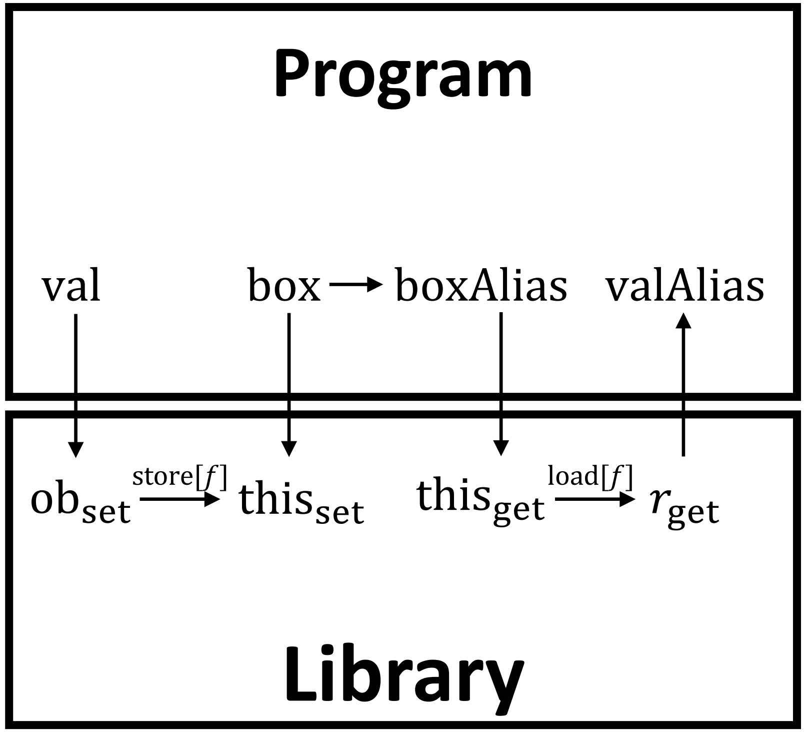
Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```


Program
val
box
boxAlias
valAlias
$\mathrm{ob}_{\mathrm{set}} \xrightarrow{\mathrm{store}[f]} \mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}} \xrightarrow{\mathrm{load}[f]} r_{\mathrm{get}}$
Library

Points-To Analysis

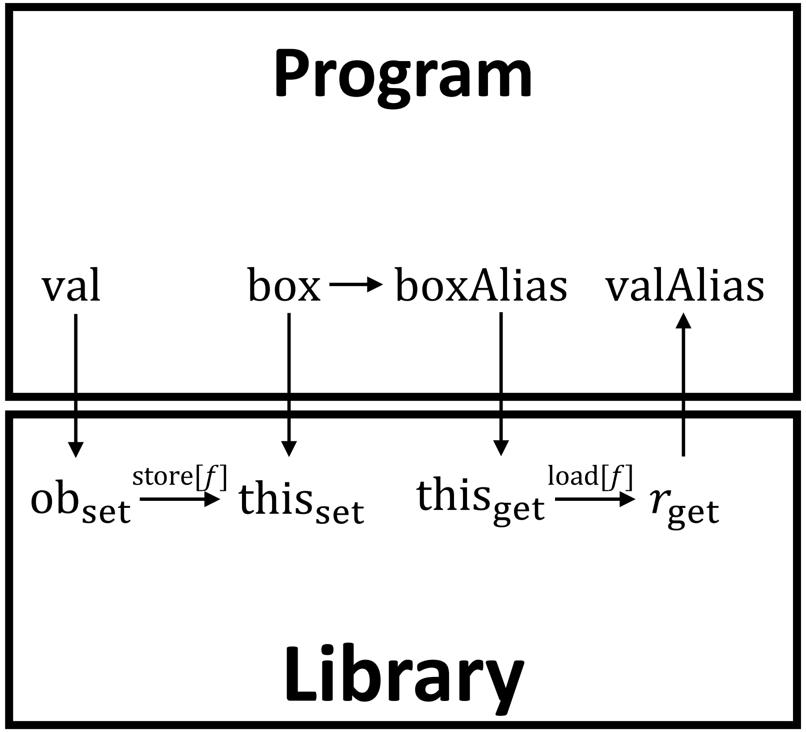
```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```


Program
val
box
boxAlias
valAlias
$\text{ob}_{\text{set}} \xrightarrow{\text{store}[f]} \text{this}_{\text{set}}$
$\text{this}_{\text{get}} \xrightarrow{\text{load}[f]} r_{\text{get}}$
Library

$\Rightarrow \qquad v \xrightarrow{\text{alias}} v$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();


class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```

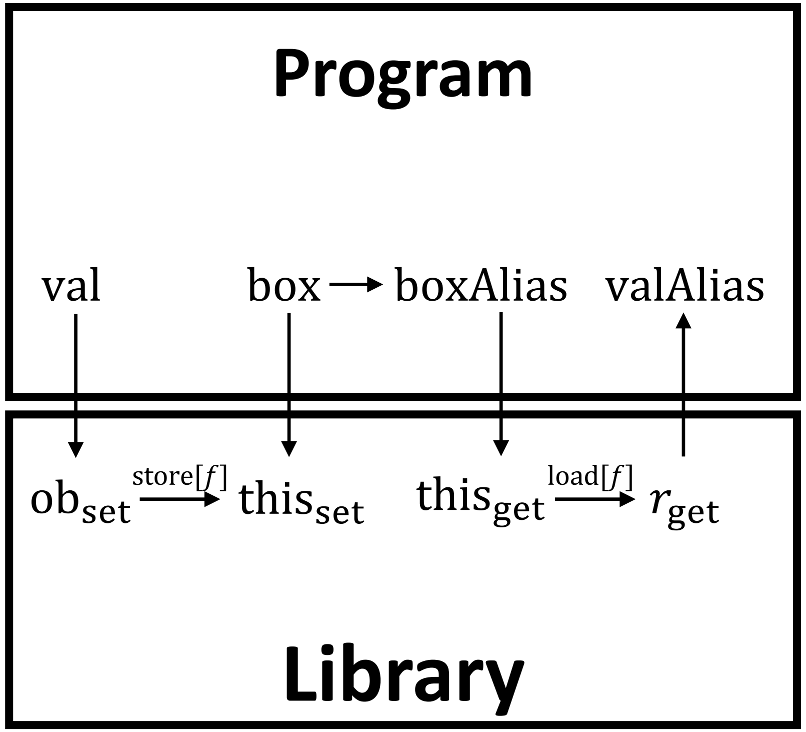


$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```



Program

val $\text{box} \to \text{boxAlias}$ valAlias

$\text{ob}_{\text{set}} \xrightarrow{\text{store}[f]} \text{this}_{\text{set}}$ $\text{this}_{\text{get}} \xrightarrow{\text{load}[f]} r_{\text{get}}$

Library

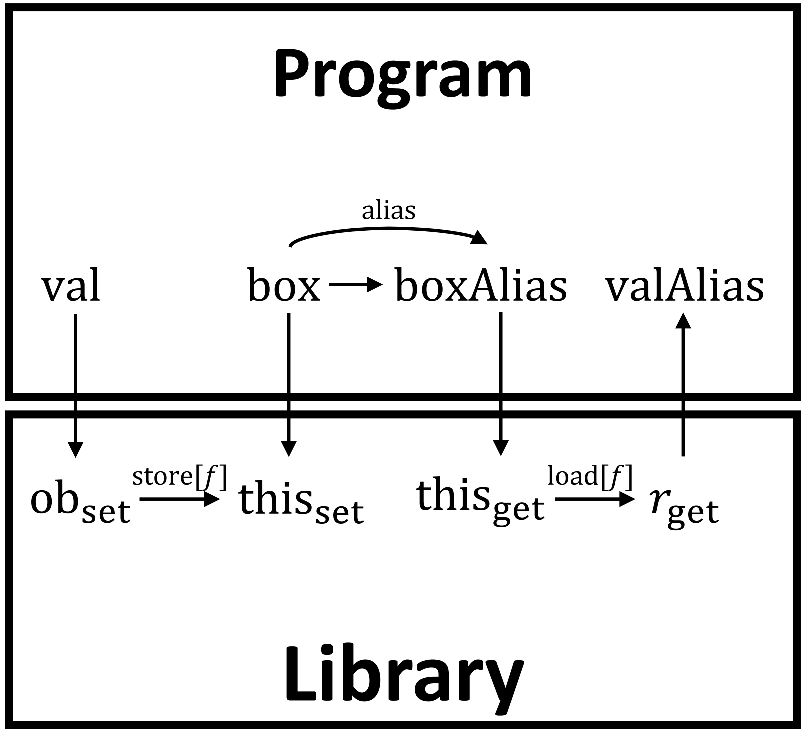
$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \to w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

Points-To Analysis

```
1.  Double val = new Double(0.0);
2.  Box box = new Box();
3.  box.set(val);
4.  Box boxAlias = box;
5.  Double valAlias = boxAlias.get();

6.  class Box:  // library
7.      Object f;
8.      void set(Object ob): f = ob;
9.      Object get(): return f;
```


Program
alias
val
box
boxAlias
valAlias
$\mathrm{ob}_{\mathrm{set}}$
store$[f]$
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
load$[f]$
r_{get}
Library

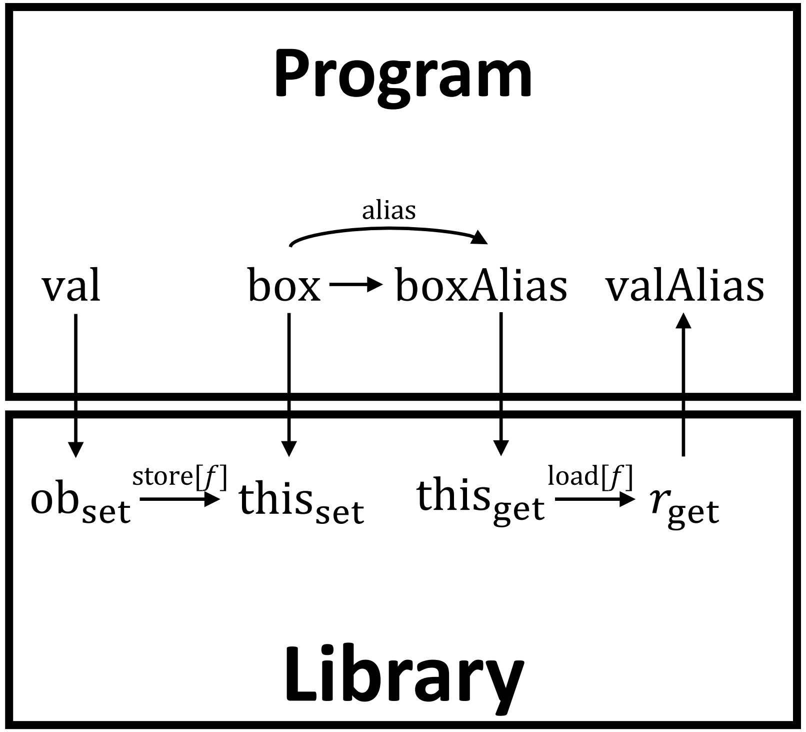
$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \to w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```



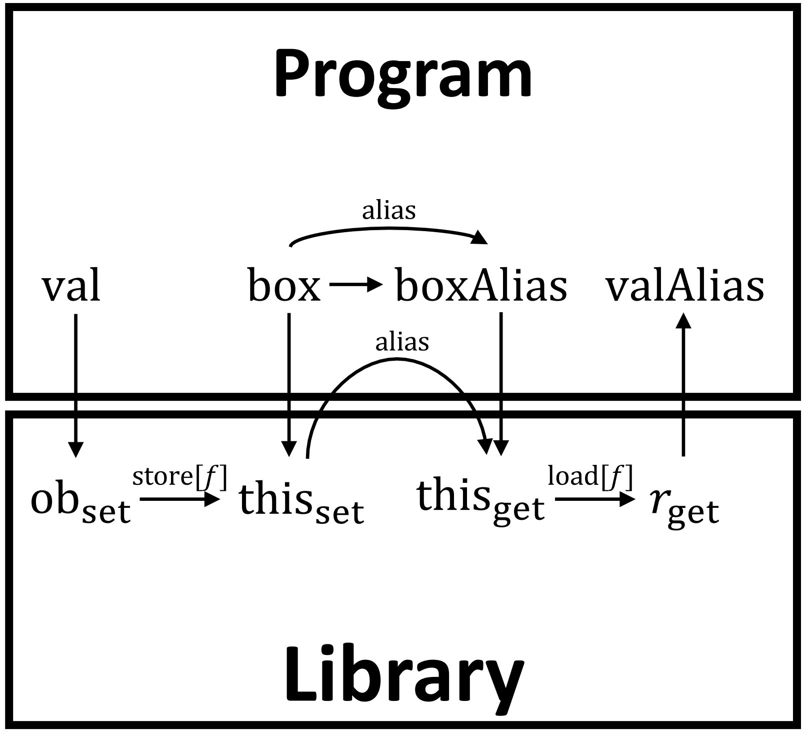
Program
alias
val
box
boxAlias
valAlias
$\mathrm{ob}_{\mathrm{set}}$
store[f]
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
load[f]
r_{get}
Library

$$\Rightarrow \quad v \xrightarrow{\text{alias}} v \qquad\qquad u \xrightarrow{\text{alias}} v \to w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w \qquad\qquad u \xrightarrow{\text{alias}} v \leftarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```


Program
alias
val
box
boxAlias
valAlias
alias
$\mathrm{ob}_{\mathrm{set}}$
store[f]
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
load[f]
r_{get}
Library

$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

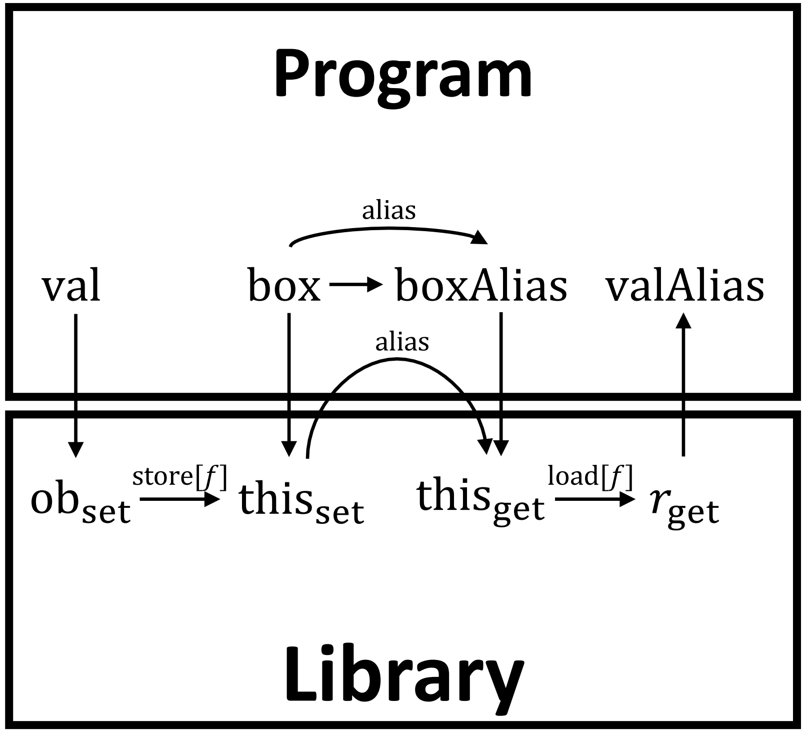
$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \rightarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \leftarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```



$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

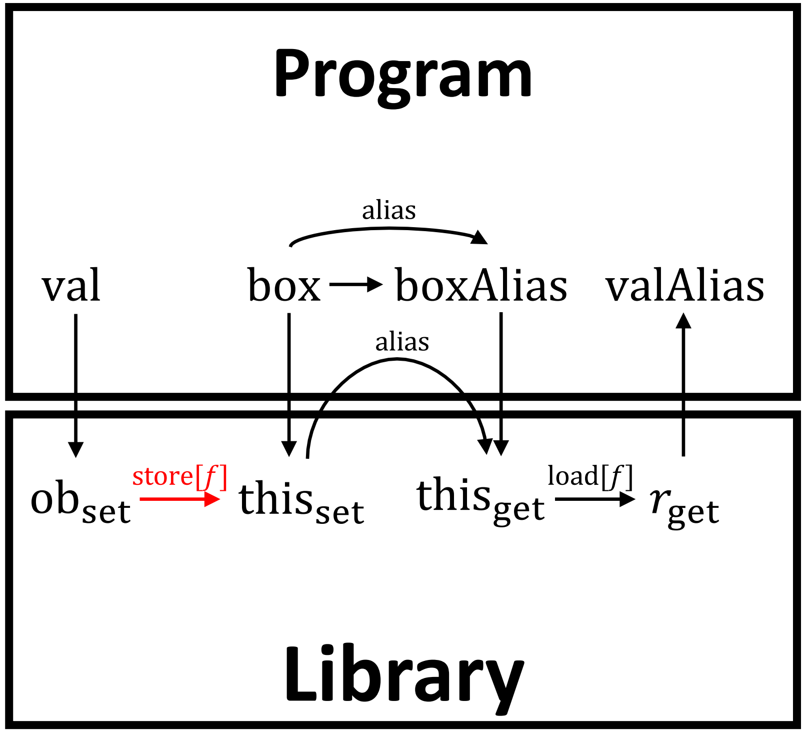
$$u \xrightarrow{\text{alias}} v \rightarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \leftarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{store[f]}} v \xrightarrow{\text{alias}} w \xrightarrow{\text{load[f]}} x \quad \Rightarrow \quad u \xrightarrow{\text{alias}} x$$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```



$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

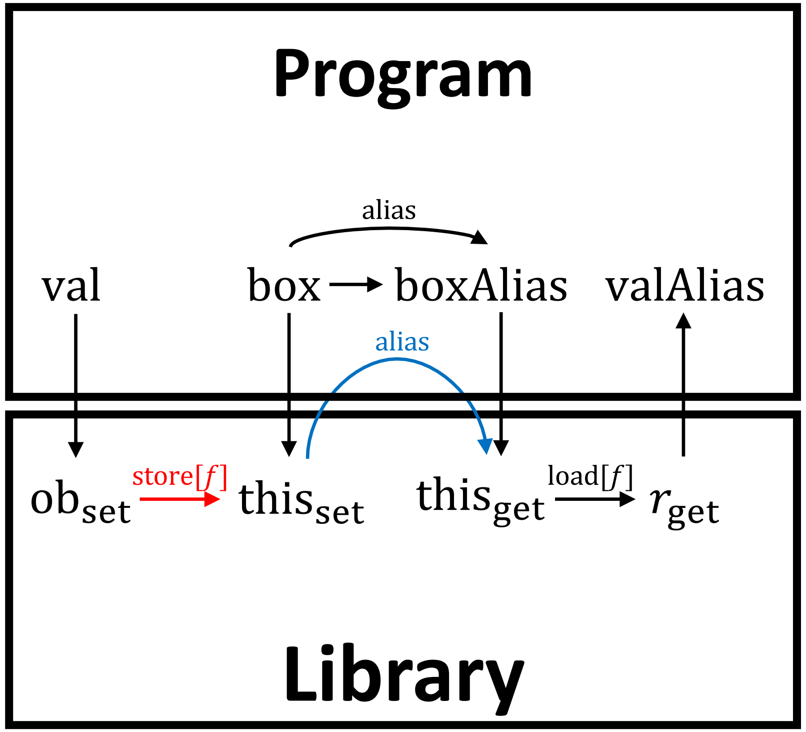
$$u \xrightarrow{\text{alias}} v \rightarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \leftarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{store[f]}} v \xrightarrow{\text{alias}} w \xrightarrow{\text{load[f]}} x \quad \Rightarrow \quad u \xrightarrow{\text{alias}} x$$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```



$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

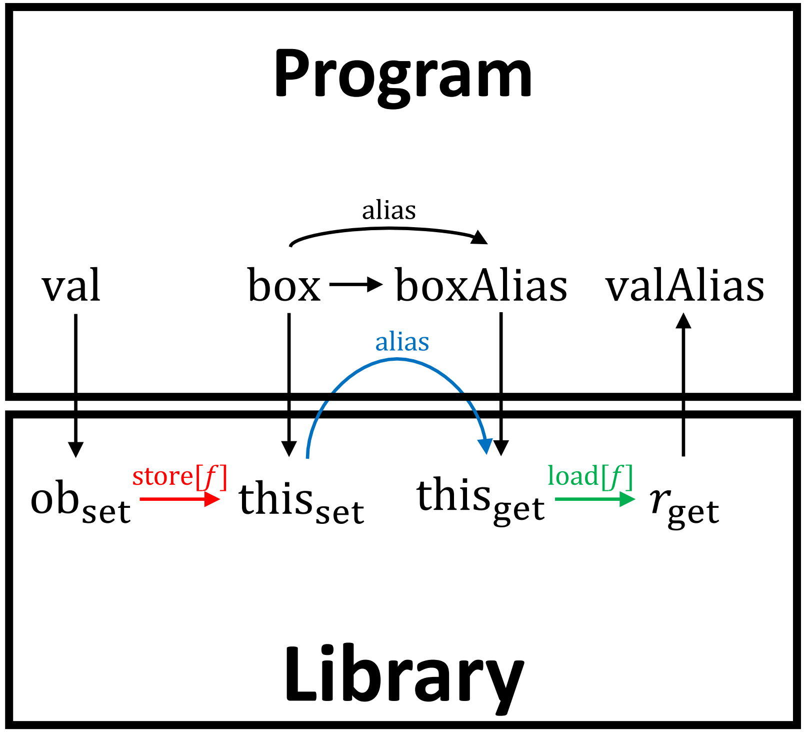
$$u \xrightarrow{\text{alias}} v \rightarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \leftarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{store[f]}} v \xrightarrow{\text{alias}} w \xrightarrow{\text{load[f]}} x \quad \Rightarrow \quad u \xrightarrow{\text{alias}} x$$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```



$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

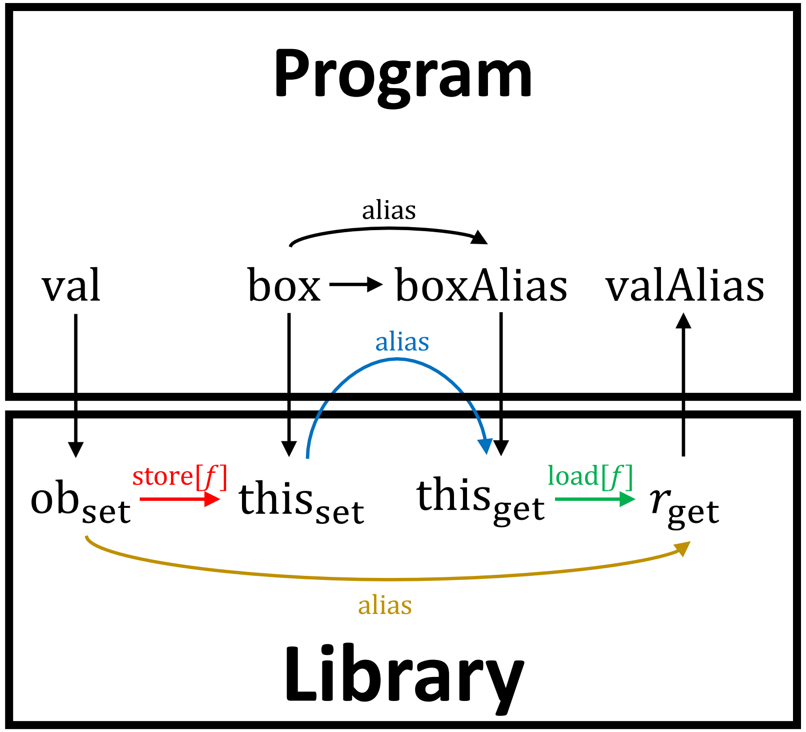
$$u \xrightarrow{\text{alias}} v \rightarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \leftarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{store[f]}} v \xrightarrow{\text{alias}} w \xrightarrow{\text{load[f]}} x \quad \Rightarrow \quad u \xrightarrow{\text{alias}} x$$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();


class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```



$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

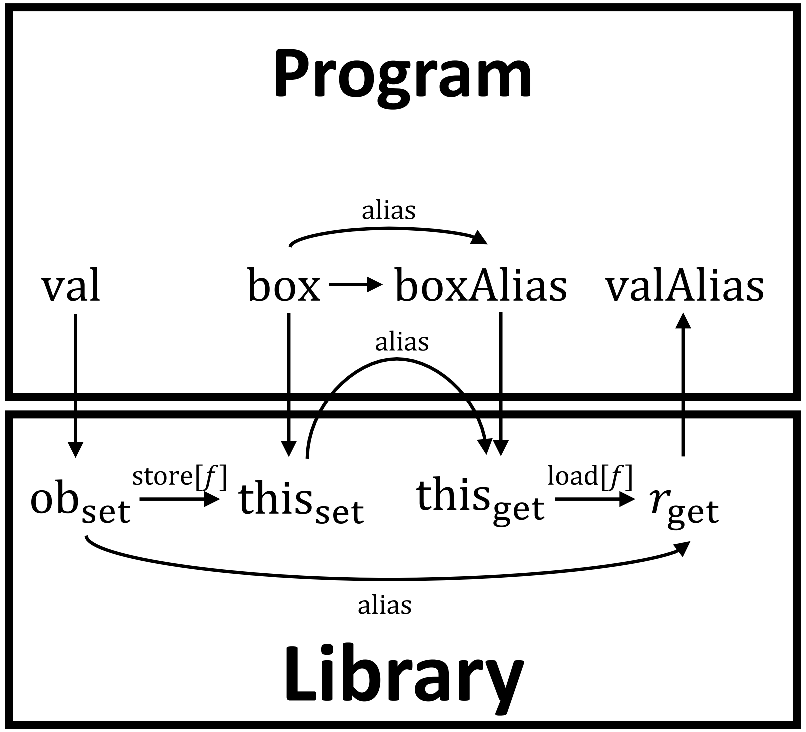
$$u \xrightarrow{\text{alias}} v \rightarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \leftarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{store[f]}} v \xrightarrow{\text{alias}} w \xrightarrow{\text{load[f]}} x \quad \Rightarrow \quad u \xrightarrow{\text{alias}} x$$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```



$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

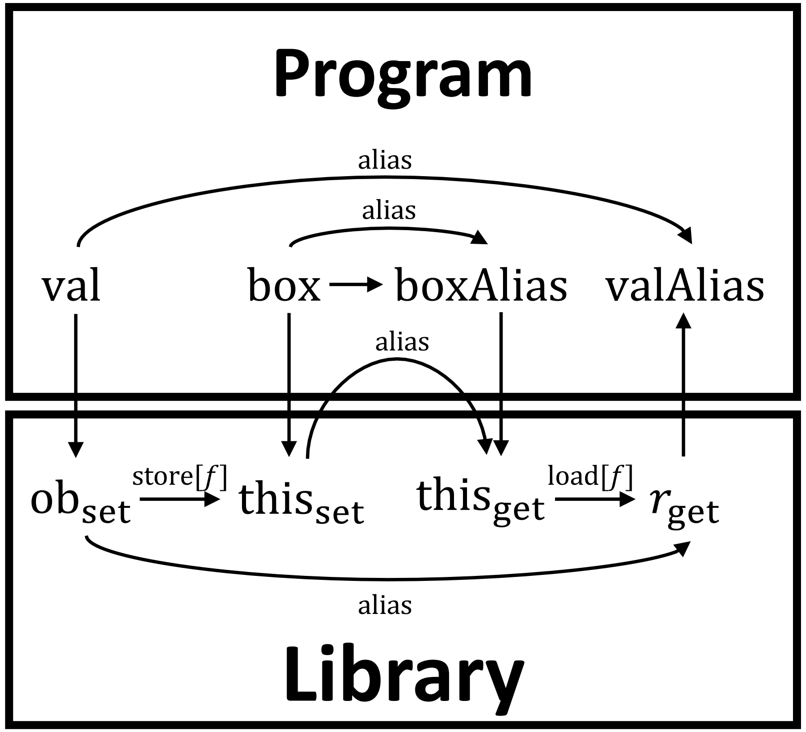
$$u \xrightarrow{\text{alias}} v \rightarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \leftarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{store[f]}} v \xrightarrow{\text{alias}} w \xrightarrow{\text{load[f]}} x \quad \Rightarrow \quad u \xrightarrow{\text{alias}} x$$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();

class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```



$$\Rightarrow \qquad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \qquad \Rightarrow \qquad u \xrightarrow{\text{alias}} w$$

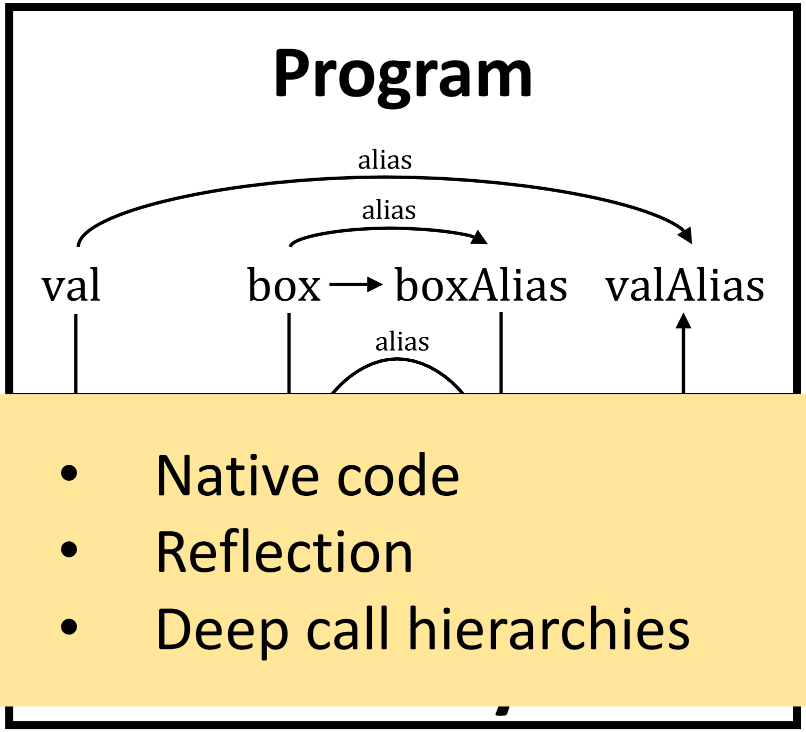
$$u \xrightarrow{\text{alias}} v \rightarrow w \qquad \Rightarrow \qquad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \leftarrow w \qquad \Rightarrow \qquad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{store[f]}} v \xrightarrow{\text{alias}} w \xrightarrow{\text{load[f]}} x \qquad \Rightarrow \qquad u \xrightarrow{\text{alias}} x$$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();


class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
```



- Native code
- Reflection
- Deep call hierarchies

$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

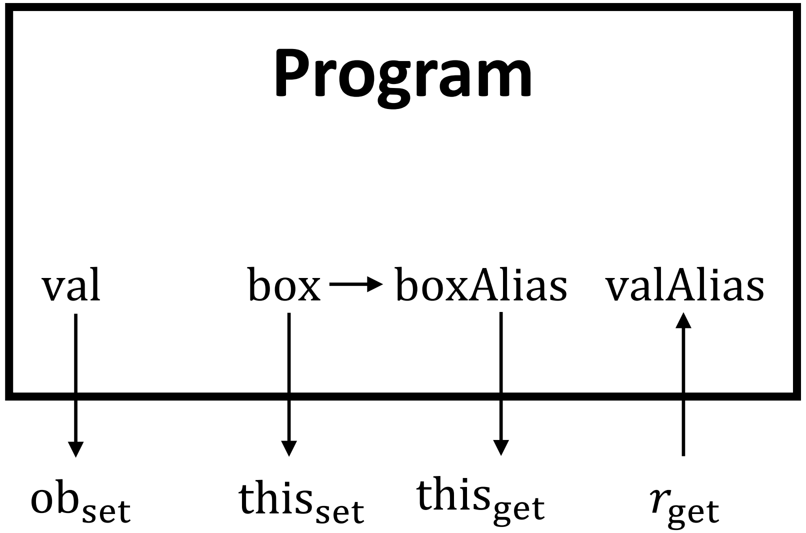
$$u \xrightarrow{\text{alias}} v \rightarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \leftarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{store[f]}} v \xrightarrow{\text{alias}} w \xrightarrow{\text{load[f]}} x \quad \Rightarrow \quad u \xrightarrow{\text{alias}} x$$

Points-To Analysis

```
1.  Double val = new Double(0.0);
2.  Box box = new Box();
3.  box.set(val);
4.  Box boxAlias = box;
5.  Double valAlias = boxAlias.get();
```


Program
val
box
boxAlias
valAlias
$\mathrm{ob}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
r_{get}

$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

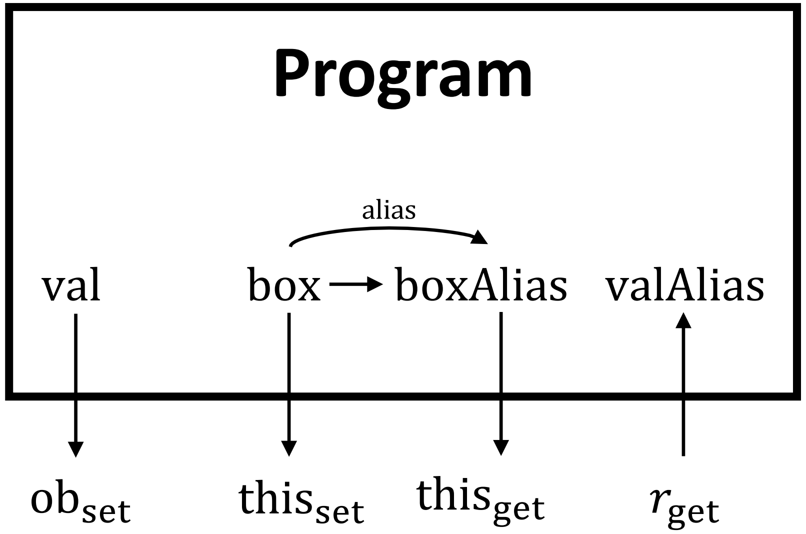
$$u \xrightarrow{\text{alias}} v \rightarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \leftarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{store[f]}} v \xrightarrow{\text{alias}} w \xrightarrow{\text{load[f]}} x \quad \Rightarrow \quad u \xrightarrow{\text{alias}} x$$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();
```


Program
alias
val
box
boxAlias
valAlias
$\mathrm{ob}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
r_{get}

$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

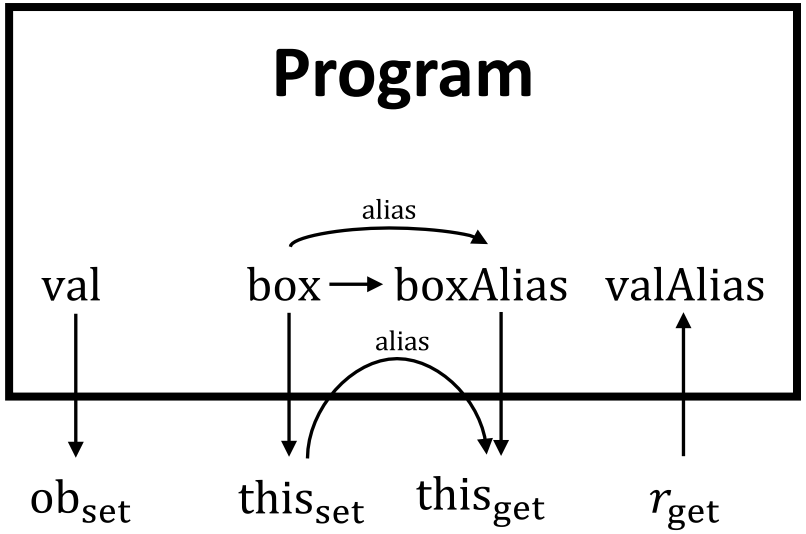
$$u \xrightarrow{\text{alias}} v \rightarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \leftarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{store[f]}} v \xrightarrow{\text{alias}} w \xrightarrow{\text{load[f]}} x \quad \Rightarrow \quad u \xrightarrow{\text{alias}} x$$

Points-To Analysis

```
Double val = new Double(0.0);
Box box = new Box();
box.set(val);
Box boxAlias = box;
Double valAlias = boxAlias.get();
```


Program
alias
val
box
boxAlias
valAlias
alias
$\mathrm{ob}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
r_{get}

$$\Rightarrow \quad v \xrightarrow{\text{alias}} v$$

$$u \xrightarrow{\text{alias}} v \xrightarrow{\text{alias}} w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{alias}} v \rightarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

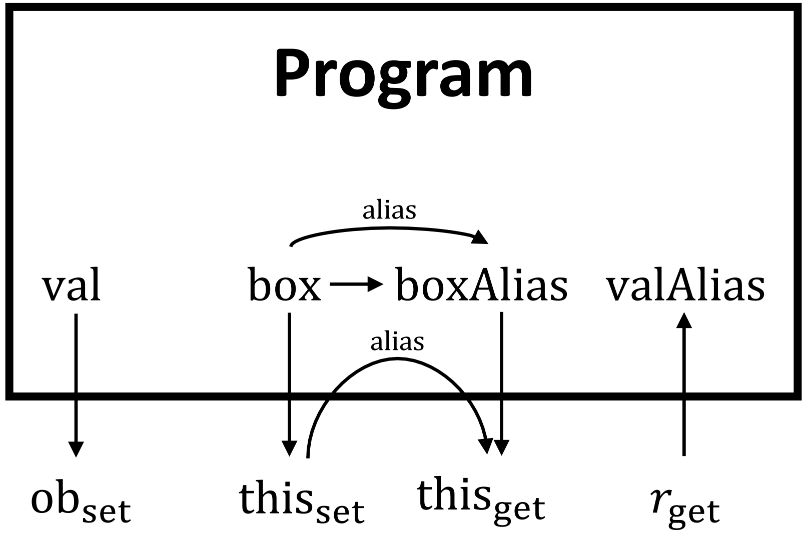
$$u \xrightarrow{\text{alias}} v \leftarrow w \quad \Rightarrow \quad u \xrightarrow{\text{alias}} w$$

$$u \xrightarrow{\text{store[f]}} v \xrightarrow{\text{alias}} w \xrightarrow{\text{load[f]}} x \quad \Rightarrow \quad u \xrightarrow{\text{alias}} x$$

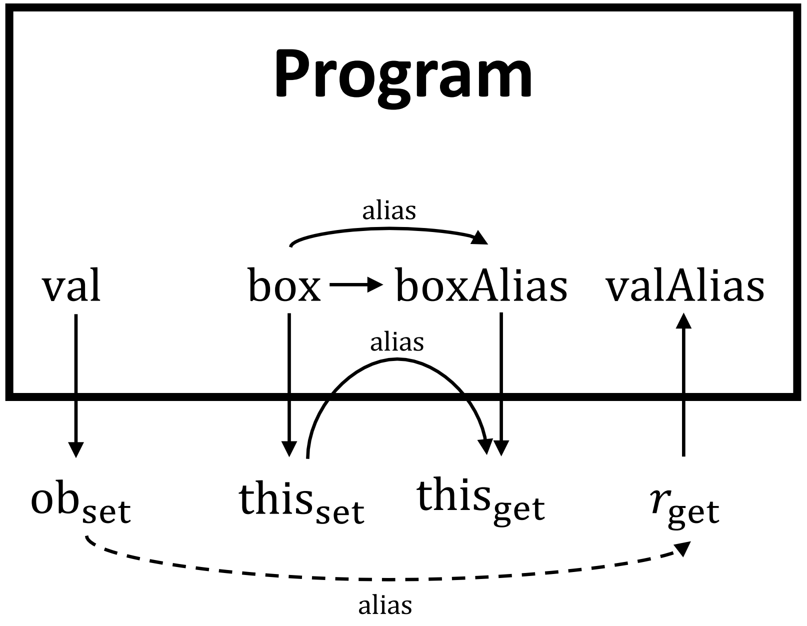
Roadmap

- Points-to analysis
- **Path specifications**
- Inference algorithm
- Evaluation

Path Specifications: Intuition


Program
alias
val
box
boxAlias
valAlias
alias
$\mathrm{ob}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
r_{get}

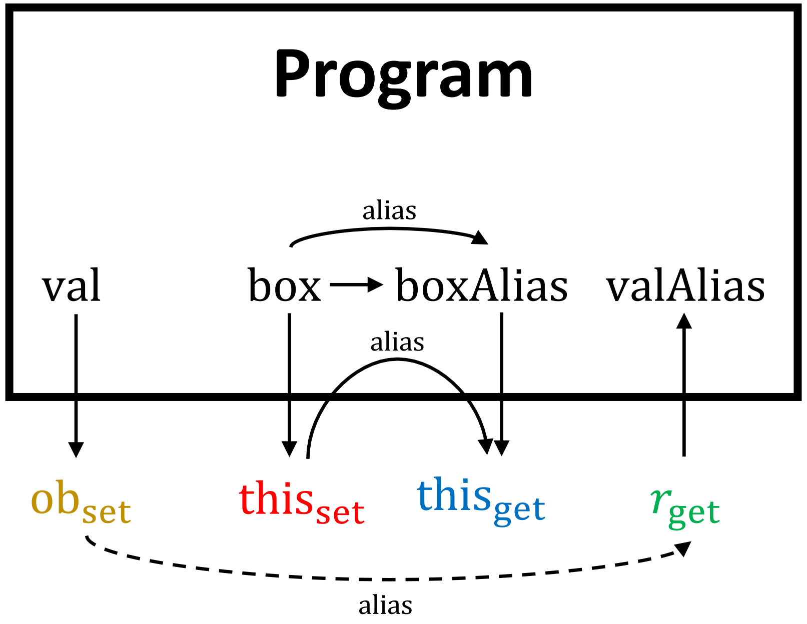
Path Specifications: Intuition


Program
alias
val
box
boxAlias
valAlias
alias
$\mathrm{ob}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
r_{get}
alias

Path Specifications: Intuition

- When the library code is available, the edge $\mathrm{ob}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} r_{\mathrm{get}}$ is produced by the path

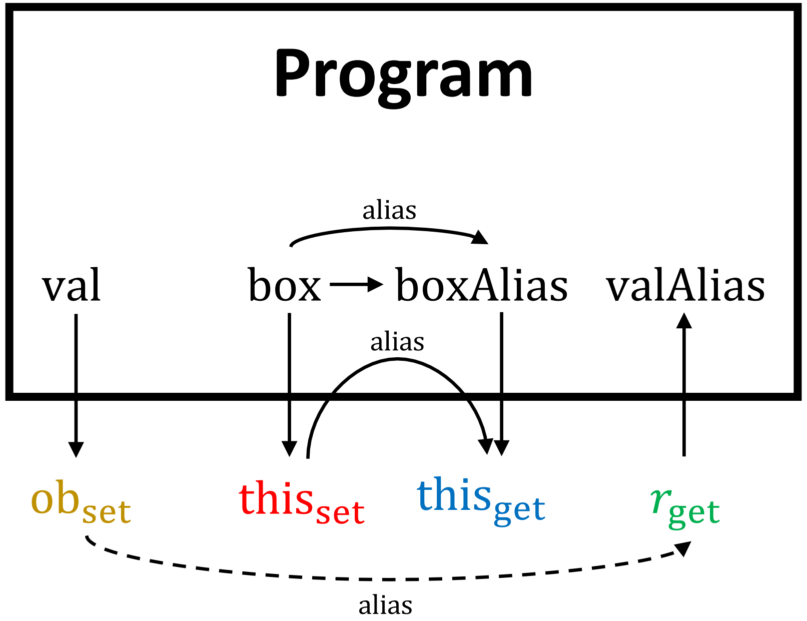
$$\begin{tikzcd} \mathrm{ob}_{\mathrm{set}} \arrow[r, dashed, "\mathrm{store}[f]"] & \mathrm{this}_{\mathrm{set}} \arrow[r, "\mathrm{alias}"] & \mathrm{this}_{\mathrm{get}} \arrow[r, dashed, "\mathrm{load}[f]"] & r_{\mathrm{get}} \end{tikzcd}$$



Path Specifications: Intuition

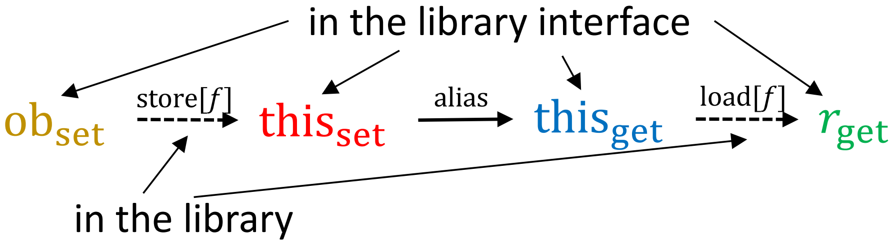
- When the library code is available, the edge $\mathrm{ob}_{\mathrm{set}} \xrightarrow{\text{alias}} r_{\mathrm{get}}$ is produced by the path in the library interface

$$\begin{tikzcd}
\mathrm{ob}_{\mathrm{set}} \arrow[r, dashed, "\mathrm{store}[f]"] & \mathrm{this}_{\mathrm{set}} \arrow[r, "\mathrm{alias}"] & \mathrm{this}_{\mathrm{get}} \arrow[r, dashed, "\mathrm{load}[f]"] & r_{\mathrm{get}}
\end{tikzcd}$$

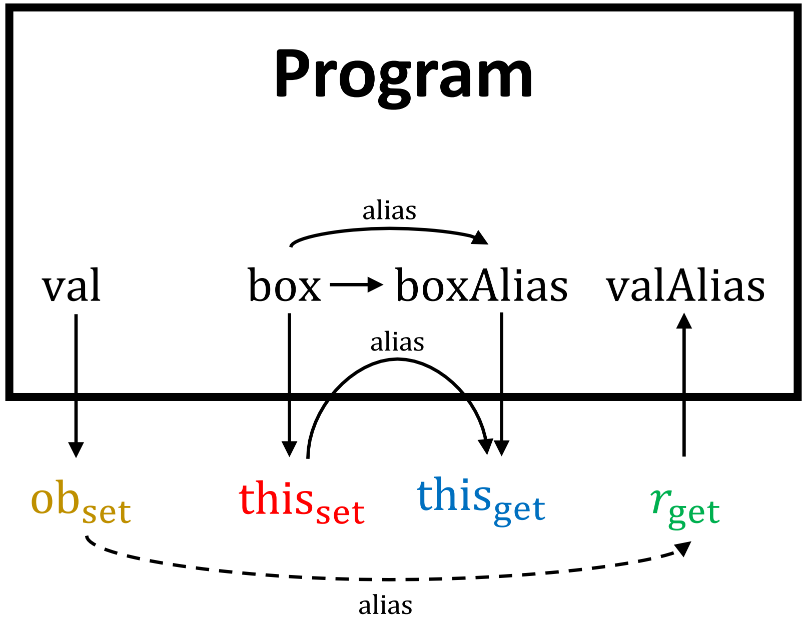


Path Specifications: Intuition

- When the library code is available, the edge $\mathrm{ob}_{\mathrm{set}} \xrightarrow{\text{alias}} r_{\mathrm{get}}$ is produced by the path



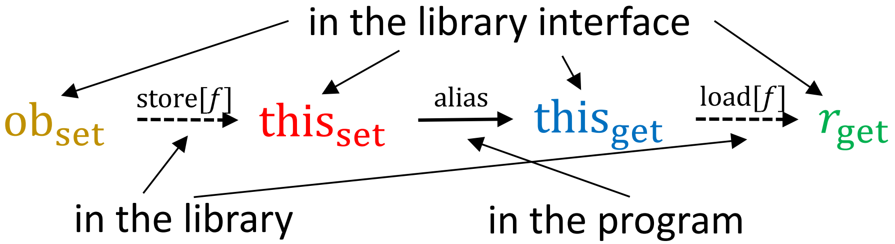
in the library interface
$\mathrm{ob}_{\mathrm{set}}$
store[f]
$\mathrm{this}_{\mathrm{set}}$
alias
$\mathrm{this}_{\mathrm{get}}$
load[f]
r_{get}
in the library

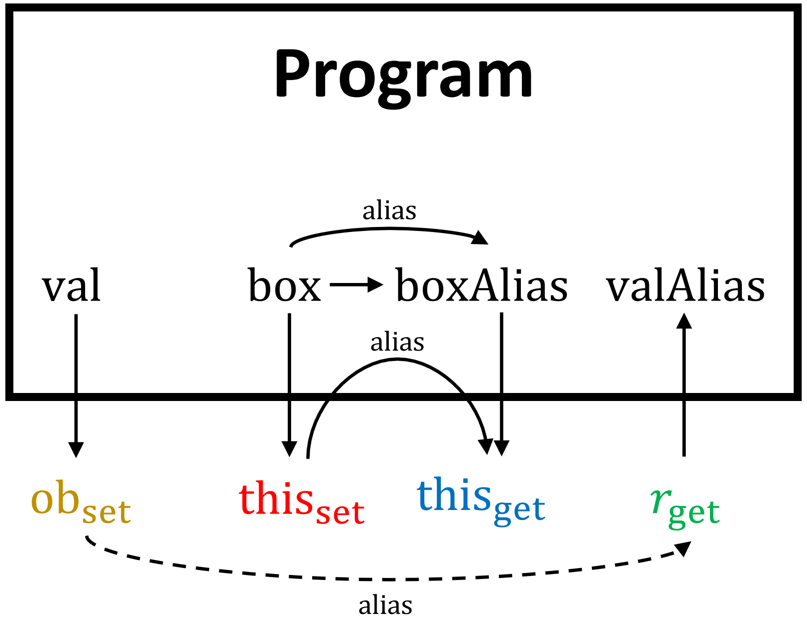


Program
alias
val
box
boxAlias
valAlias
alias
$\mathrm{ob}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
r_{get}
alias

Path Specifications: Intuition

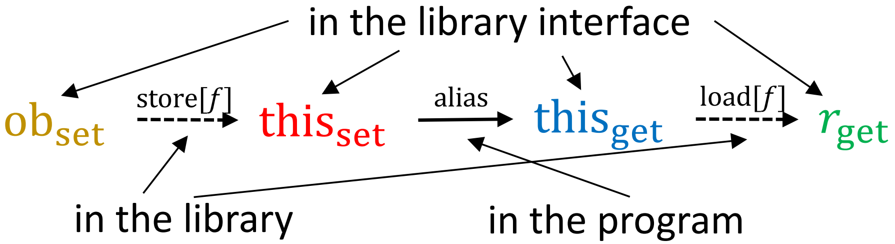
- When the library code is available, the edge $\mathrm{ob}_{\mathrm{set}} \xrightarrow{\text{alias}} r_{\mathrm{get}}$ is produced by the path




Program
alias
val
box
boxAlias
valAlias
alias
$\mathrm{ob}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
r_{get}
alias

Path Specifications: Intuition

- When the library code is available, the edge $\mathrm{ob}_{\mathrm{set}} \xrightarrow{\text{alias}} r_{\mathrm{get}}$ is produced by the path



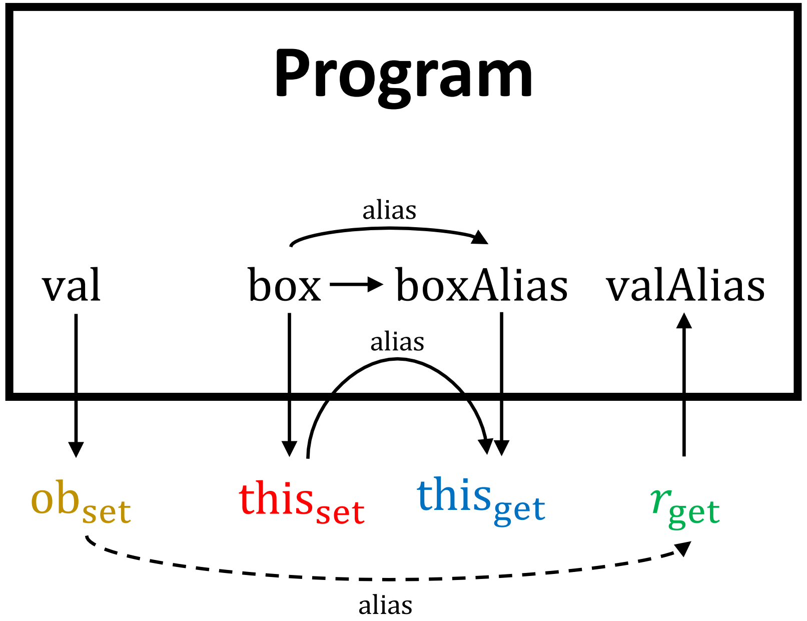
in the library interface

$\mathrm{ob}_{\mathrm{set}}$ store$[f]$ $\mathrm{this}_{\mathrm{set}}$ alias $\mathrm{this}_{\mathrm{get}}$ load$[f]$ r_{get}

in the library

in the program

- A path specification says “if the solid edges are in the program, then the edges in the library complete them into a path that produces an alias edge”



Program

alias

val box boxAlias valAlias

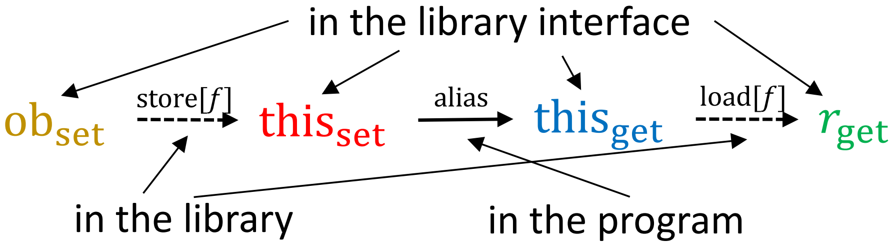
alias

$\mathrm{ob}_{\mathrm{set}}$ $\mathrm{this}_{\mathrm{set}}$ $\mathrm{this}_{\mathrm{get}}$ r_{get}

alias

Path Specifications: Intuition

- When the library code is available, the edge $ob_{set} \xrightarrow{\text{alias}} r_{get}$ is produced by the path



in the library interface

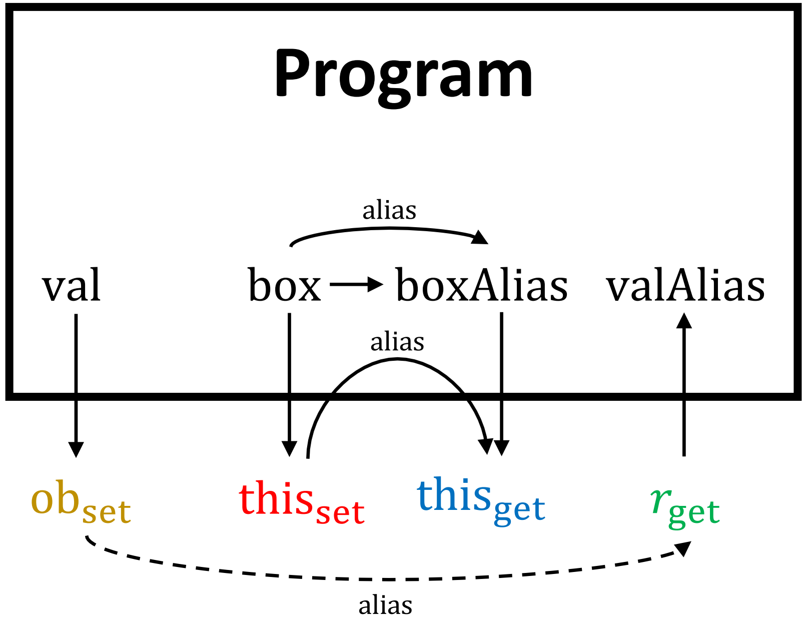
$ob_{set} \overset{store[f]}{\dashrightarrow} this_{set} \xrightarrow{\text{alias}} this_{get} \overset{load[f]}{\dashrightarrow} r_{get}$

in the library

in the program

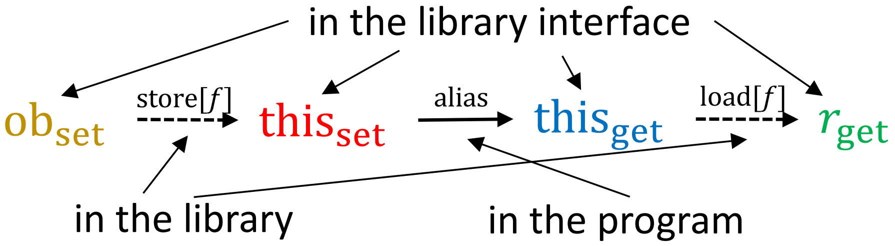
- A path specification says "if the solid edges are in the program, then the edges in the library complete them into a path that produces an alias edge"

Syntax: $ob_{set} \dashrightarrow this_{set} \xrightarrow{\text{alias}} this_{get} \dashrightarrow r_{get}$



Path Specifications: Intuition

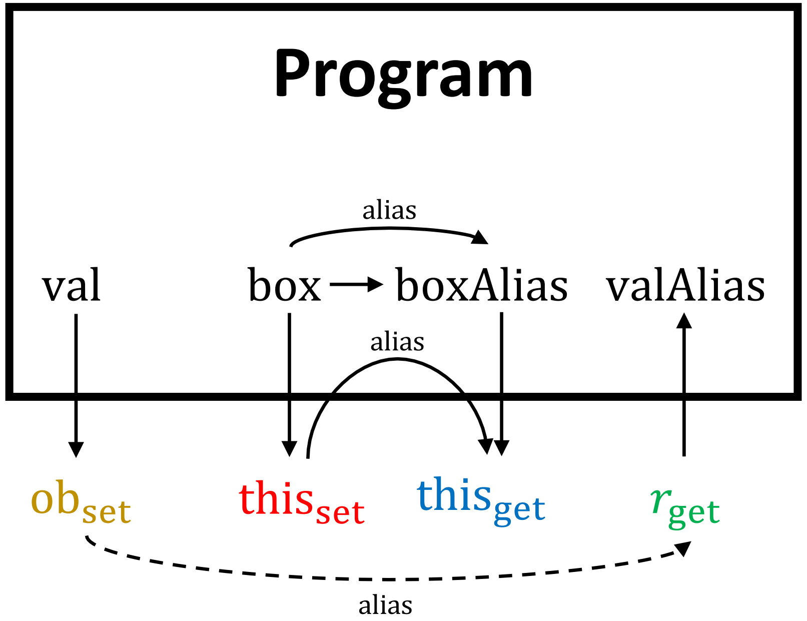
- When the library code is available, the edge $\mathrm{ob}_{\mathrm{set}} \xrightarrow{\text{alias}} r_{\mathrm{get}}$ is produced by the path



- A path specification says “if the solid edges are in the program, then the edges in the library complete them into a path that produces an alias edge”

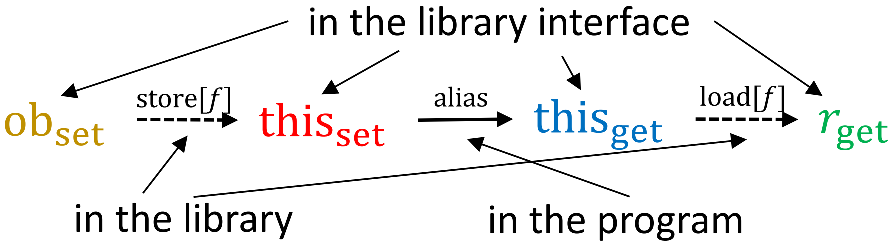
Syntax: $\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}} \xrightarrow{\text{alias}} \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

Semantics: $\mathrm{this}_{\mathrm{set}} \xrightarrow{\text{alias}} \mathrm{this}_{\mathrm{get}} \Rightarrow \mathrm{ob}_{\mathrm{set}} \xrightarrow{\text{alias}} r_{\mathrm{get}}$


Program
alias
val
box
boxAlias
valAlias
alias
$\mathrm{ob}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
r_{get}
alias

Path Specifications: Intuition

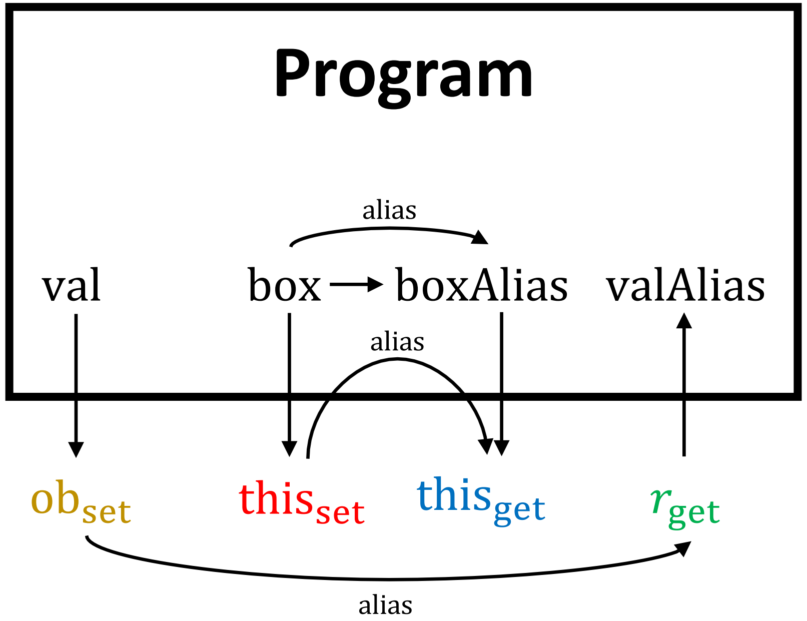
- When the library code is available, the edge $\mathrm{ob}_{\mathrm{set}} \xrightarrow{\text{alias}} r_{\mathrm{get}}$ is produced by the path



- A path specification says “if the solid edges are in the program, then the edges in the library complete them into a path that produces an alias edge”

Syntax: $\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}} \xrightarrow{\text{alias}} \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

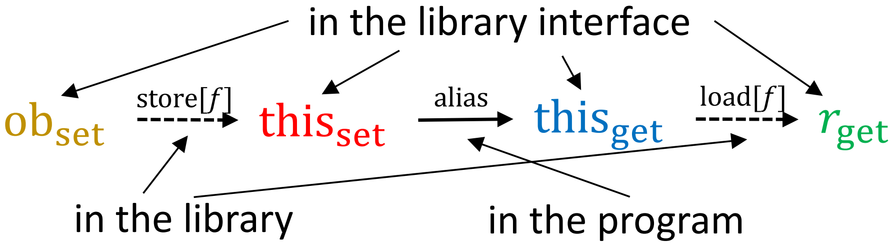
Semantics: $\mathrm{this}_{\mathrm{set}} \xrightarrow{\text{alias}} \mathrm{this}_{\mathrm{get}} \Rightarrow \mathrm{ob}_{\mathrm{set}} \xrightarrow{\text{alias}} r_{\mathrm{get}}$



Program
alias
val
box
boxAlias
valAlias
alias
$\mathrm{ob}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
r_{get}
alias

Path Specifications: Intuition

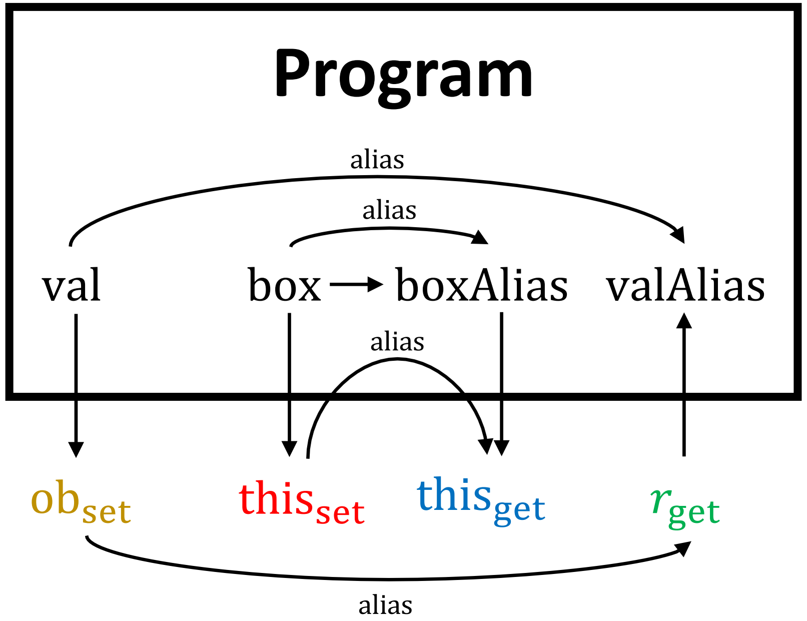
- When the library code is available, the edge $\mathrm{ob}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} r_{\mathrm{get}}$ is produced by the path



- A path specification says "if the solid edges are in the program, then the edges in the library complete them into a path that produces an alias edge"

Syntax: $\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

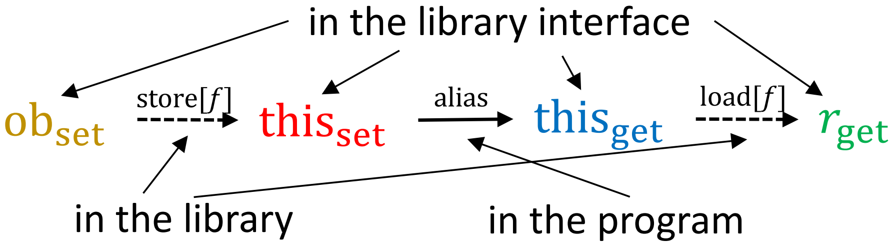
Semantics: $\mathrm{this}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} \mathrm{this}_{\mathrm{get}} \Rightarrow \mathrm{ob}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} r_{\mathrm{get}}$



Program
alias
alias
val
box
boxAlias
valAlias
alias
$\mathrm{ob}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
r_{get}
alias

Path Specifications: Intuition

- When the library code is available, the edge $\mathrm{ob}_{\mathrm{set}} \xrightarrow{\text{alias}} r_{\mathrm{get}}$ is produced by the path

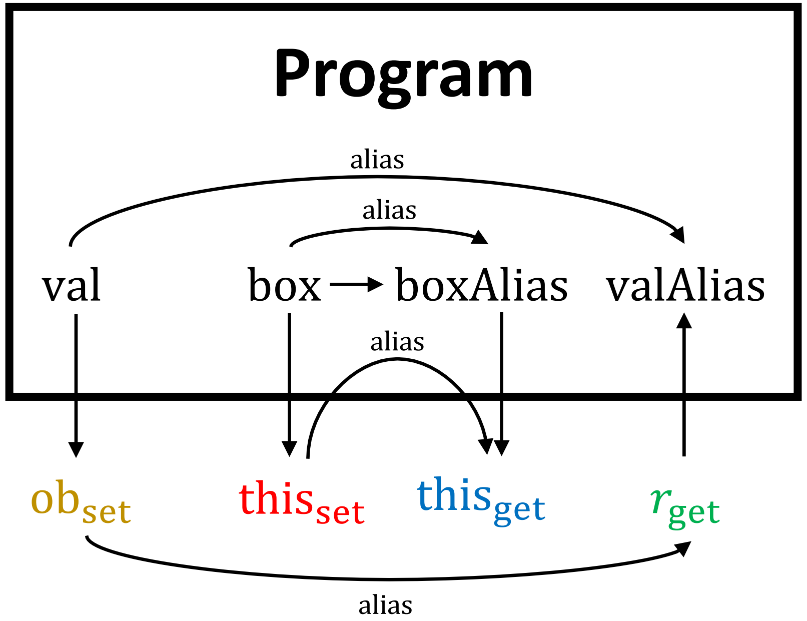


- A path specification says "if the solid edges are in the program, then the edges in the library complete them into a path that produces an alias edge"

Syntax: $\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}} \xrightarrow{\text{alias}} \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

Semantics: $\mathrm{this}_{\mathrm{set}} \xrightarrow{\text{alias}} \mathrm{this}_{\mathrm{get}} \Rightarrow \mathrm{ob}_{\mathrm{set}} \xrightarrow{\text{alias}} r_{\mathrm{get}}$

- **Theorem:** It "suffices" to use path specifications where the solid edges are alias edges



Program
alias
alias
val
box
boxAlias
valAlias
alias
$\mathrm{ob}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{set}}$
$\mathrm{this}_{\mathrm{get}}$
r_{get}
alias

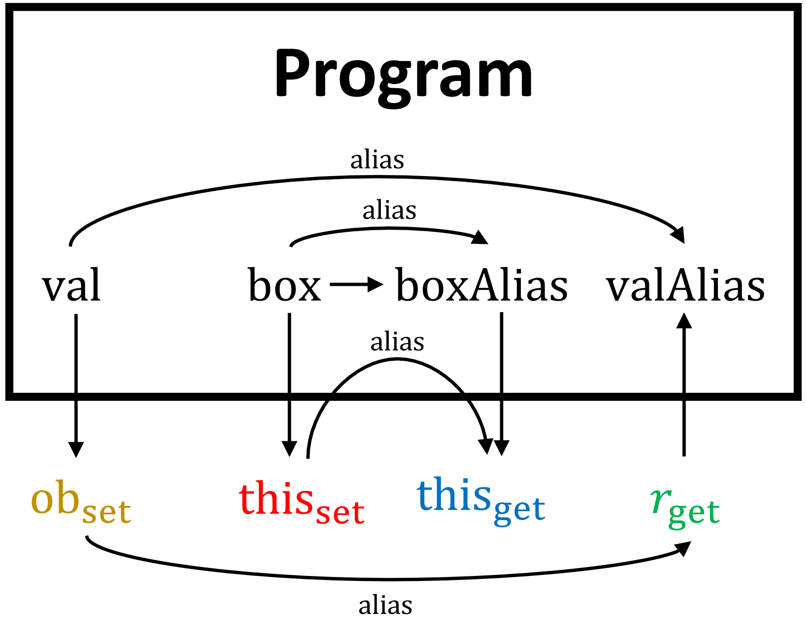
Path Specifications: Intuition

Syntax: A path specification is a sequence of library interface variables

$\text{ob}_{\text{set}} \dashrightarrow \text{this}_{\text{set}} \rightarrow \text{this}_{\text{get}} \dashrightarrow r_{\text{get}} \in V_{\text{lib}}^{*}$

Semantics: If the program edges in the path occur, then the path produces an alias edge

$\text{this}_{\text{set}} \xrightarrow{\text{alias}} \text{this}_{\text{get}} \quad \Rightarrow \quad \text{ob}_{\text{set}} \xrightarrow{\text{alias}} r_{\text{get}}$



Path Specifications: Definition

- **Syntax**

$$z_1 \dashrightarrow w_1 \to z_2 \dashrightarrow w_2 \to z_3 \dashrightarrow \cdots \dashrightarrow w_{k-1} \to z_k \dashrightarrow w_k \in V_{\text{lib}}^*$$

where z_i, w_i are variables in the interface of library function f_i

- **Semantics**

$$w_1 \xrightarrow{\text{alias}} z_2 \wedge w_2 \xrightarrow{\text{alias}} z_3 \wedge \cdots \wedge w_{k-1} \xrightarrow{\text{alias}} z_k \Rightarrow z_1 \xrightarrow{\text{alias}} w_k$$

Issue with Path Specifications

- **Intuition:**

$$\begin{array}{c}\text{sequence of library function calls}\\ \text{in the program}\end{array} \Rightarrow \begin{array}{c}\text{alias relationship}\\ \text{in the program}\end{array}$$

- **Issue:** A different path specification is needed for every possible sequence of library function calls that may produce an alias edge

Infinite Sets of Path Specifications

Infinite Sets of Path Specifications

```
class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
    Object clone():
        Box b = new Box();
        b.f = f;
        return b;
```

Infinite Sets of Path Specifications

```
class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
    Object clone():
        Box b = new Box();
        b.f = f;
        return b;
```

```
Object in = new Object();
Box box0 = new Box();
box0.add(in);
Object out = box0.get();
return in == out;
```

$$\text{ob}_{\text{set}} \dashrightarrow \text{this}_{\text{set}} \rightarrow \text{this}_{\text{get}} \dashrightarrow r_{\text{get}}$$

Infinite Sets of Path Specifications

```
class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
    Object clone():
        Box b = new Box();
        b.f = f;
        return b;
```

```
Object in = new Object();
Box box0 = new Box();
box0.set(in);
Box box1 = box0.clone();
Object out = box1.get();
return in == out;
```

$$\text{ob}_{\text{set}} \dashrightarrow \text{this}_{\text{set}} \rightarrow \text{this}_{\text{clone}} \dashrightarrow r_{\text{clone}} \rightarrow \text{this}_{\text{get}} \dashrightarrow r_{\text{get}}$$

Infinite Sets of Path Specifications

```
class Box:  // library
    Object f;
    void set(Object ob): f = ob;
    Object get(): return f;
    Object clone():
        Box b = new Box();
        b.f = f;
        return b;
```

```
Object in = new Object();
Box box0 = new Box();
box0.set(in);
Box box1 = box0.clone();
Box box2 = box1.clone();
Object out = box2.get();
return in == out;
```

$$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}} \rightarrow \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}} \rightarrow \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}} \rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$$

Infinite Sets of Path Specifications

$$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}(\rightarrow \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}})^{*} \rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}} \subseteq V_{\mathrm{lib}}^{*}$$

Roadmap

- Motivating example
- Path specifications
- **Inference algorithm**
- Evaluation

Inference Algorithm

Inference Algorithm

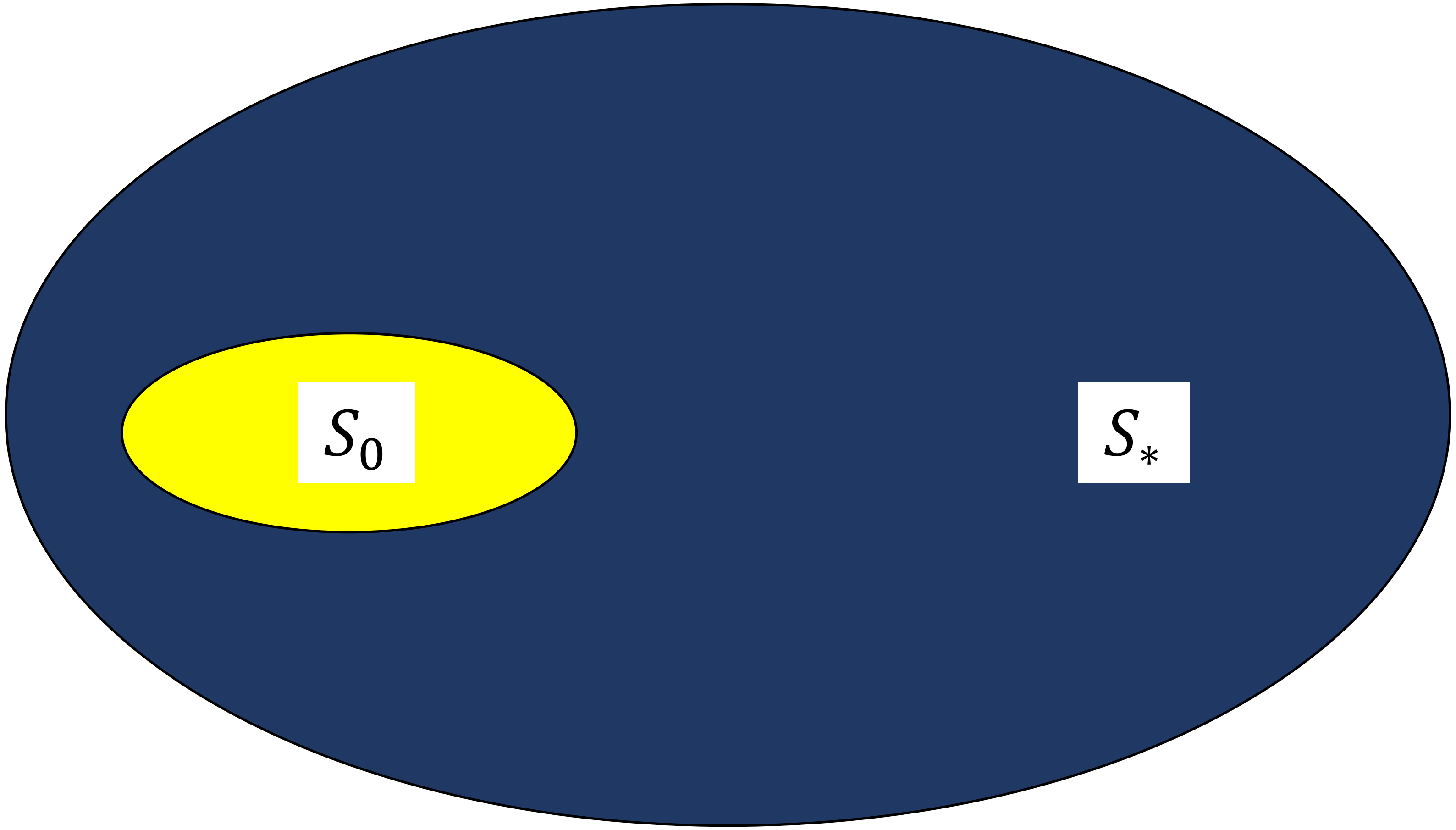
Idea: Construct increasingly general sets of path specifications

Inference Algorithm

Idea: Construct increasingly general sets of path specifications

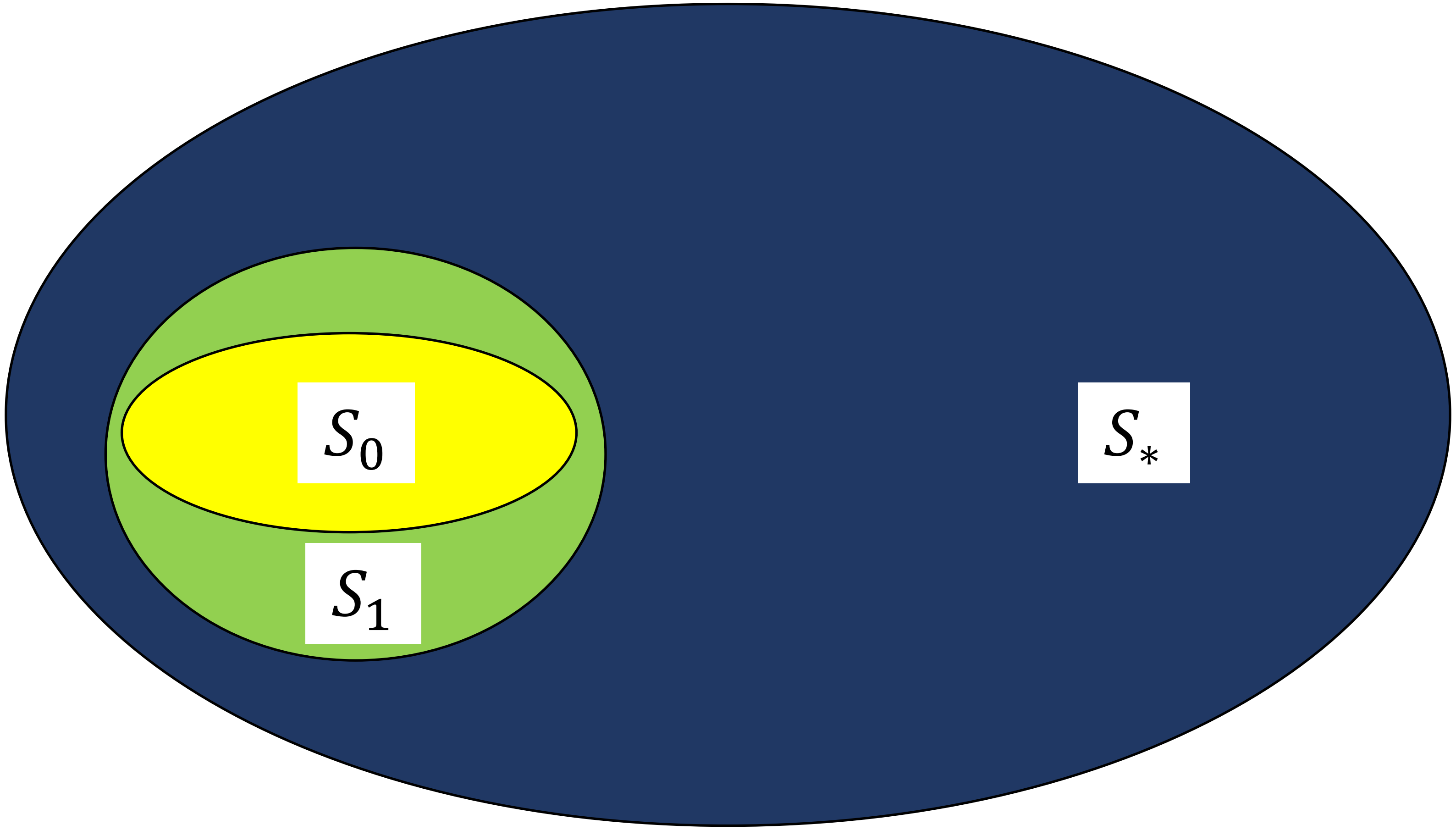
Inference Algorithm

Idea: Construct increasingly general sets of path specifications



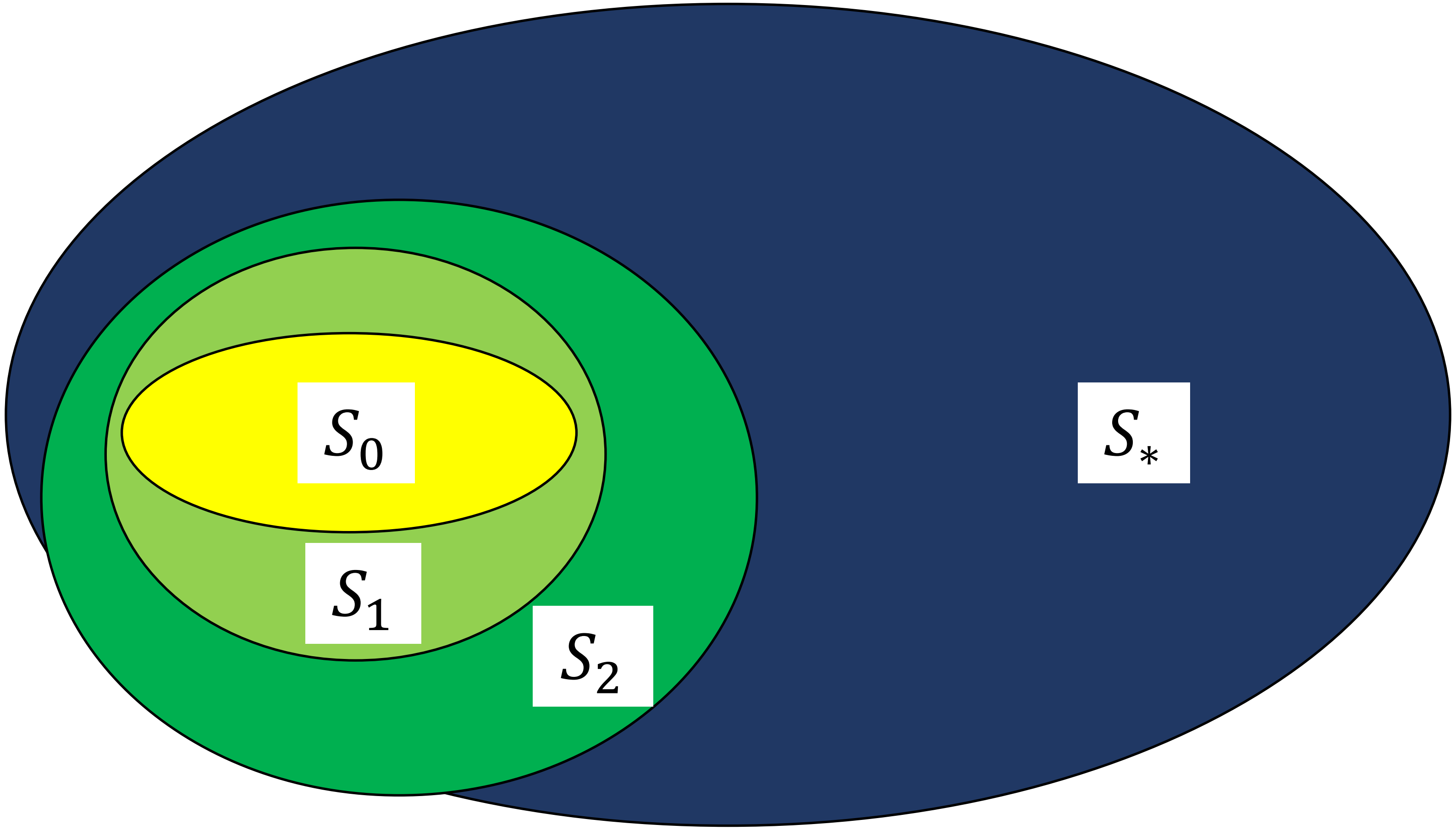
Inference Algorithm

Idea: Construct increasingly general sets of path specifications



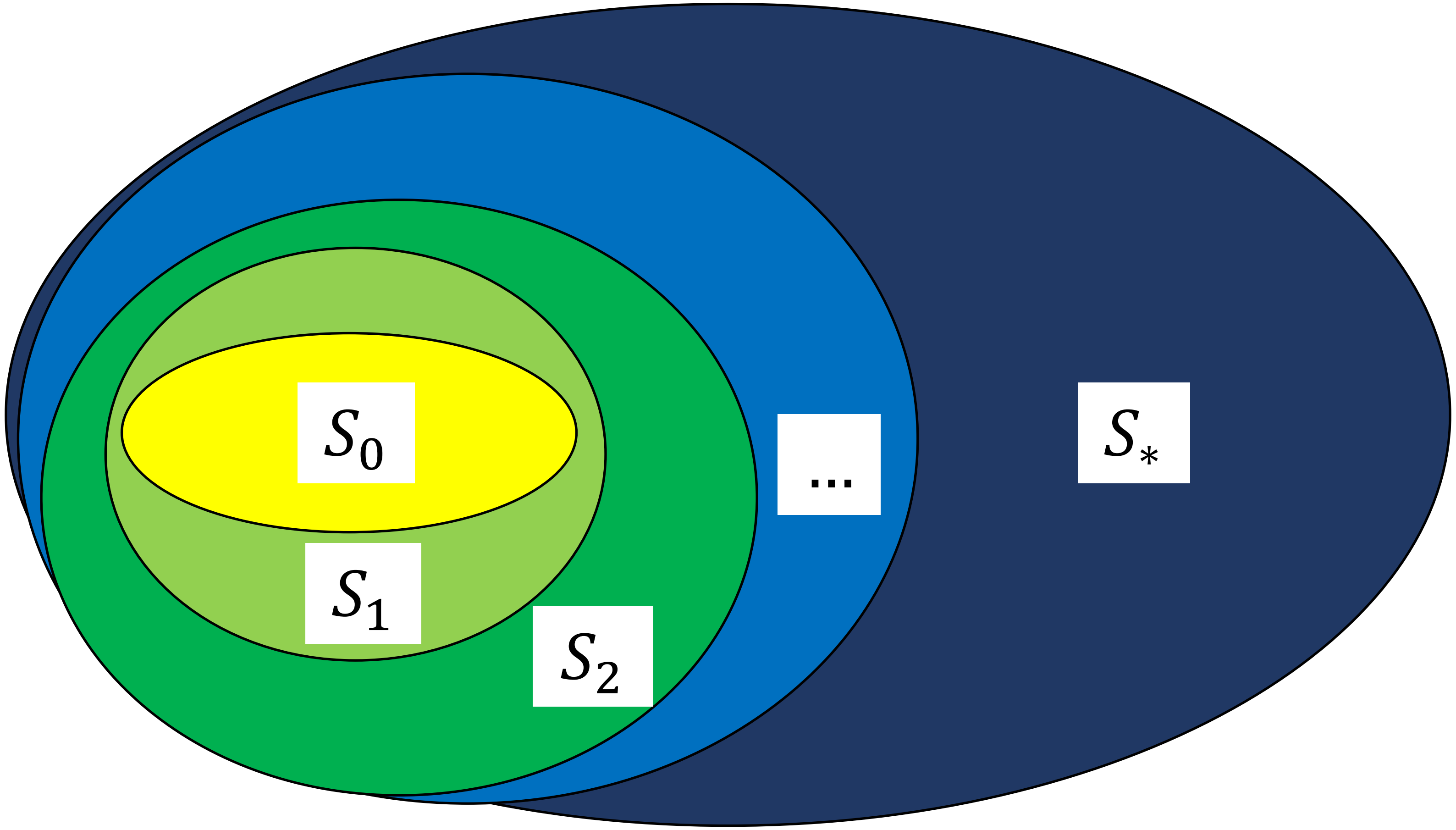
Inference Algorithm

Idea: Construct increasingly general sets of path specifications



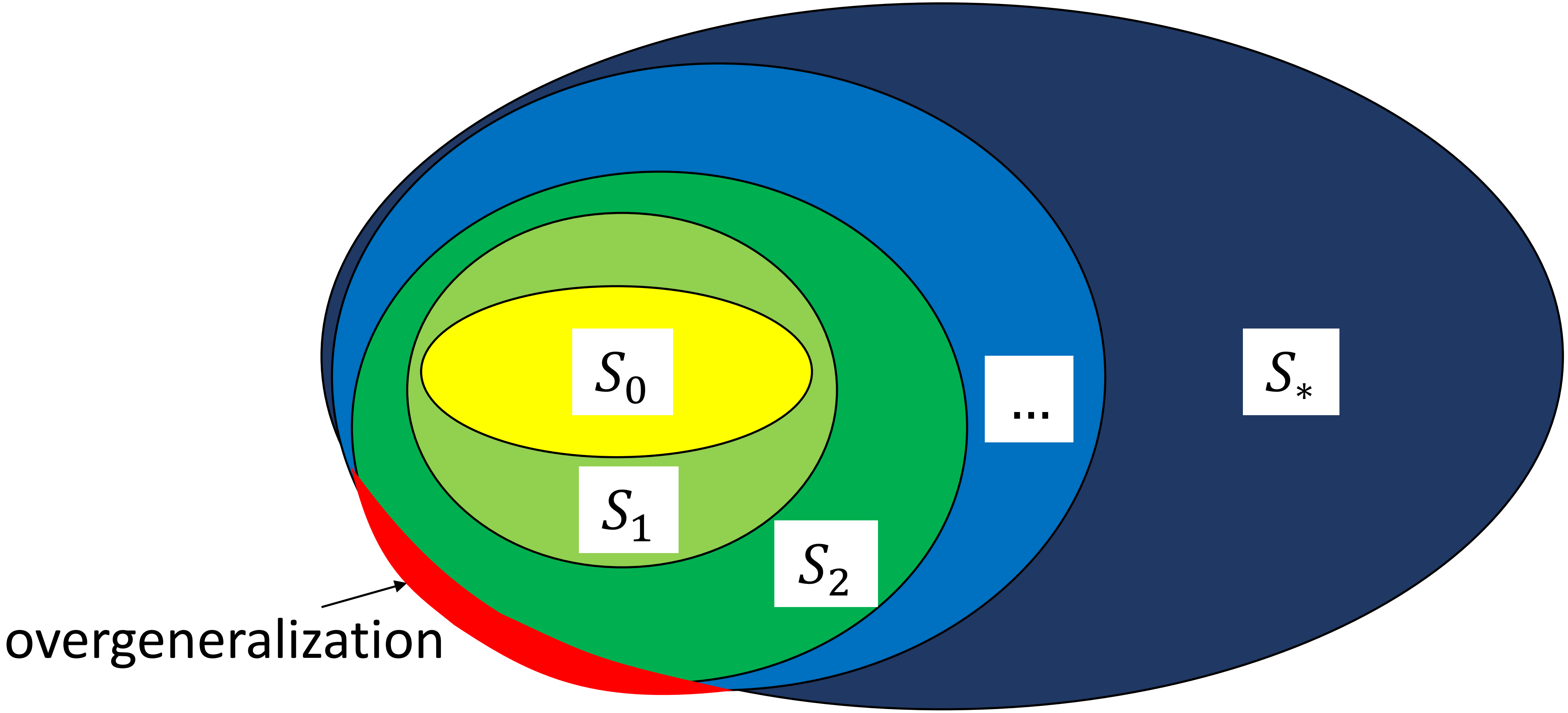
Inference Algorithm

Idea: Construct increasingly general sets of path specifications



Inference Algorithm

Idea: Construct increasingly general sets of path specifications



Inference Algorithm

Inference Algorithm

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

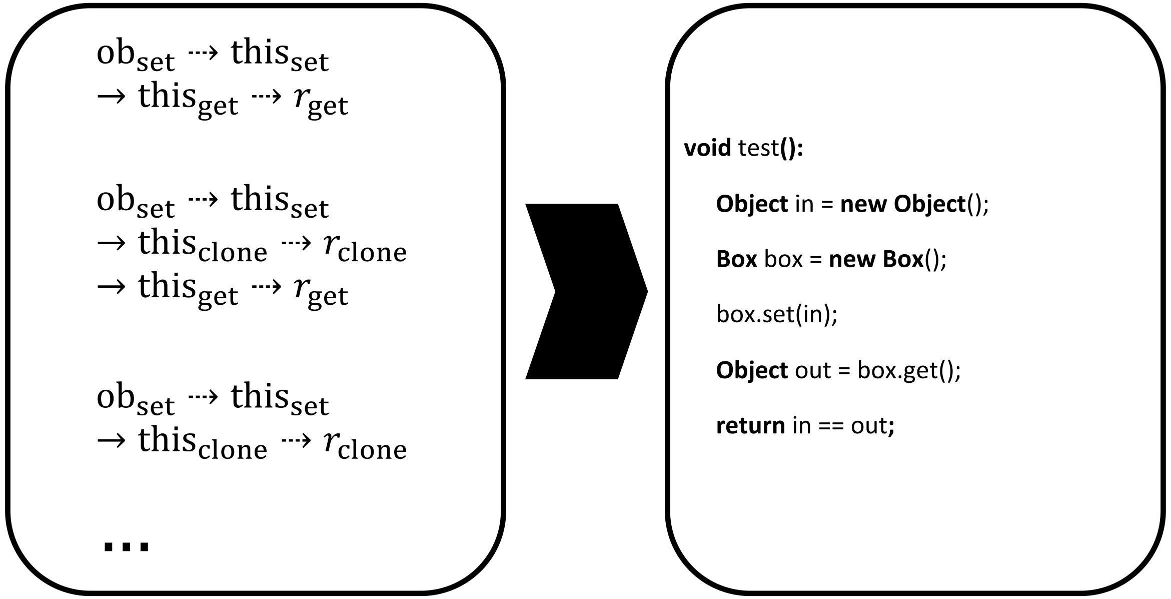
$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\rightarrow \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$
$\rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\rightarrow \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$

…

Step 1: Generate candidates

Inference Algorithm



$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\to \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\to \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$
$\to \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\to \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$

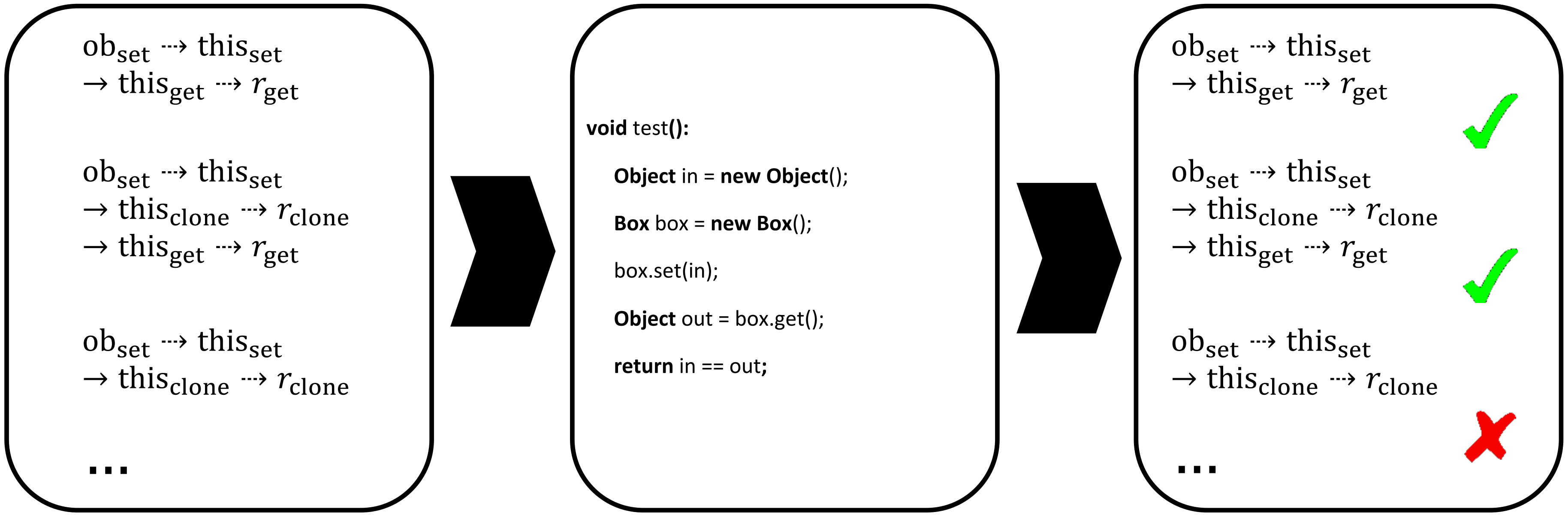
…

```
void test():
    Object in = new Object();
    Box box = new Box();
    box.set(in);
    Object out = box.get();
    return in == out;
```

Step 1: Generate candidates

Step 2a: Synthesize unit tests to check candidates

Inference Algorithm



$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\rightarrow \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$
$\rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\rightarrow \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$

…

```
void test():
    Object in = new Object();
    Box box = new Box();
    box.set(in);
    Object out = box.get();
    return in == out;
```

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$ ✓

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\rightarrow \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$
$\rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$ ✓

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\rightarrow \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$ ✗

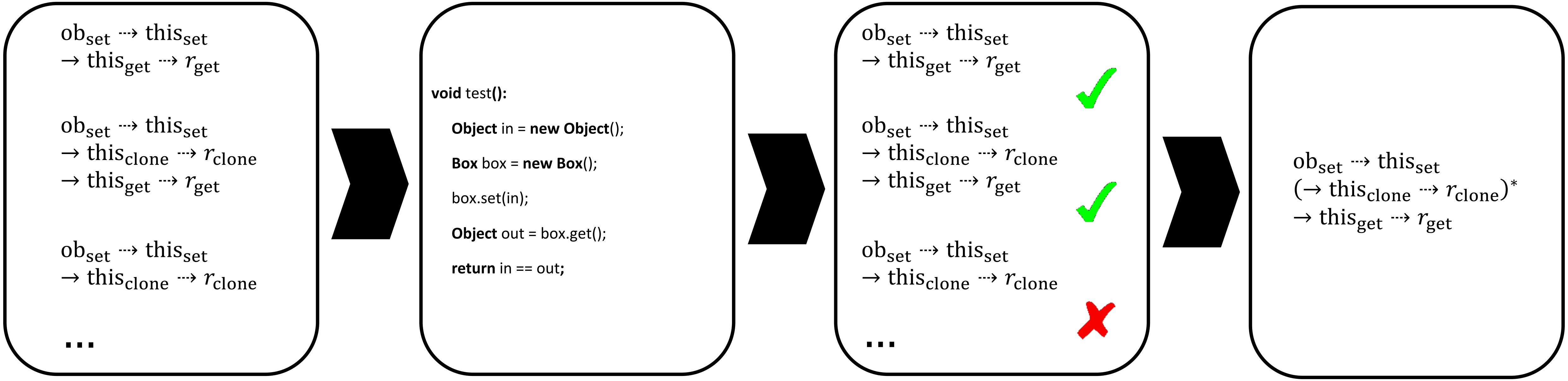
…

Step 1: Generate candidates

Step 2a: Synthesize unit tests to check candidates

Step 2b: Retain precise candidates S_0

Inference Algorithm



$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\to \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\to \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$
$\to \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\to \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$

…

```
void test():
    Object in = new Object();
    Box box = new Box();
    box.set(in);
    Object out = box.get();
    return in == out;
```

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\to \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$ ✓

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\to \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$
$\to \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$ ✓

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\to \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$ ✗

…

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$(\to \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}})^*$
$\to \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

Step 1: Generate candidates

Step 2a: Synthesize unit tests to check candidates

Step 2b: Retain precise candidates S_0

Step 3: Generalize S_0 to a regular set

Generating Candidate Specifications

Generating Candidate Specifications

- A path specification is a sequence

$$z_1 w_1 z_2 w_2 \ldots z_k w_k \in V_{\text{lib}}^*$$

Generating Candidate Specifications

- A path specification is a sequence

$$z_1 w_1 z_2 w_2 \dots z_k w_k \in V_{\mathrm{lib}}^*$$

- **Algorithm:** We can use any sampling algorithm
  - Random sampling
  - Reinforcement learning (Monte Carlo tree search)

Unit Test Synthesis

Unit Test Synthesis

$$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}} \rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow \boldsymbol{r}_{\mathrm{get}}$$

$$\mathrm{this}_{\mathrm{set}} \xrightarrow{\text{alias}} \mathrm{this}_{\mathrm{get}} \quad \Rightarrow \quad \mathrm{ob}_{\mathrm{set}} \xrightarrow{\text{alias}} r_{\mathrm{get}}$$

Unit Test Synthesis

$$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}} \rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow \boldsymbol{r}_{\mathrm{get}}$$

$$\mathrm{this}_{\mathrm{set}} \xrightarrow{\text{alias}} \mathrm{this}_{\mathrm{get}} \quad \Rightarrow \quad \mathrm{ob}_{\mathrm{set}} \xrightarrow{\text{alias}} r_{\mathrm{get}}$$

```
void test():
```

Unit Test Synthesis

$$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}} \rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow \boldsymbol{r}_{\mathrm{get}}$$

$$\mathrm{this}_{\mathrm{set}} \xrightarrow{\text{alias}} \mathrm{this}_{\mathrm{get}} \quad \Rightarrow \quad \mathrm{ob}_{\mathrm{set}} \xrightarrow{\text{alias}} r_{\mathrm{get}}$$

```
void test():
    ??.set(??);
    ?? = ??.get();
```

Unit Test Synthesis

$$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}} \rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow \boldsymbol{r}_{\mathrm{get}}$$

$$\mathrm{this}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} \mathrm{this}_{\mathrm{get}} \quad \Rightarrow \quad \mathrm{ob}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} r_{\mathrm{get}}$$

```
void test():
    box.set(??);
    ?? = box.get();
```

Unit Test Synthesis

$$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}} \rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow \boldsymbol{r}_{\mathrm{get}}$$

$$\mathrm{this}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} \mathrm{this}_{\mathrm{get}} \quad \Rightarrow \quad \mathrm{ob}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} r_{\mathrm{get}}$$

```
void test():
    Box box = new Box();
    box.set(??);
    ?? = box.get();
```

Unit Test Synthesis

$$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}} \rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow \boldsymbol{r}_{\mathrm{get}}$$

$$\mathrm{this}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} \mathrm{this}_{\mathrm{get}} \quad \Rightarrow \quad \mathrm{ob}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} r_{\mathrm{get}}$$

```
void test():
    Box box = new Box();
    box.set(in);
    ?? = box.get();
```

Unit Test Synthesis

$$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}} \rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow \boldsymbol{r}_{\mathrm{get}}$$

$$\mathrm{this}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} \mathrm{this}_{\mathrm{get}} \quad \Rightarrow \quad \mathrm{ob}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} r_{\mathrm{get}}$$

```
void test():
    Object in = new Object();
    Box box = new Box();
    box.set(in);
    ?? = box.get();
```

Unit Test Synthesis

$$\text{ob}_{\text{set}} \dashrightarrow \text{this}_{\text{set}} \rightarrow \text{this}_{\text{get}} \dashrightarrow \boldsymbol{r}_{\text{get}}$$

$$\text{this}_{\text{set}} \xrightarrow{\text{alias}} \text{this}_{\text{get}} \quad \Rightarrow \quad \text{ob}_{\text{set}} \xrightarrow{\text{alias}} r_{\text{get}}$$

```
void test():
    Object in = new Object();
    Box box = new Box();
    box.set(in);
    Object out = box.get();
```

Unit Test Synthesis

$$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}} \rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow \boldsymbol{r}_{\mathrm{get}}$$

$$\mathrm{this}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} \mathrm{this}_{\mathrm{get}} \quad \Rightarrow \quad \mathrm{ob}_{\mathrm{set}} \xrightarrow{\mathrm{alias}} r_{\mathrm{get}}$$

```
void test():
    Object in = new Object();
    Box box = new Box();
    box.set(in);
    Object out = box.get();
    return in == out;
```

Unit Test Synthesis

- **Guarantee:** Unit test returns true ⇒ candidate specification is precise
  - Converse is not true!
  - Works well in practice

Generalizing to a Regular Set

Generalizing to a Regular Set

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\rightarrow \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$
$\rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

…

Inputs: Positive examples S_0

Generalizing to a Regular Set

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\to \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

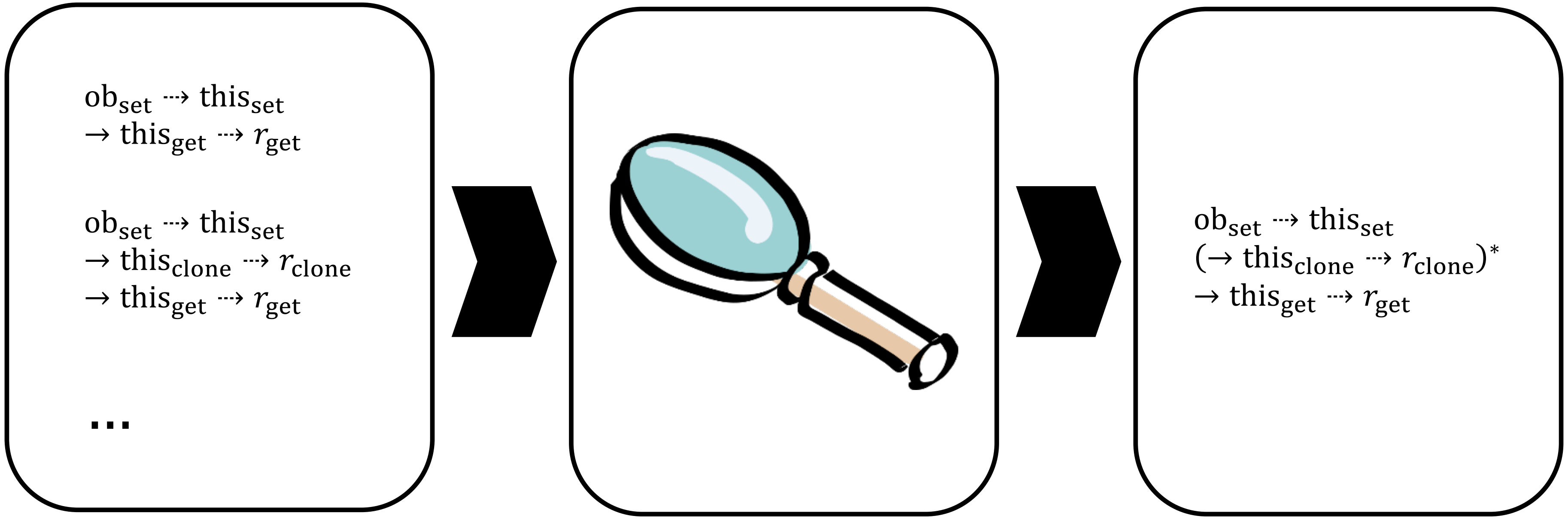
$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\to \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$
$\to \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

…

Inputs: Positive examples S_0

Generalization: Language learning based on RPNI

Generalizing to a Regular Set



$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$\rightarrow \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}}$
$\rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

…

$\mathrm{ob}_{\mathrm{set}} \dashrightarrow \mathrm{this}_{\mathrm{set}}$
$(\rightarrow \mathrm{this}_{\mathrm{clone}} \dashrightarrow r_{\mathrm{clone}})^*$
$\rightarrow \mathrm{this}_{\mathrm{get}} \dashrightarrow r_{\mathrm{get}}$

Inputs: Positive examples S_0

Generalization: Language learning based on RPNI

Output: Regular set of path specifications

Roadmap

- Motivating example
- Path specifications
- Inference algorithm
- **Evaluation**

Evaluation

- Focus on 12 most commonly used classes in the Java Collections API

- Comparisons
  - **Inferred specs:** Specifications inferred by our algorithm
  - **Ground truth specs:** Handwritten ground truth specifications (1700 LOC)
  - **Implementation:** The library implementation bytecode

- Metric for evaluating points-to analysis

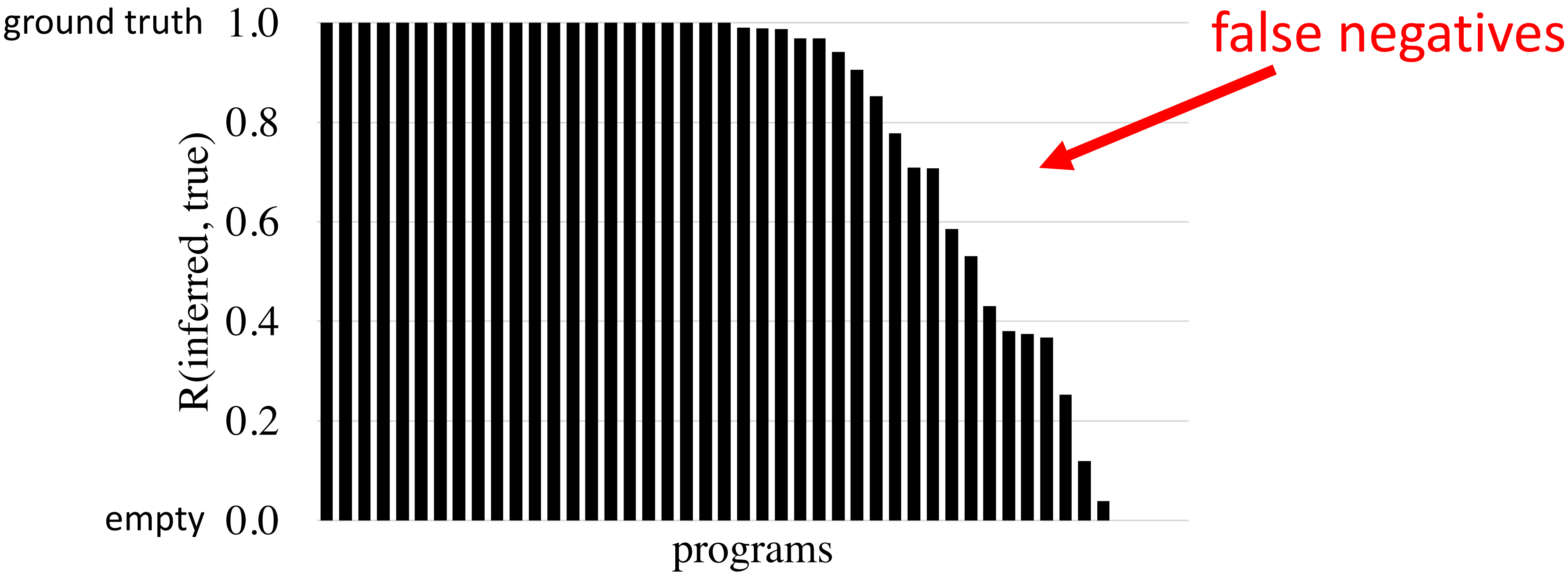
$$R(S, S') = \frac{\#\mathrm{pt}(S) - \#\mathrm{pt}(\emptyset)}{\#\mathrm{pt}(S') - \#\mathrm{pt}(\emptyset)}$$

- Benchmark of 46 programs

Inferred vs. Ground Truth (Specifications)

- Measured precision/recall on 50 most frequently used library functions
  - **Precision:** 100%
  - **Recall:** 97%

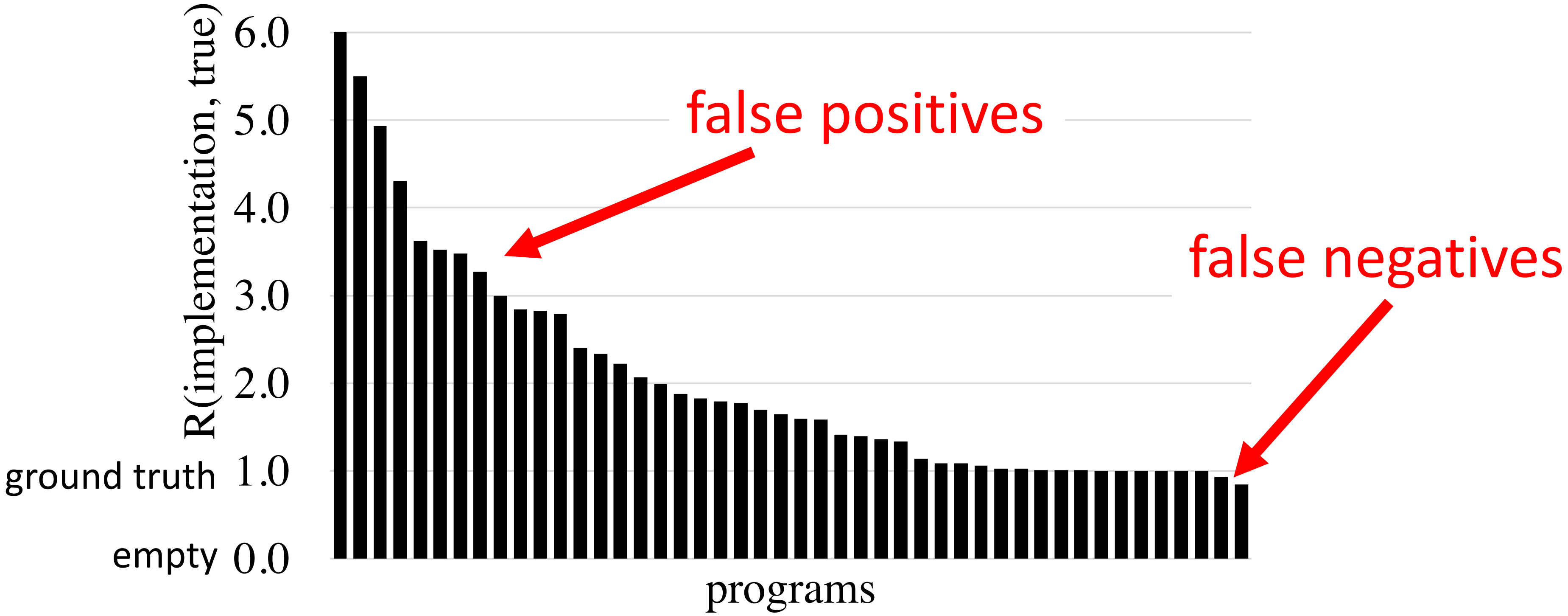
Inferred vs. Ground Truth (Points-To)



Average FN rate of inferred: 24%

Median FN rate of inferred: 1%

Implementation vs. Ground Truth



Average FP rate of implementation: 62%
Median FP rate of implementation: 115%

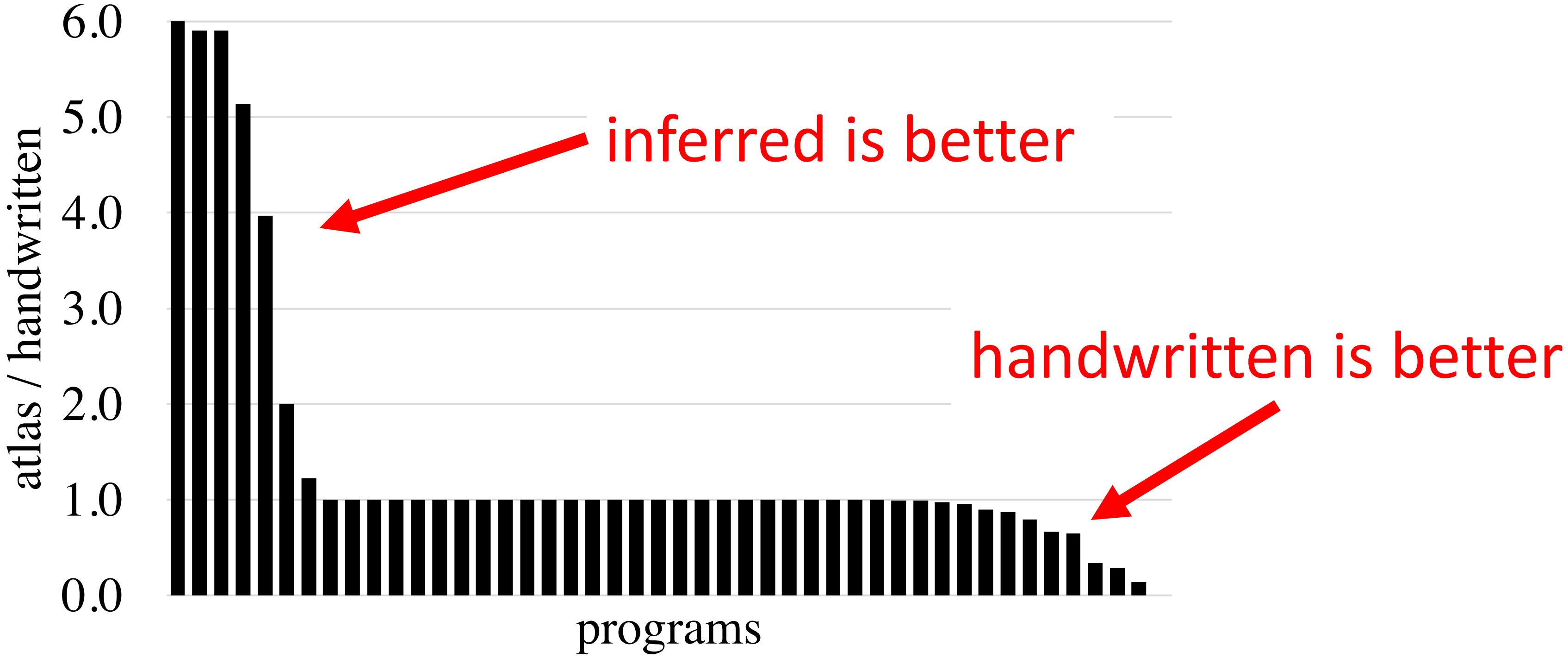
Conclusion

Specification inference can substantially improve the usability of static analysis tools

Inferred vs. Prior

- 878 inferred vs. 159 prior
- 89% recall

Inferred vs. Existing (Taint Flows)



inferred is better

handwritten is better

inferred finds 52% more information flows