Beyond Deductive Inference in Program Analysis

Osbert Bastani

Stanford University

Thesis Defense

October 2017

80 KLOC

10 MLOC

Linux

20.2 MLOC

Google

2 BLOC

coverity®

A Synopsys Company

100 LOC

1 bug

Program analysis tools help developers write correct software coverity®

A Synopsys Company

sage

SLAM

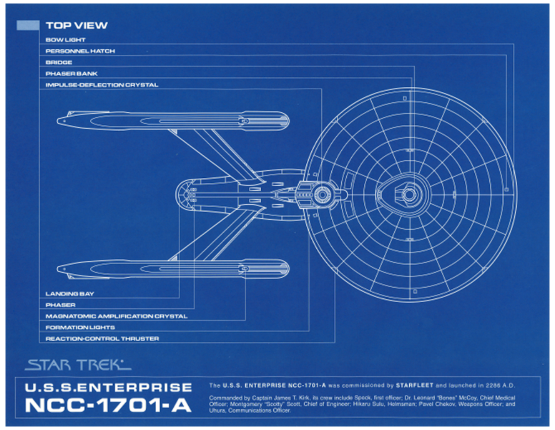
american fuzzy lop

...

e

:an
op

```
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];
  // ...
}
```



TOP VIEW
BOW LIGHT
PERSONNEL HATCH
BRIDGE
PHASER BANK
IMPULSE-DEFLECTION CRYSTAL
LANDING BAY
PHASER
MAGNATOMIC AMPLIFICATION CRYSTAL
FORMATION LIGHTS
REACTION-CONTROL THRUSTER
STAR TREK
U.S.S.ENTERPRISE
NCC-1701-A
The U.S.S. ENTERPRISE NCC-1701-A was commissioned by STARFLEET and launched in 2286 A.D. Commanded by Captain James T. Kirk, its crew include Spock, first officer; Dr. Leonard "Bones" McCoy, Chief Medical Officer; Montgomery "Scotty" Scott, Chief of Engineer; Hikaru Sulu, Helmsman; Pavel Chekov, Weapons Officer; and Uhura, Communications Officer.

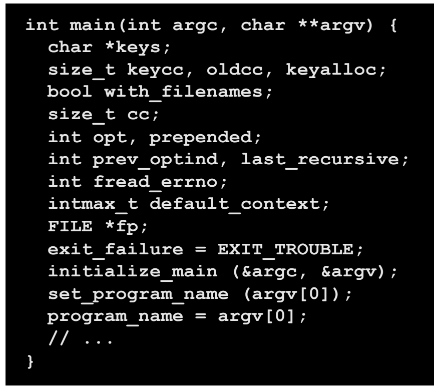
coverity®
A Synopsys Company

sage

SLAM

american
fuzzy lop

...



```
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];
  // ...
}
```

- No memory bugs
- Not malicious

coverity
A Synopsys Company

sage

SLAM

american
fuzzy lop

...

```
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];
  // ...
}
```

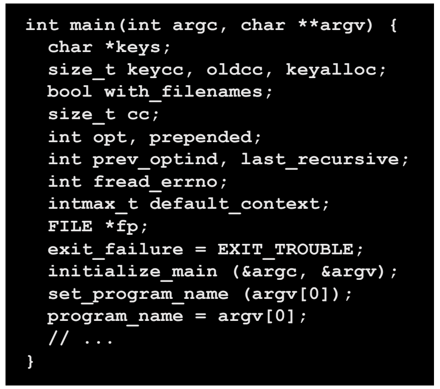
- No memory bugs
- Not malicious

Static

- Analyze

Dynamic:

- Execute



```
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];
  // ...
}
```

- No memory bugs
- Not malicious

Static

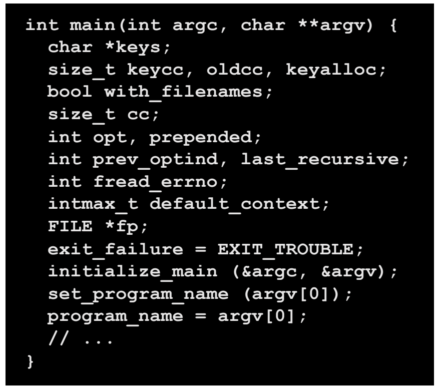
- Analyze

Dynamic:

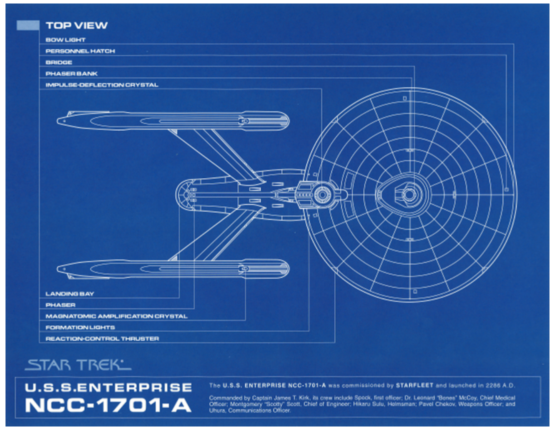
- Execute

- Correctness proof
- Bugs

Problem: Missing information about inputs



```
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];
  // ...
}
```


TOP VIEW
BOW LIGHT
PERSONNEL HATCH
BRIDGE
PHASER BANK
IMPULSE-DEFLECTION CRYSTAL
LANDING BAY
PHASER
MAGNATOMIC AMPLIFICATION CRYSTAL
FORMATION LIGHTS
REACTION-CONTROL THRUSTER
STAR TREK
U.S.S.ENTERPRISE
NCC-1701-A
The U.S.S. ENTERPRISE NCC-1701-A was commissioned by STARFLEET and launched in 2286 A.D. Commanded by Captain James T. Kirk, its crew include Spock, first officer; Dr. Leonard "Bones" McCoy, Chief Medical Officer; Montgomery "Scotty" Scott, Chief of Engineer; Hikaru Sulu, Helmsman; Pavel Chekov, Weapons Officer; and Uhura, Communications Officer.

coverity®
A Synopsys Company

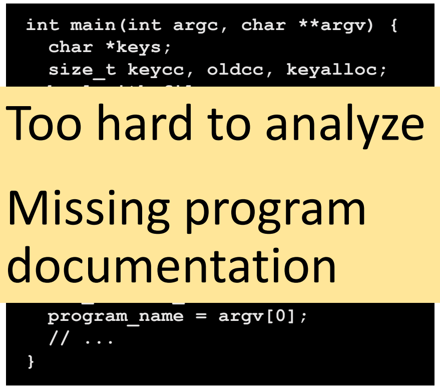
sage

SLAM

american
fuzzy lop

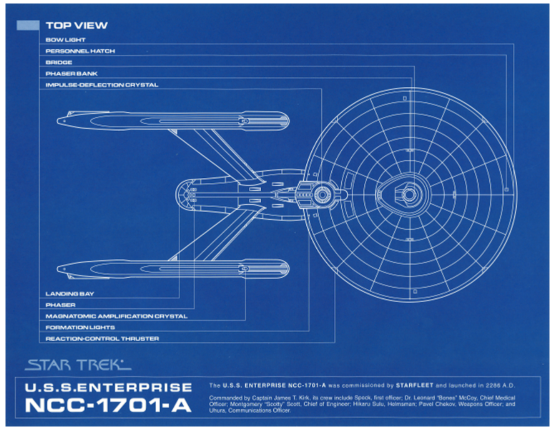
...

Problem: Missing information about inputs



```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  program_name = argv[0];
  // ...
}
```

- Too hard to analyze
- Missing program documentation



The U.S.S. ENTERPRISE NCC-1701-A was commissioned by STARFLEET and launched in 2286 A.D. Commanded by Captain James T. Kirk, its crew include Spock, first officer; Dr. Leonard "Bones" McCoy, Chief Medical Officer; Montgomery "Scotty" Scott, Chief of Engineer; Hikaru Sulu, Helmsman; Pavel Chekov, Weapons Officer; and Uhura, Communications Officer.

coverity®
A Synopsys Company

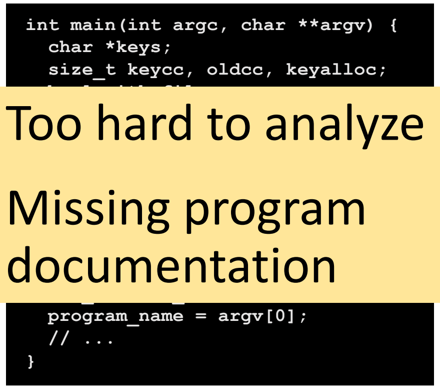
sage

SLAM

american fuzzy lop

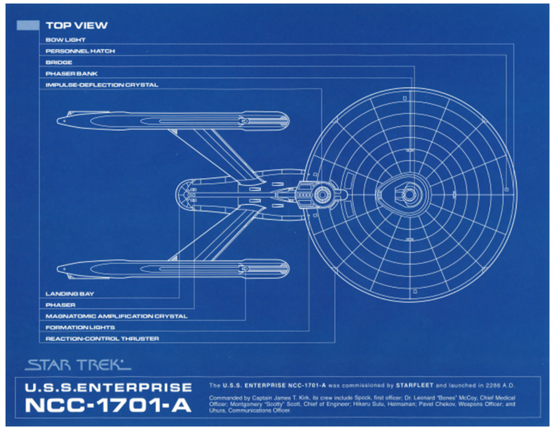
...

Problem: Missing information about inputs



```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  ...
  program_name = argv[0];
  // ...
}
```

- Too hard to analyze
- Missing program documentation



coverity A Synopsys Company

coverity A Synopsys Company

sage

SLAM

american fuzzy lop

...

Problem: Missing information about inputs

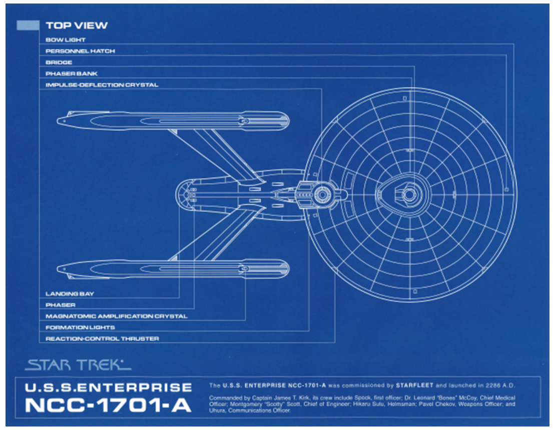
```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
```

- Too hard to analyze
- Missing program documentation

```
  program_name = argv[0];
  // ...
}
```

coverity A Synopsys Company

american fuzzy lop



coverity A Synopsys Company

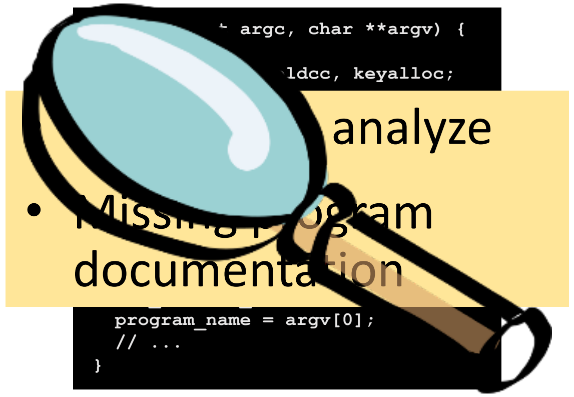
sage

SLAM

american fuzzy lop

...

Solution: Infer missing information

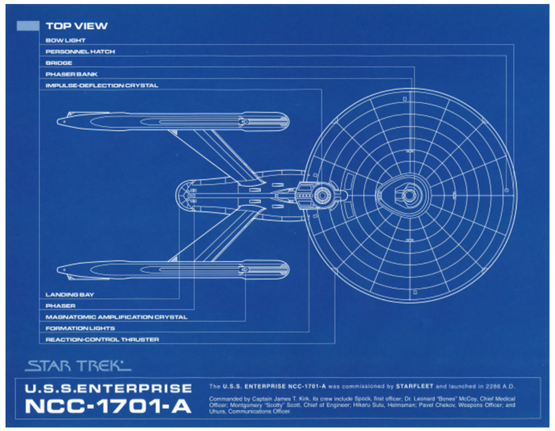


```
    argc, char **argv) {
    ldcc, keyalloc;
```

analyze

- Missing program documentation

```
  program_name = argv[0];
  // ...
}
```



coverity® A Synopsys Company

sage

SLAM

american fuzzy lop

...

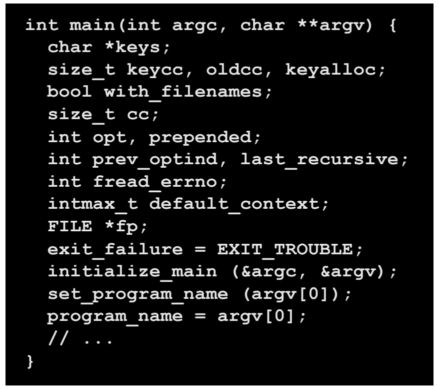
Problem: Missing code needs **summaries**
Solution: Interactively infer summaries

Problem: Fuzzer aided by **input language**
Solution: Automatically infer input language

Inferring Summaries of Missing Code

Osbert Bastani, Saswat Anand, and Alex Aiken

POPL 2015



```
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];
  // ...
}
```

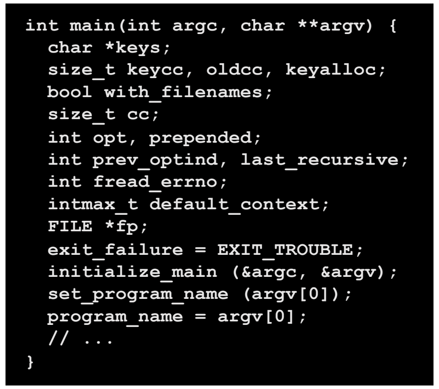
Android app

security analyst

malware?

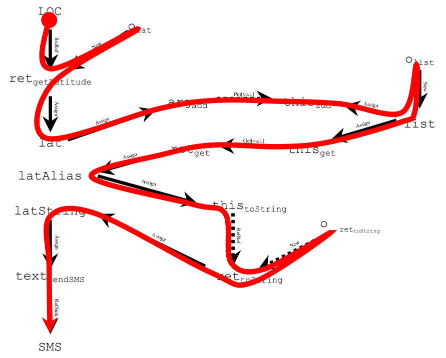
Find **malicious behaviors** using source to sink **taint flows**

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];
  // ...
}
```

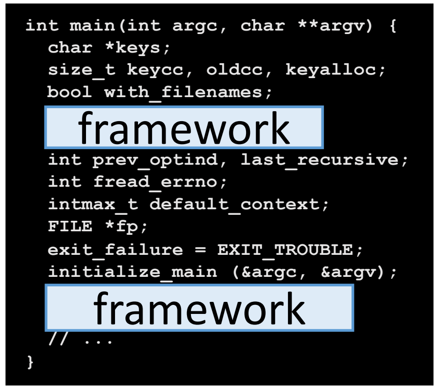
Android app



taint analysis

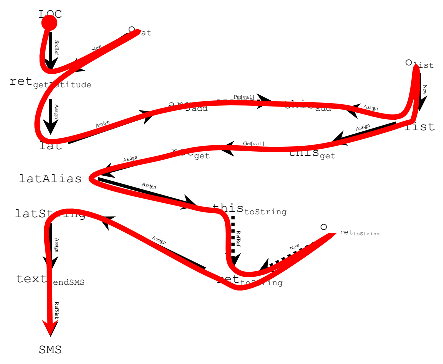
location → Internet
SMS → Internet

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
  // ...
}
```

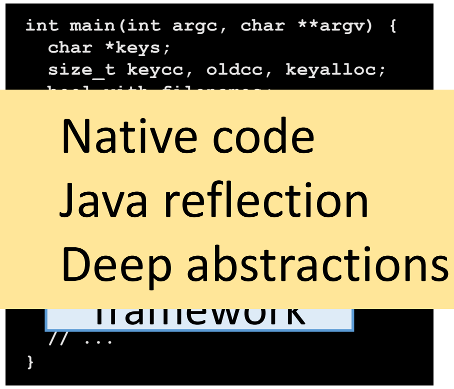
Android app



taint analysis

location → Internet
SMS → Internet

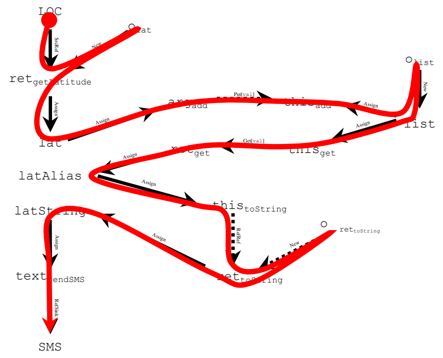
malicious behaviors



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

- Native code
- Java reflection
- Deep abstractions

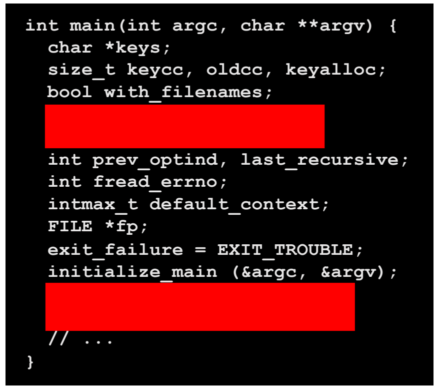
Android app



taint analysis

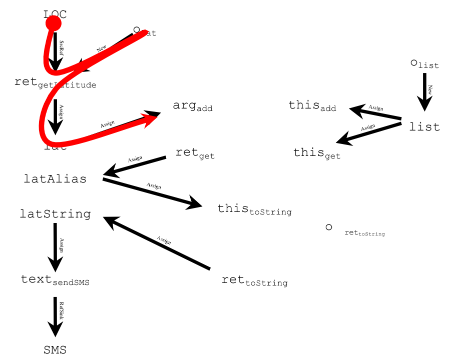
location → Internet
SMS → Internet

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]
  // ...
}
```

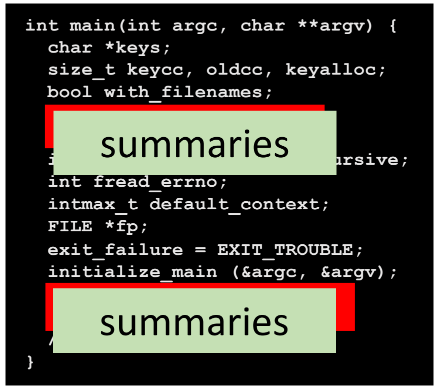
Android app



taint analysis

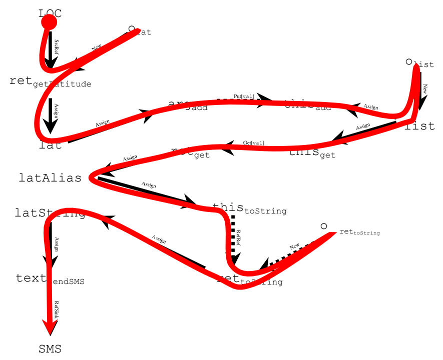
?

malicious behaviors



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

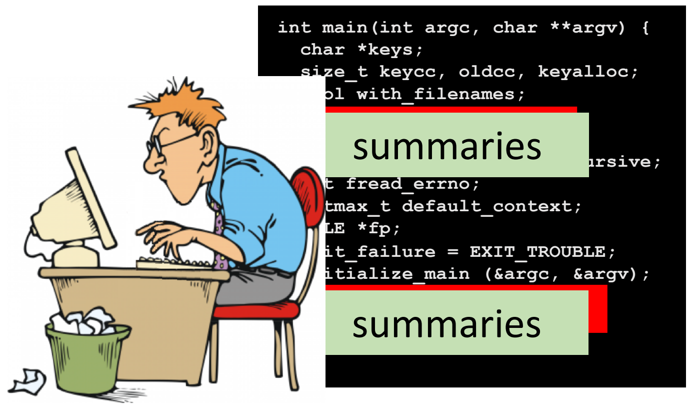
Android app



taint analysis

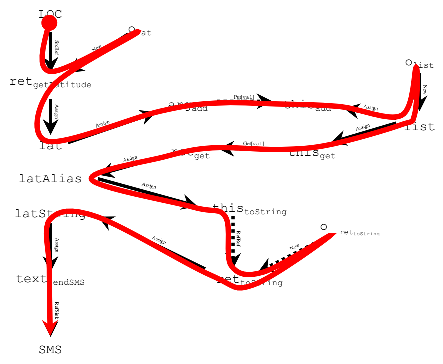
location → Internet
SMS → Internet

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);
```

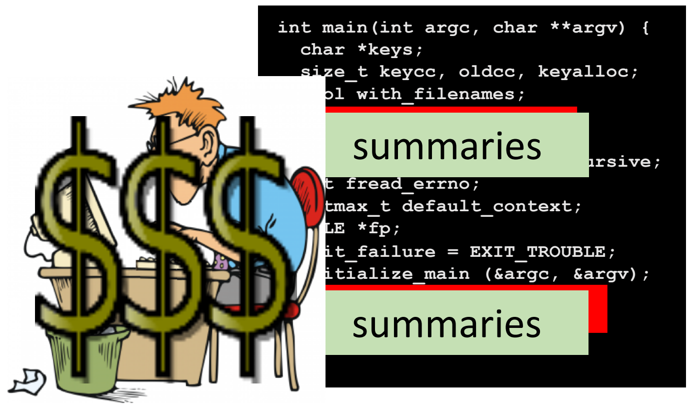
Android app



taint analysis

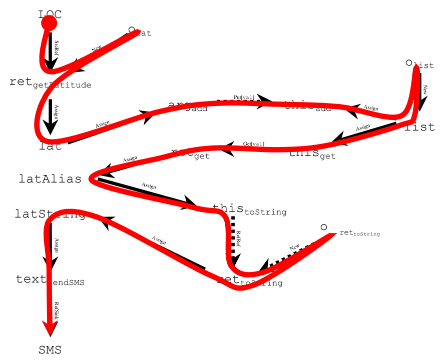
location → Internet
SMS → Internet

malicious behaviors



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

                              rsive;
   t fread_errno;
   tmax_t default_context;
   LE *fp;
   it_failure = EXIT_TROUBLE;
   itialize_main (&argc, &argv);
```

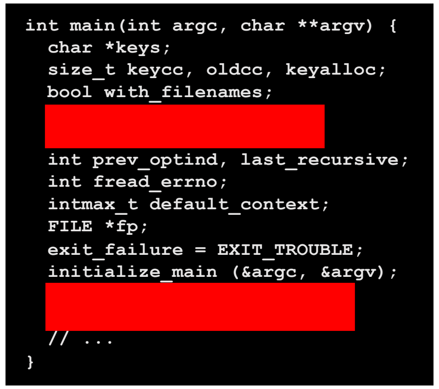
Android app



taint analysis

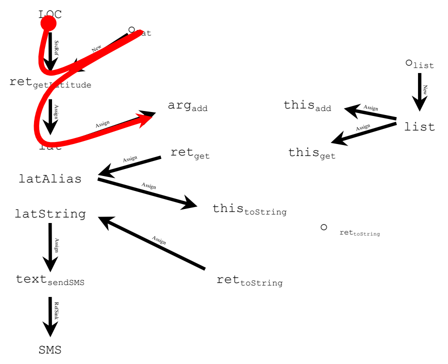
location → Internet
SMS → Internet

malicious behaviors



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

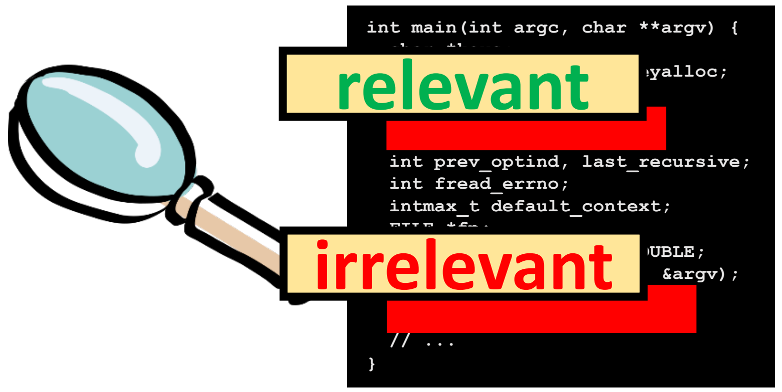
Android app



taint analysis

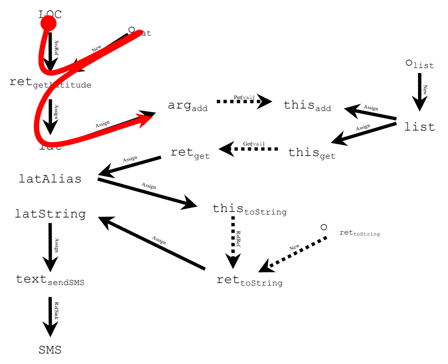
?

malicious behaviors

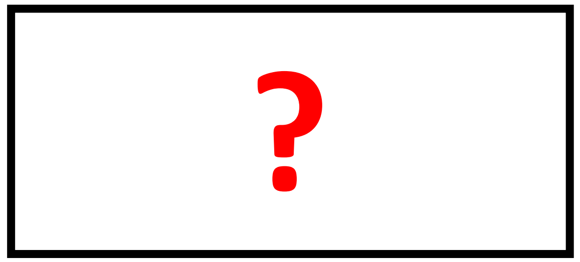


```
int main(int argc, char **argv) {
                                  yalloc;

  int prev_optind, last_recursive;
  int fread_errno;
  intmax_t default_context;

                                  UBLE;
                                  &argv);

  // ...
}
```

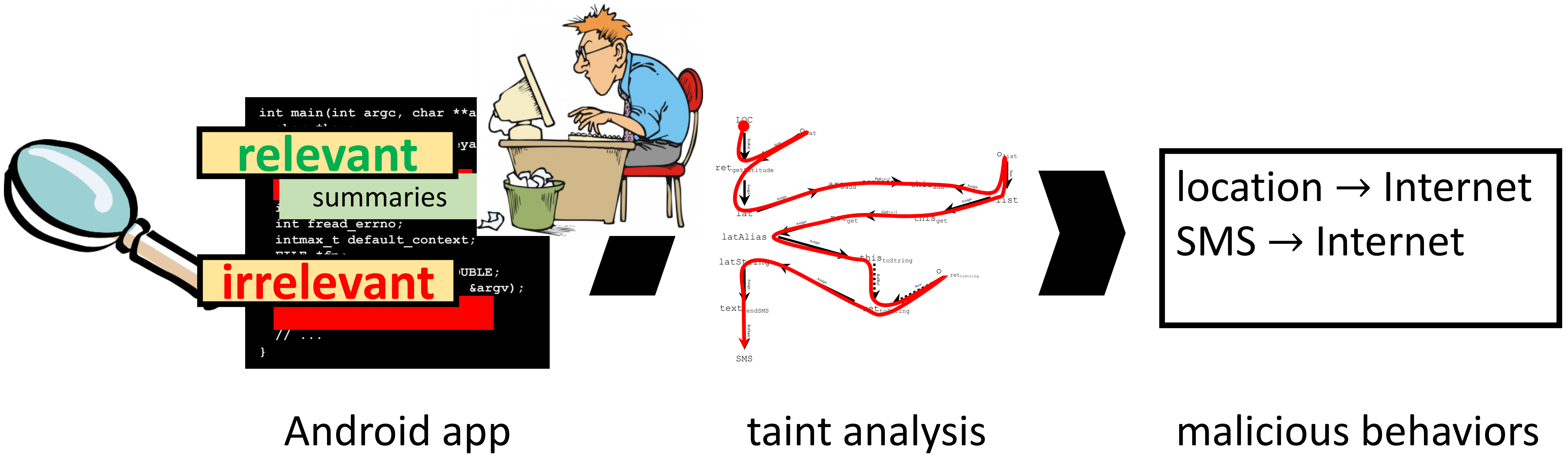
Android app



taint analysis



malicious behaviors


relevant
summaries
irrelevant
int main(int argc, char **a
int fread_errno;
intmax_t default_context;
UBLE;
&argv);
// ...
}
LOC
latAlias
SMS
location → Internet
SMS → Internet
Android app
taint analysis
malicious behaviors

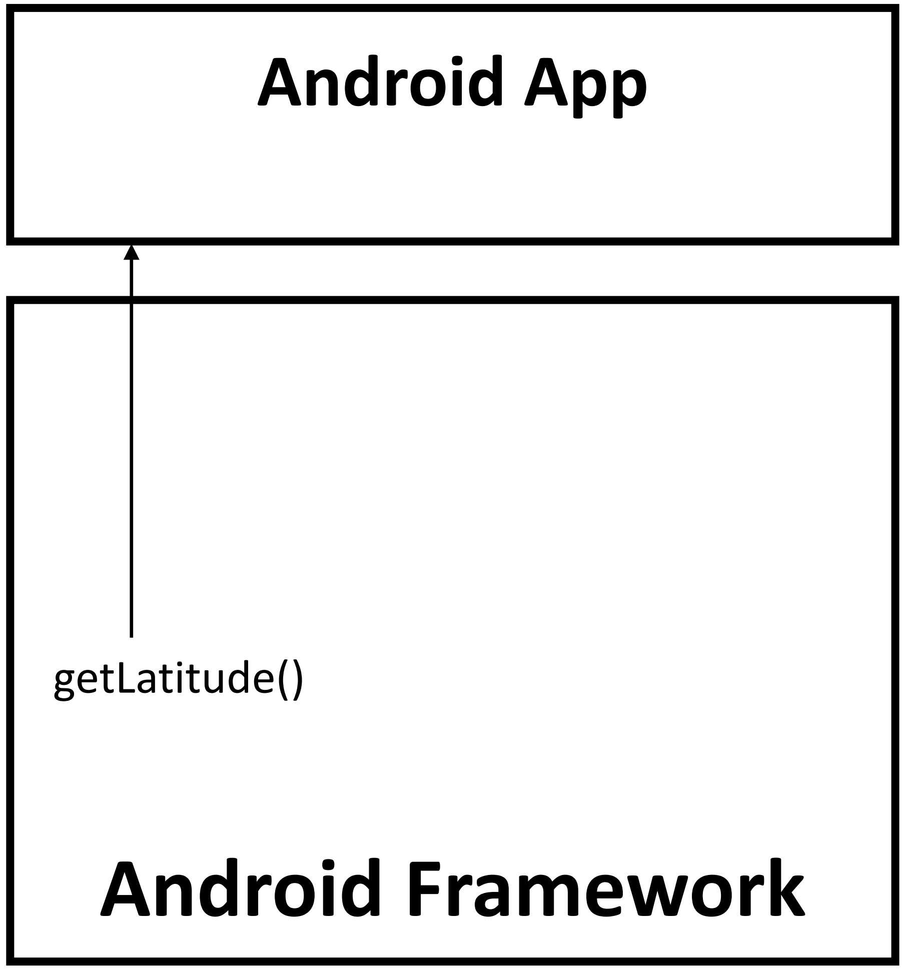
Taint Analysis

```
Double lat = getLatitude();
List list = new List();
list.add(lat);
Double latAlias = list.get(0);
String latStr = latAlias.toString();
sendHttp(latStr);
```

Android App
Android Framework

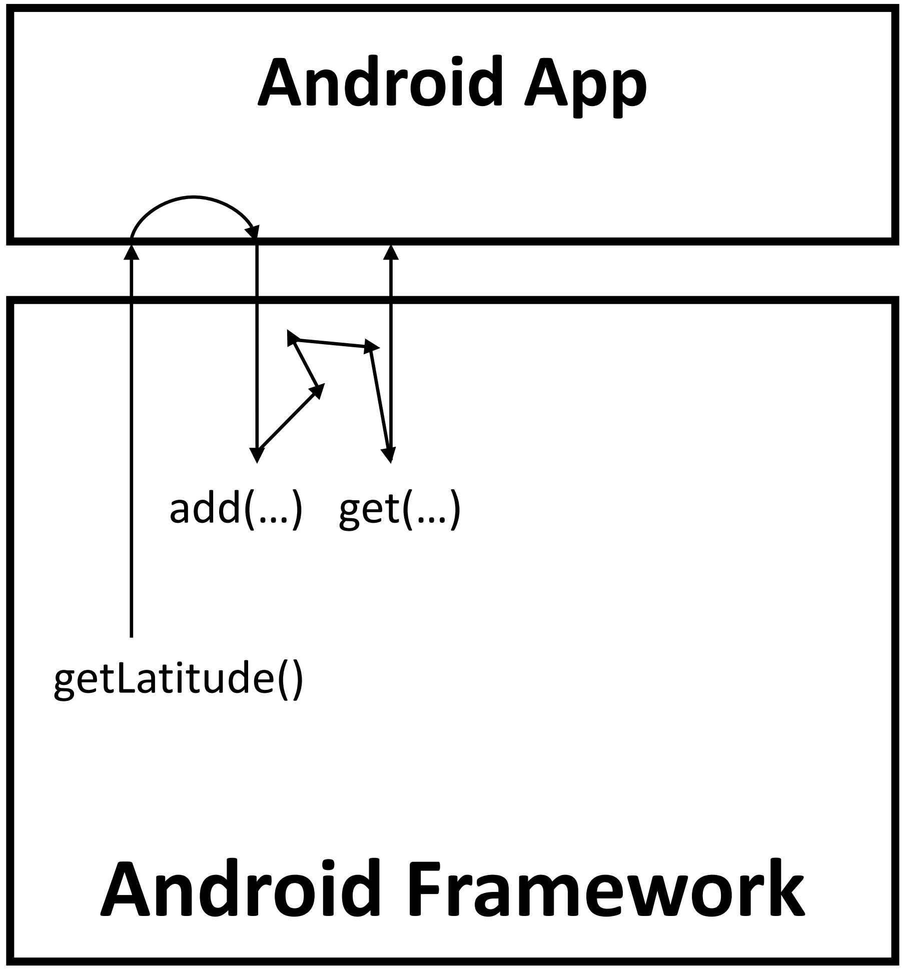
Taint Analysis

```
1. Double lat = getLatitude();
2. List list = new List();
3. list.add(lat);
4. Double latAlias = list.get(0);
5. String latStr = latAlias.toString();
6. sendHttp(latStr);
```



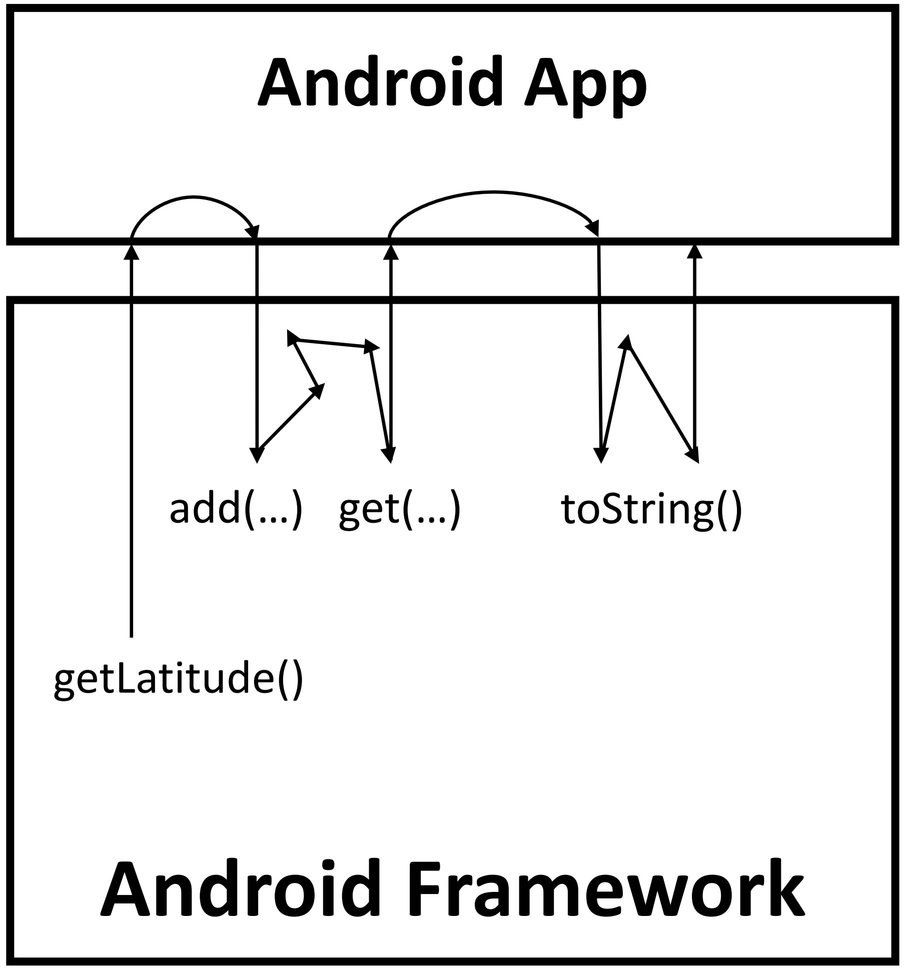
Taint Analysis

```
1. Double lat = getLatitude();
2. List list = new List();
3. list.add(lat);
4. Double latAlias = list.get(0);
5. String latStr = latAlias.toString();
6. sendHttp(latStr);
```



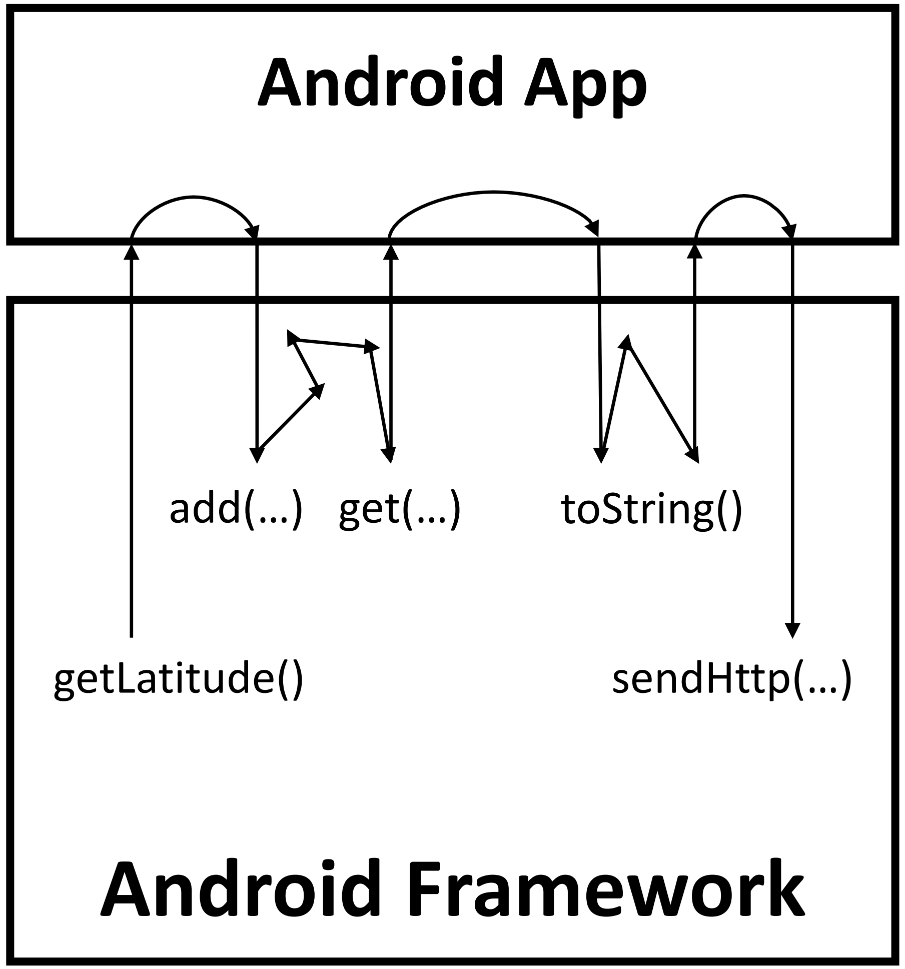
Taint Analysis

1. **Double** lat = getLatitude();
2. **List** list = **new** List();
3. list.add(lat);
4. **Double** latAlias = list.get(0);
5. **String** latStr = latAlias.toString();
6. sendHttp(latStr);



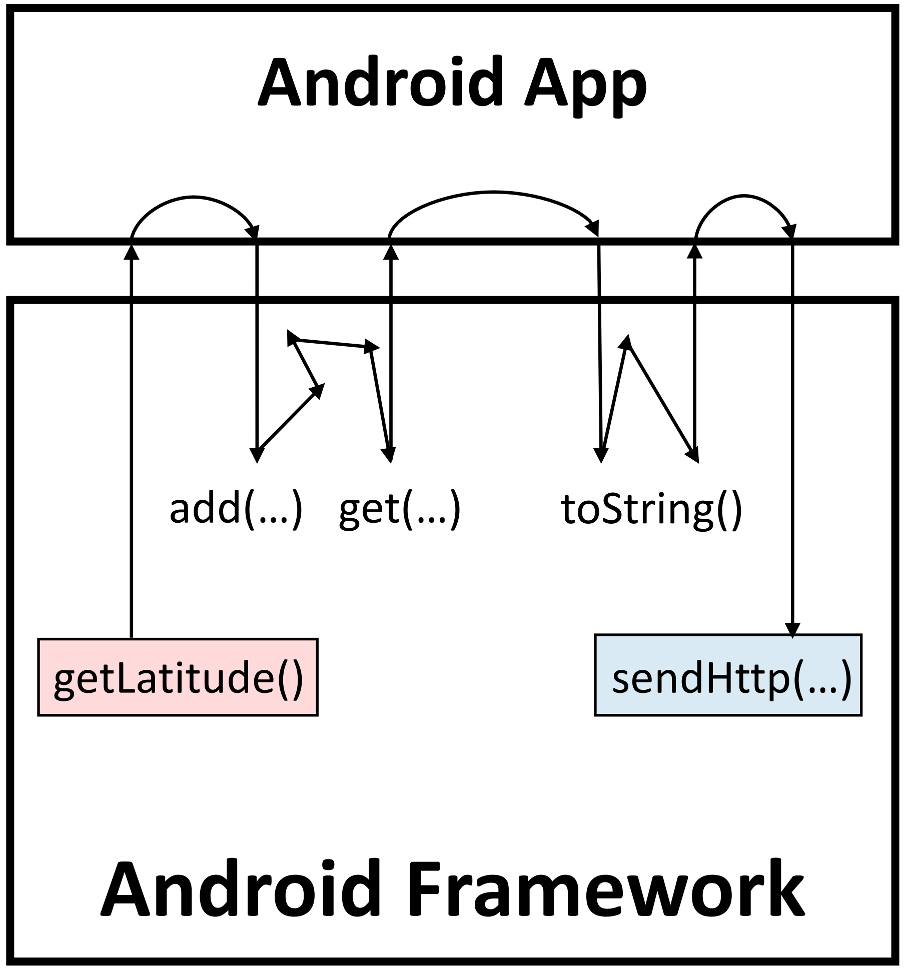
Taint Analysis

1. **Double** lat = getLatitude();
2. **List** list = **new** List();
3. list.add(lat);
4. **Double** latAlias = list.get(0);
5. **String** latStr = latAlias.toString();
6. sendHttp(latStr);



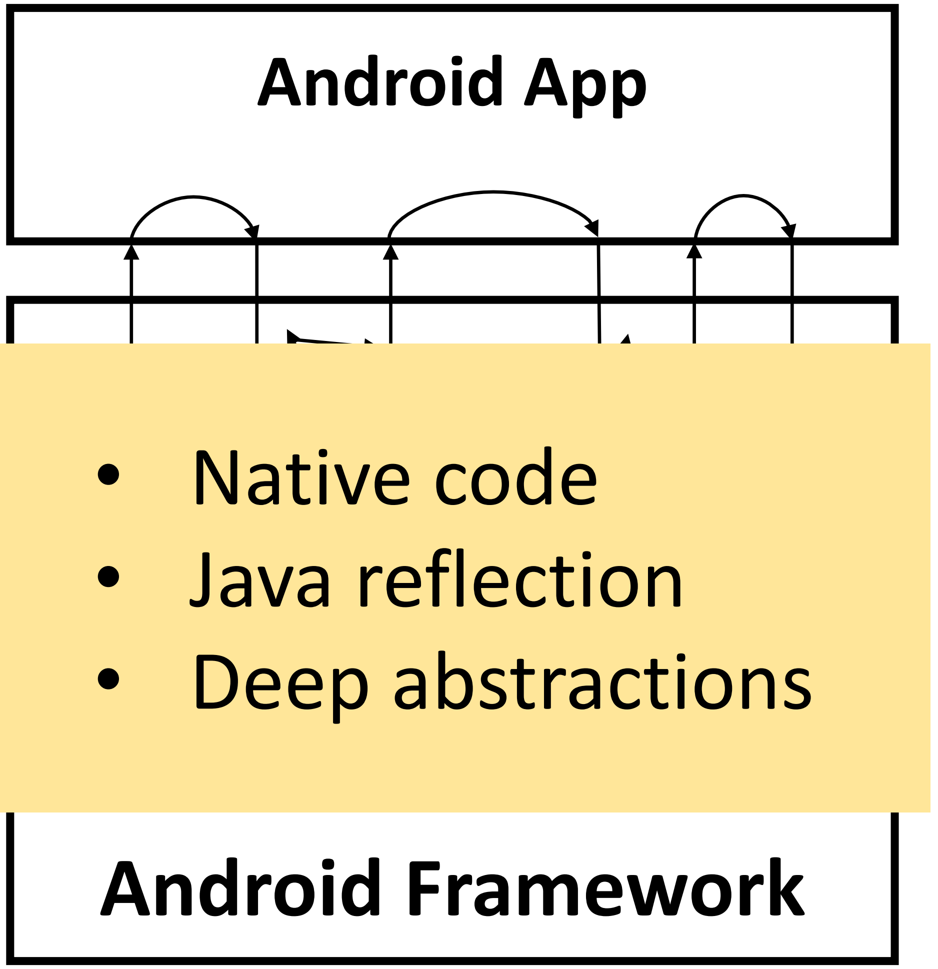
Taint Analysis

1. **Double** lat = getLatitude();
2. **List** list = **new** List();
3. list.add(lat);
4. **Double** latAlias = list.get(0);
5. **String** latStr = latAlias.toString();
6. sendHttp(latStr);



Taint Analysis

```
Double lat = getLatitude();
List list = new List();
list.add(lat);
Double latAlias = list.get(0);
String latStr = latAlias.toString();
sendHttp(latStr);
```



- Native code
- Java reflection
- Deep abstractions

Flow summary

- Describes taint transfer
- **@flow**(x, y) means "x tainted ⇒ y tainted"

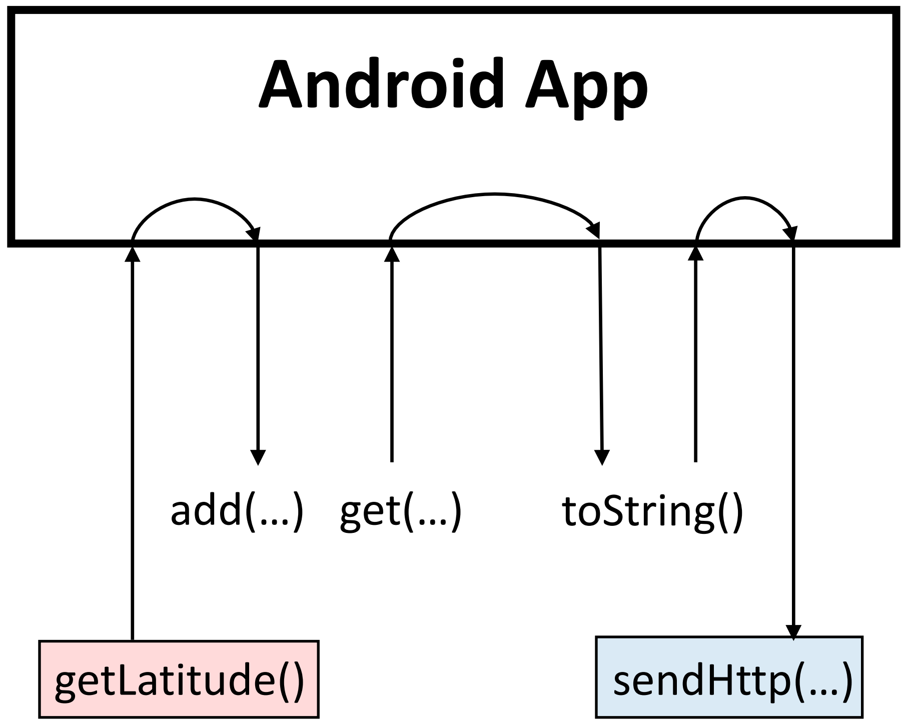
```
class Double:
    @flow(this, return)
    String toString() {}
```

Alias summary

- Describes aliasing
- **@alias**(x, y) means "x may alias y"

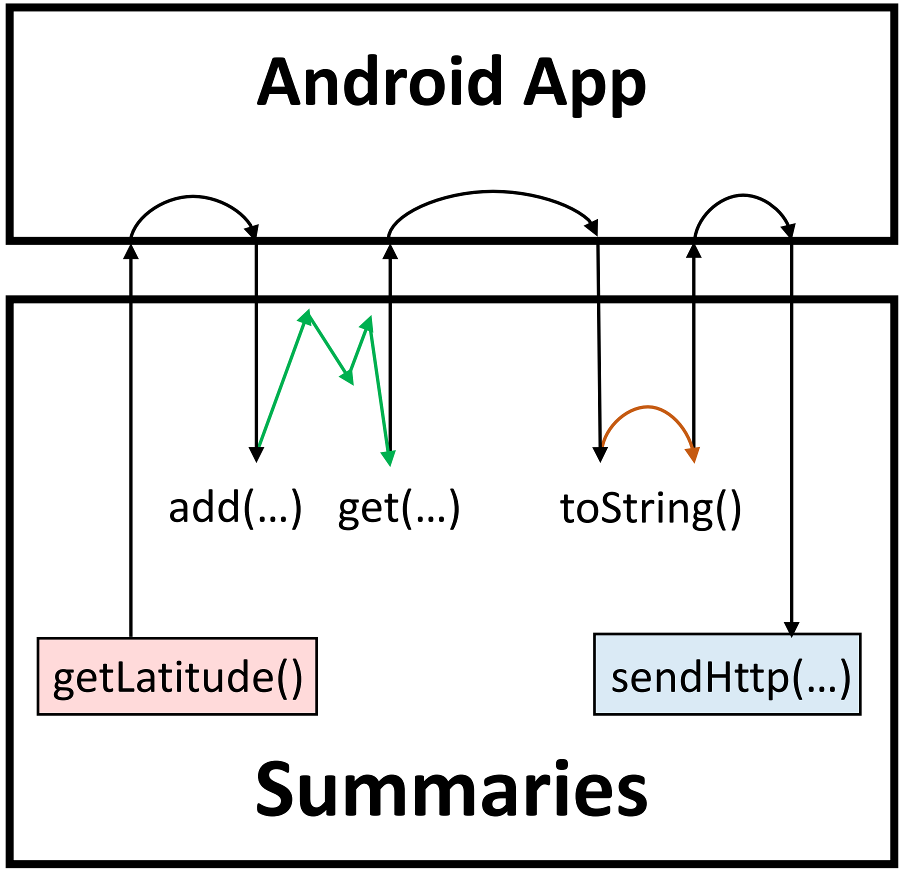
```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}
```

Taint Analysis



Taint Analysis

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



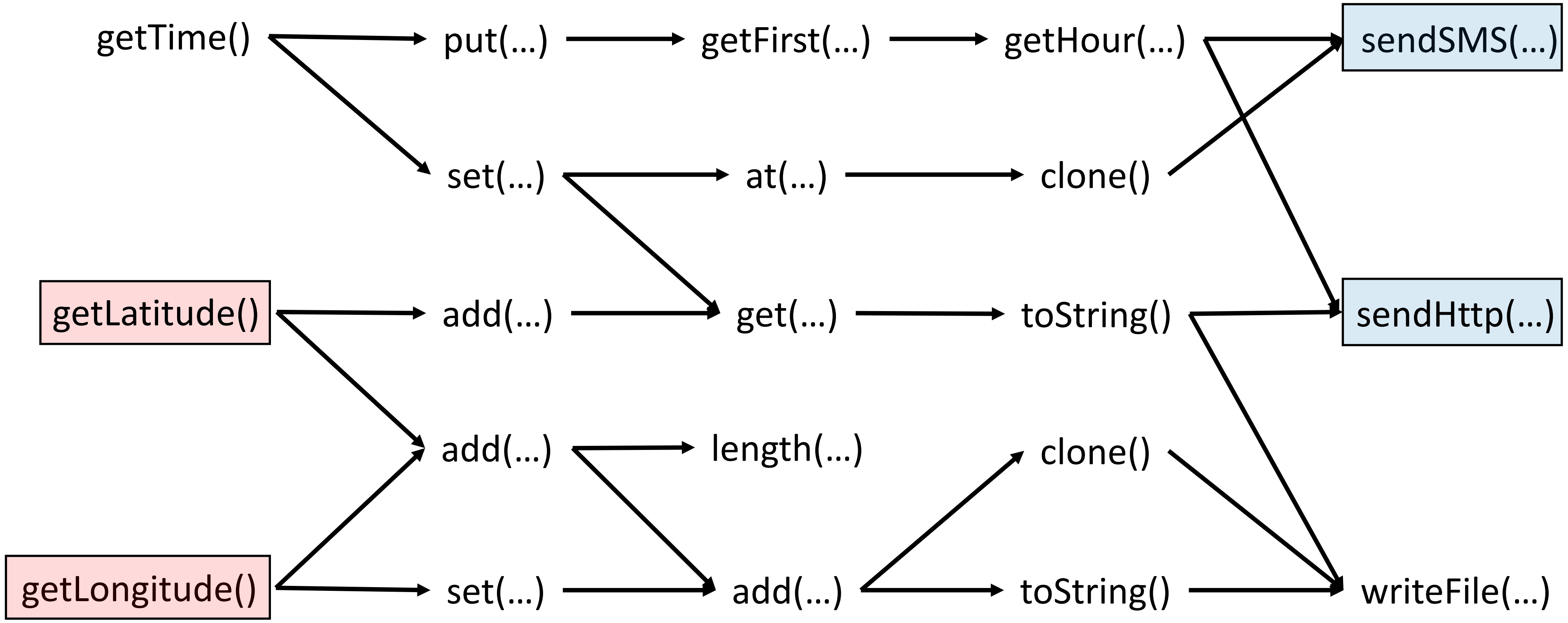
Writing summaries is time consuming:

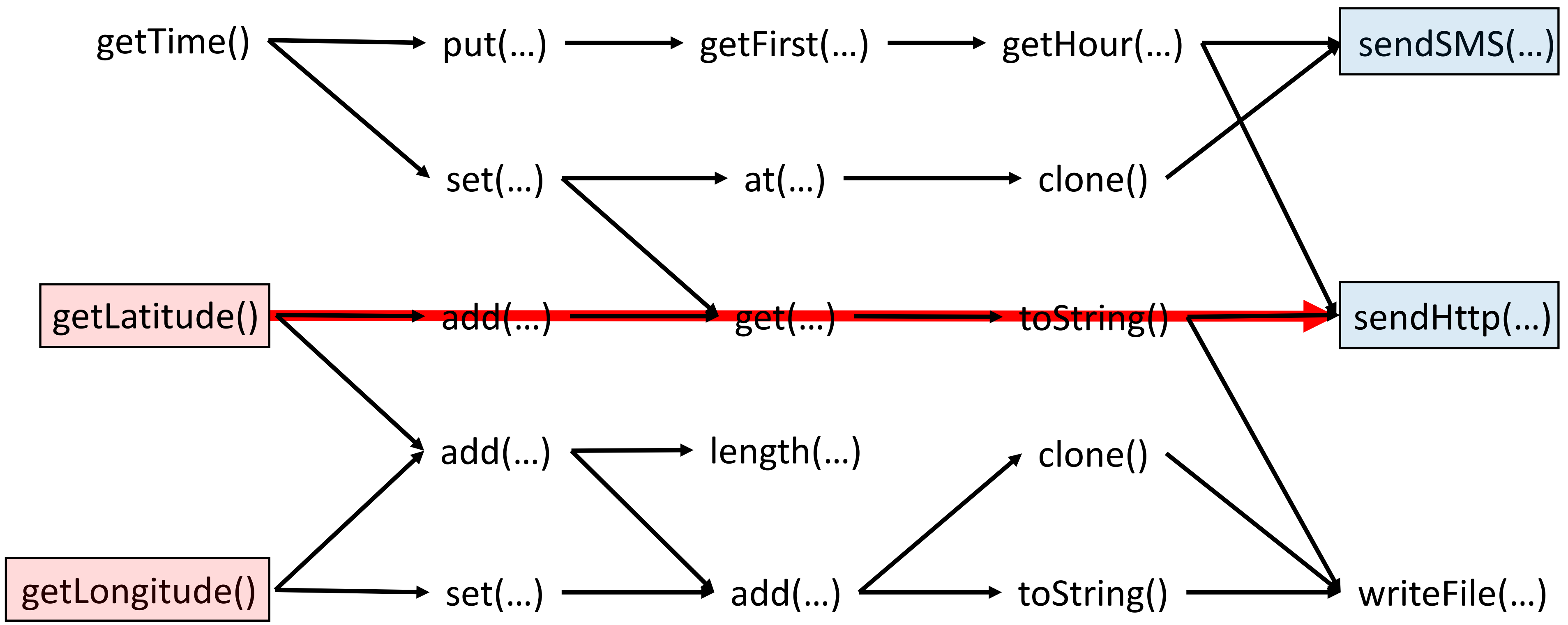
- $\approx 30{,}000$ framework methods
- $\approx 10{,}000$ needed for a typical app
- Maintenance

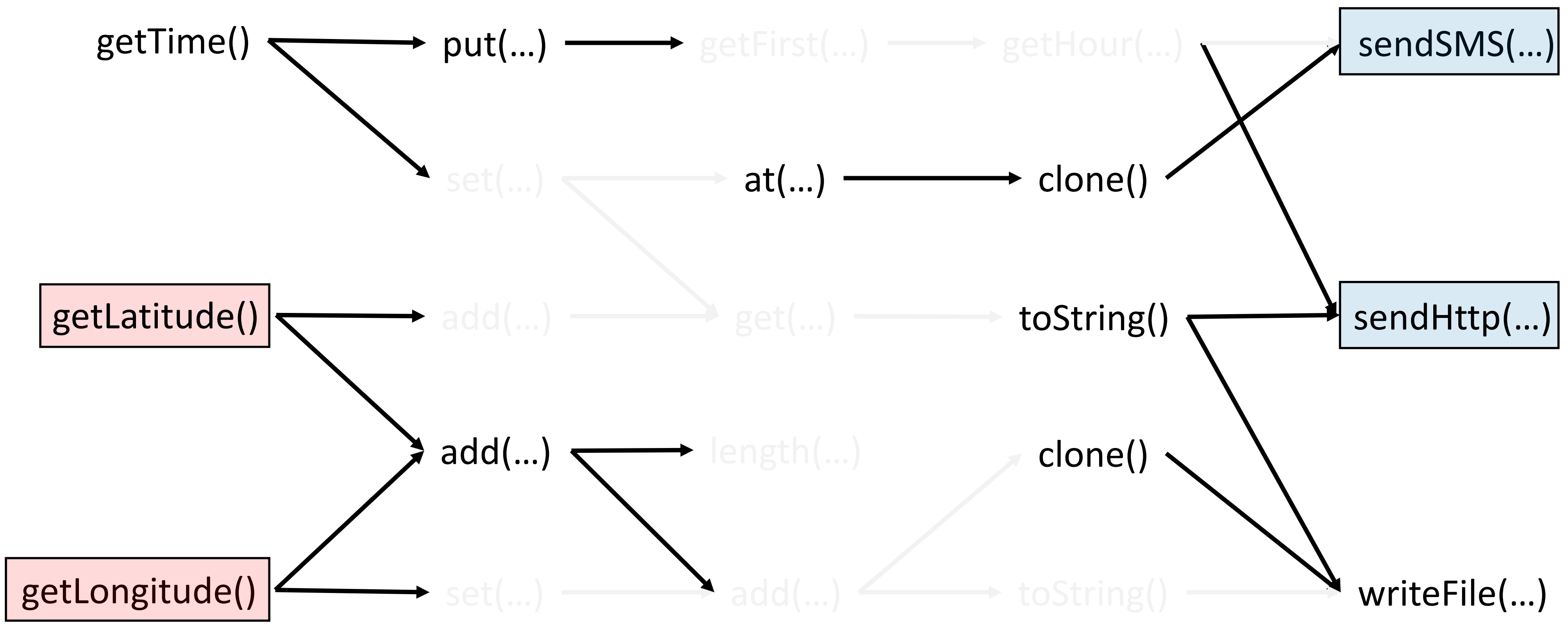
Insight: Only some methods are relevant

- > 200 apps
- About $1{,}600$ summaries

write summaries **as needed**


getTime()
put(…)
getFirst(…)
getHour(…)
sendSMS(…)
set(…)
at(…)
clone()
getLatitude()
add(…)
get(…)
toString()
sendHttp(…)
add(…)
length(…)
clone()
getLongitude()
set(…)
add(…)
toString()
writeFile(…)


getTime()
put(...)
getFirst(...)
getHour(...)
sendSMS(...)
set(...)
at(...)
clone()
getLatitude()
add(...)
get(...)
toString()
sendHttp(...)
add(...)
length(...)
clone()
getLongitude()
set(...)
add(...)
toString()
writeFile(...)


getTime()
put(...)
getFirst(...)
getHour(...)
sendSMS(...)
set(...)
at(...)
clone()
getLatitude()
add(...)
get(...)
toString()
sendHttp(...)
add(...)
length(...)
clone()
getLongitude()
set(...)
add(...)
toString()
writeFile(...)

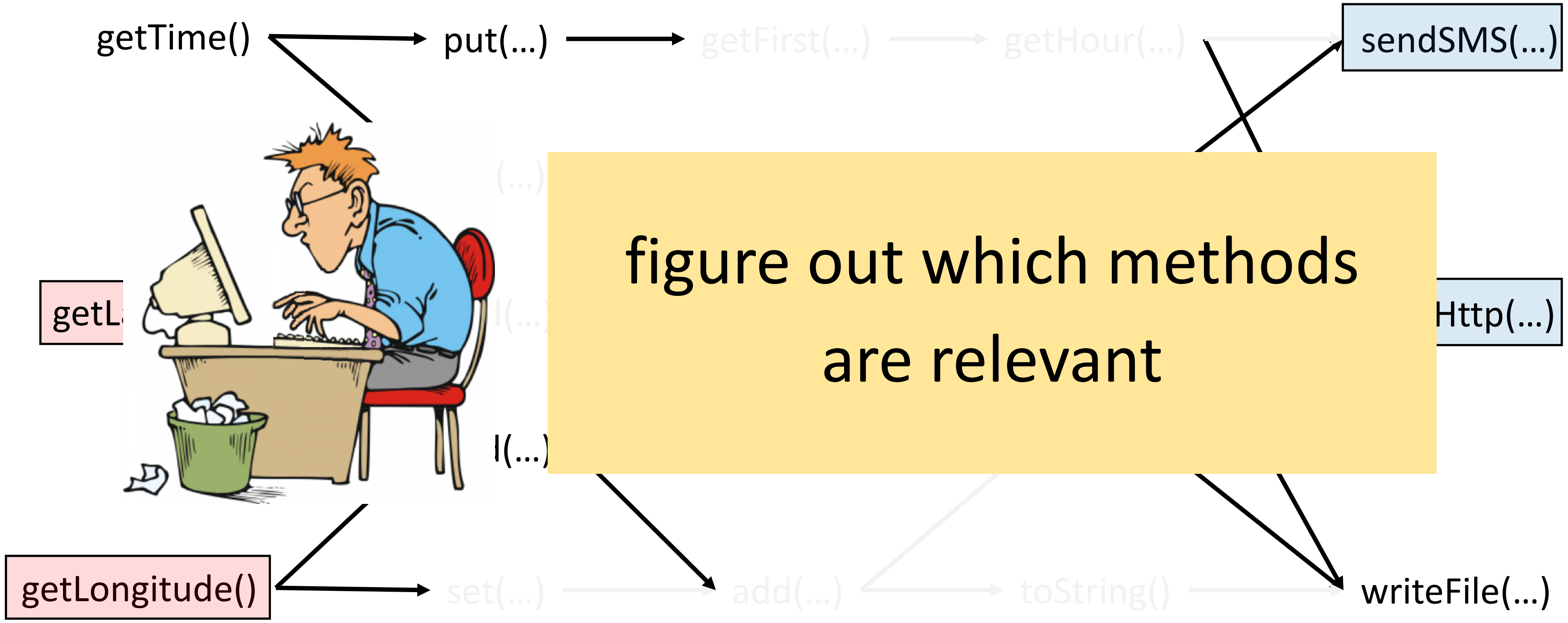
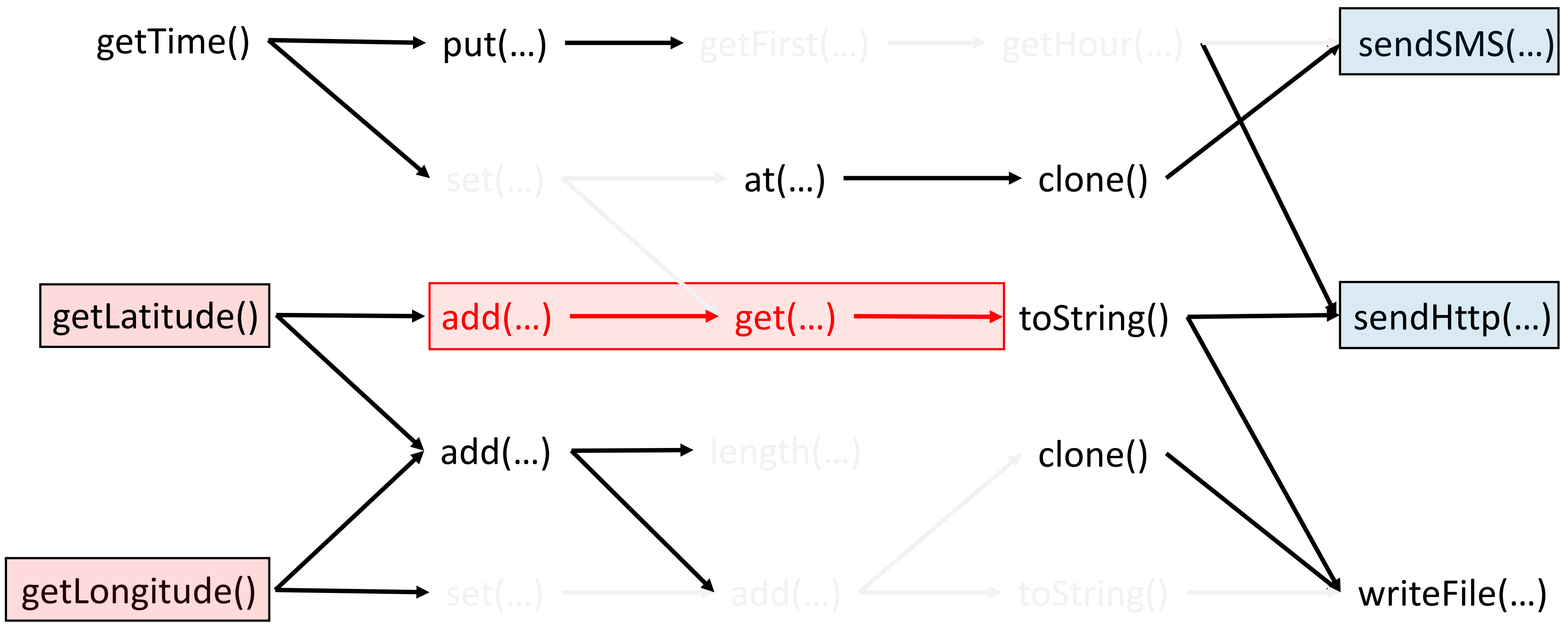


figure out which methods are relevant


getTime()
put(...)
getFirst(...)
getHour(...)
sendSMS(...)
set(...)
at(...)
clone()
getLatitude()
add(...)
get(...)
toString()
sendHttp(...)
add(...)
length(...)
clone()
getLongitude()
set(...)
add(...)
toString()
writeFile(...)

Step 1: Worst-case analysis

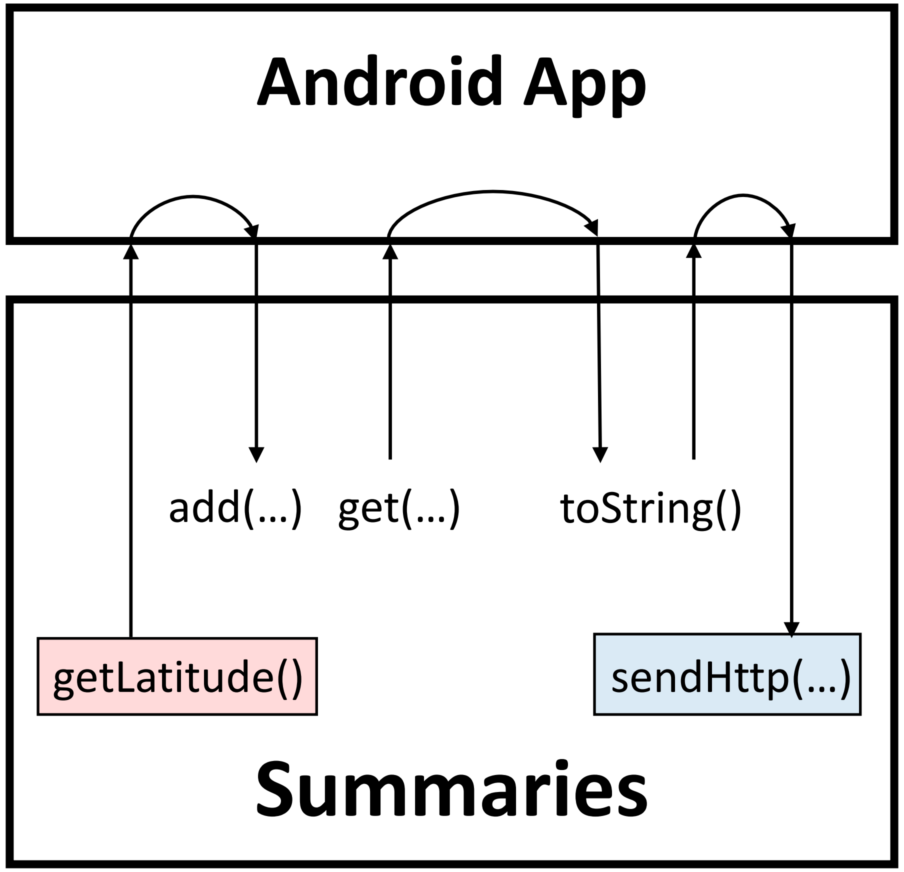
location → Internet
SMS → Internet
device ID → SMS
…

Step 2: Infer summaries

Step 3: Analyst corrections

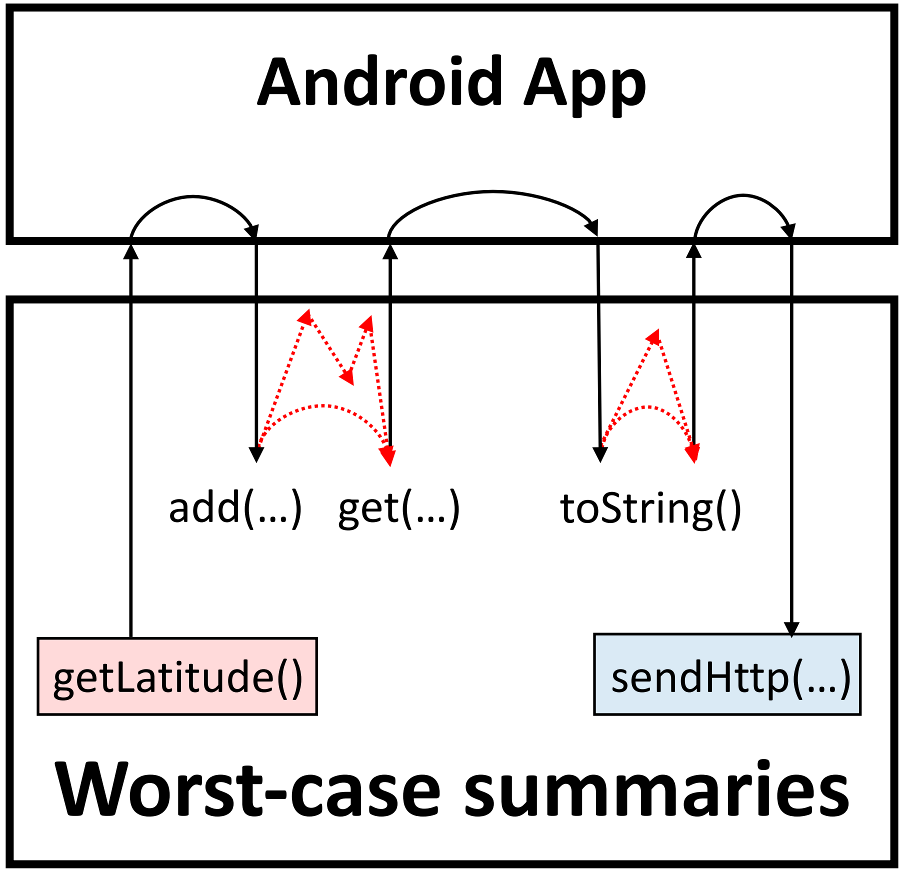
Step 1: Worst-case Analysis

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



Step 1: Worst-case Analysis

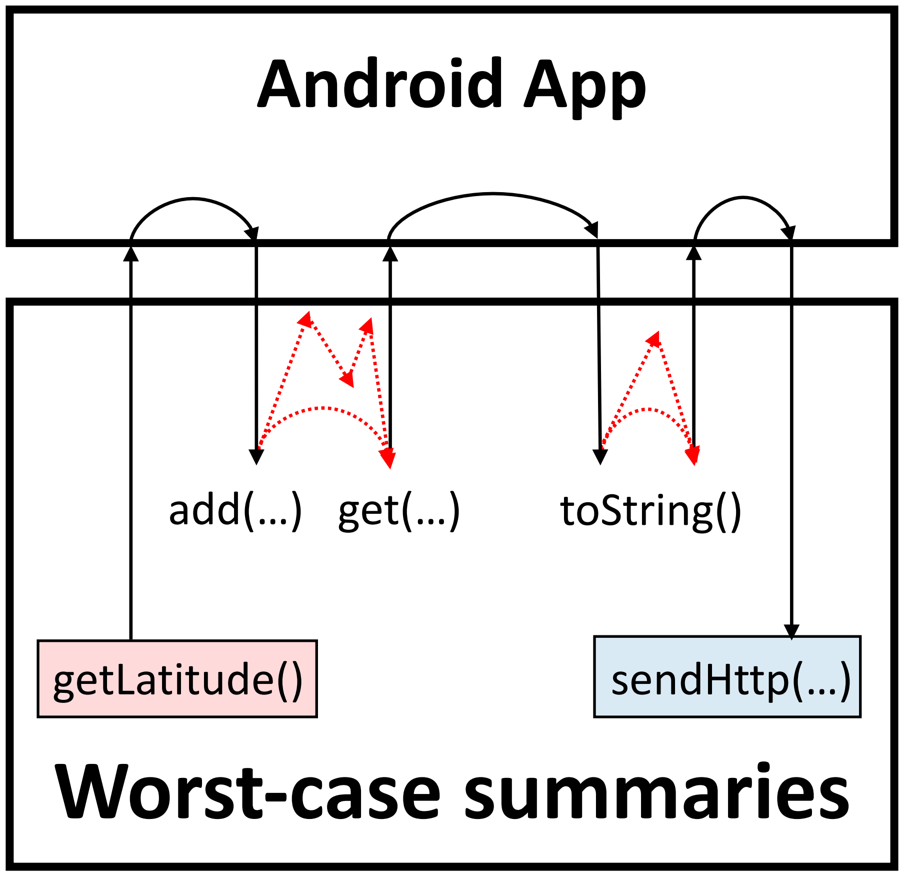
```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



location → Internet
device ID → SMS
…

Step 2: Summary Inference

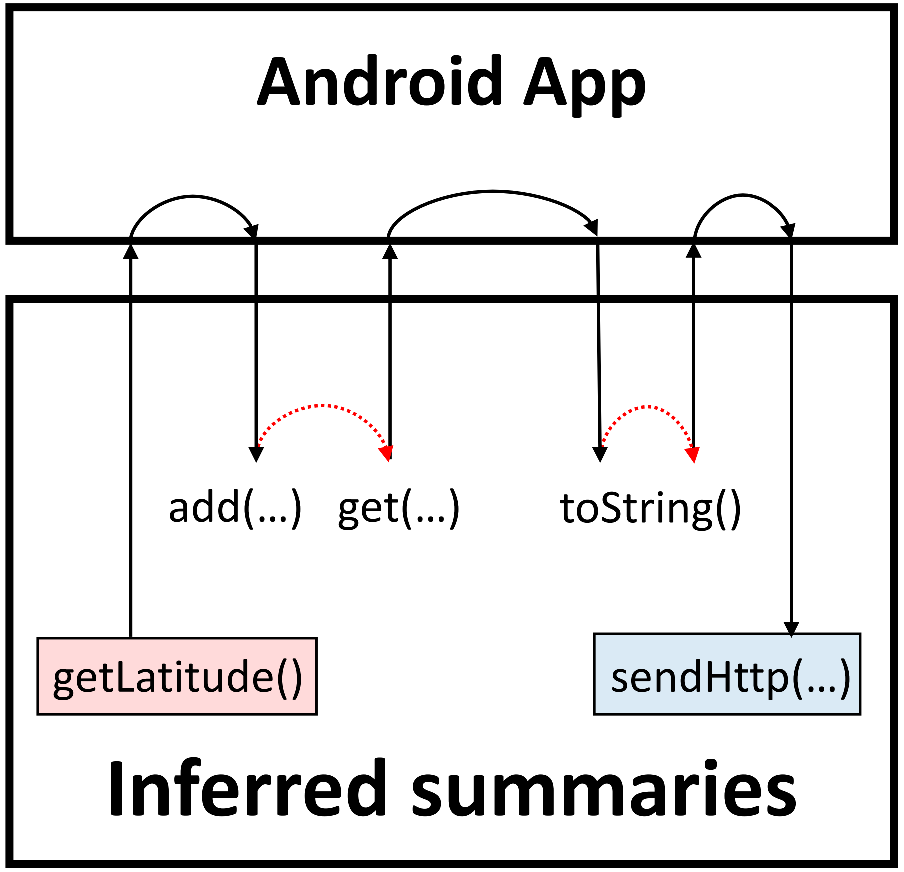
```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



location → Internet
device ID → SMS
…

Step 2: Summary Inference

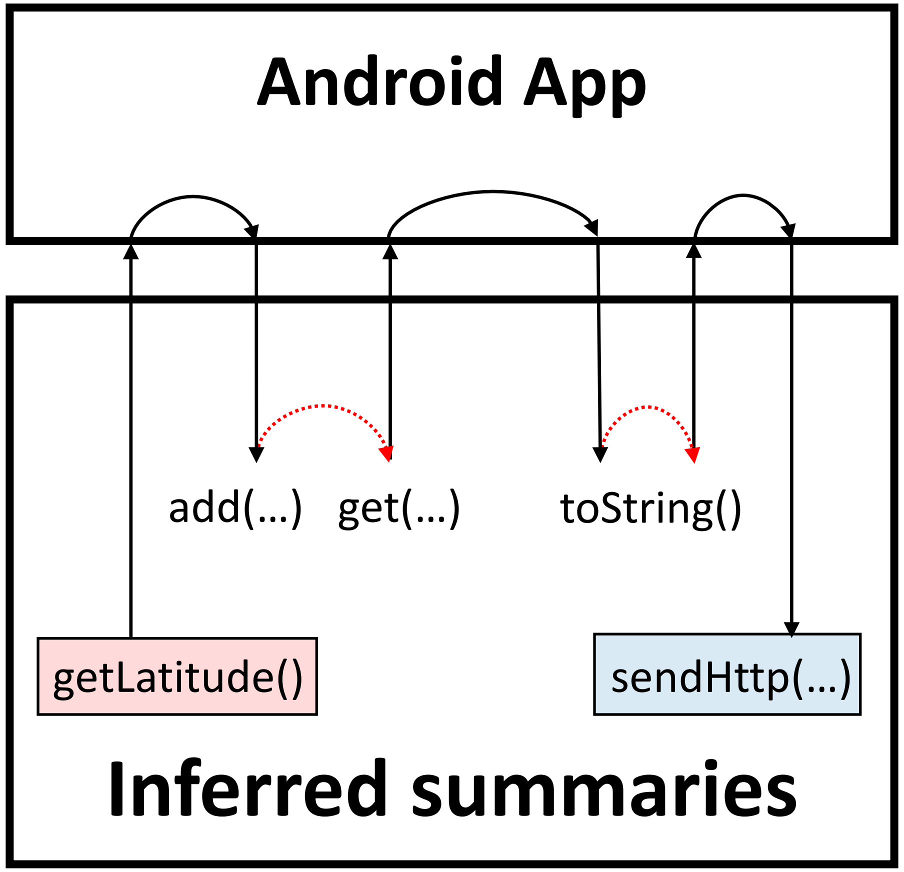
```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



location → Internet
device ID → SMS
…

Step 3: Analyst corrections

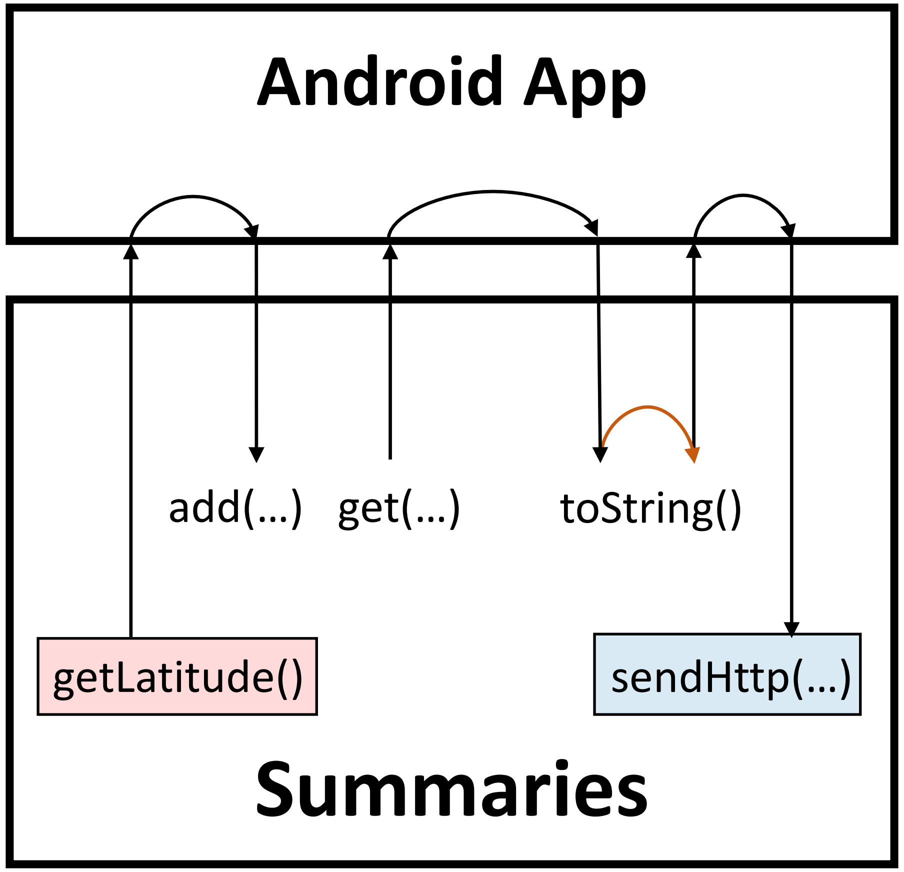
```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



location → Internet
device ID → SMS
…

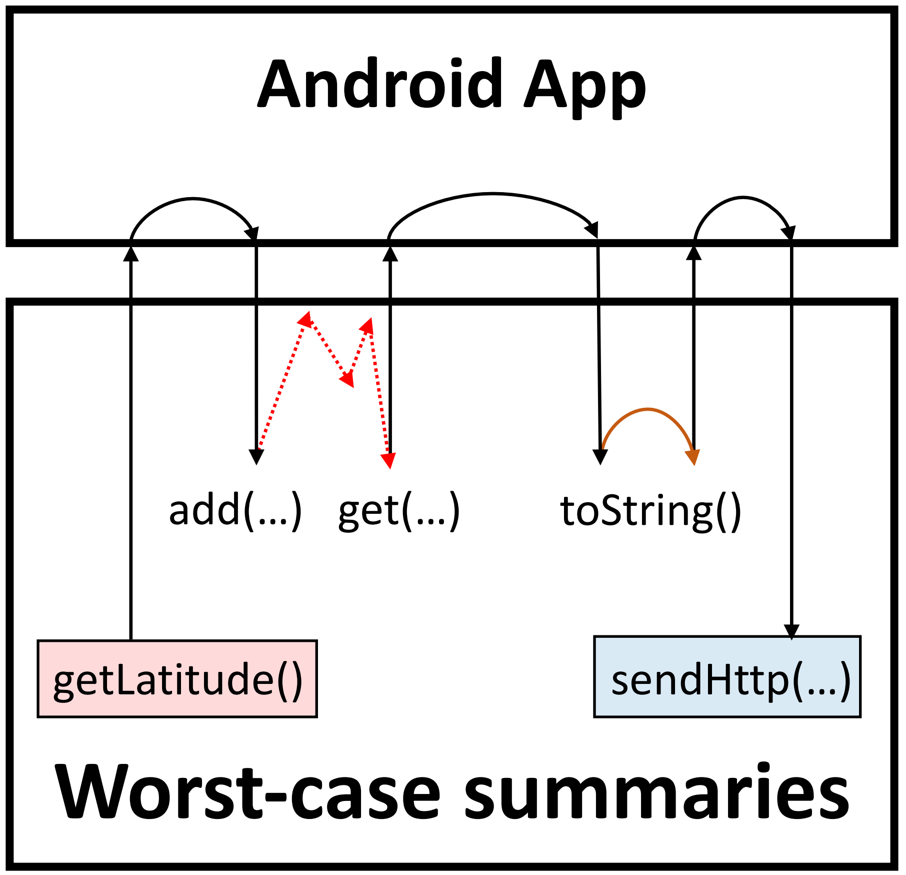
Step 3: Analyst corrections

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



Step 1: Worst-case analysis

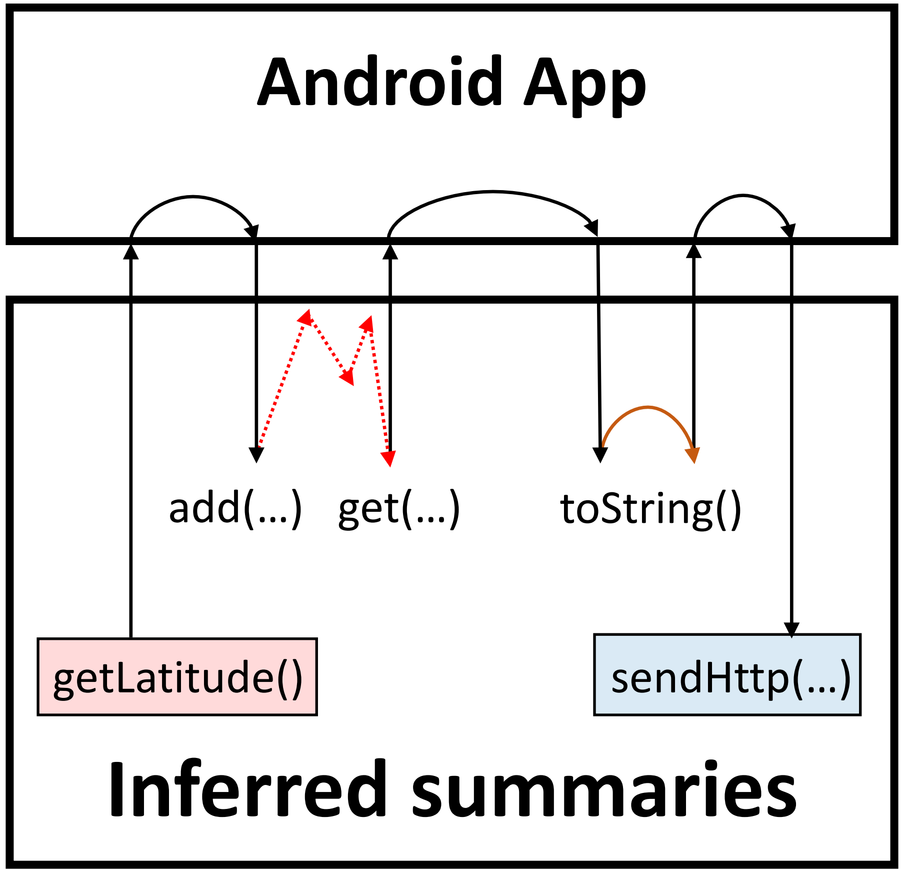
```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



location → Internet

Step 2: Summary inference

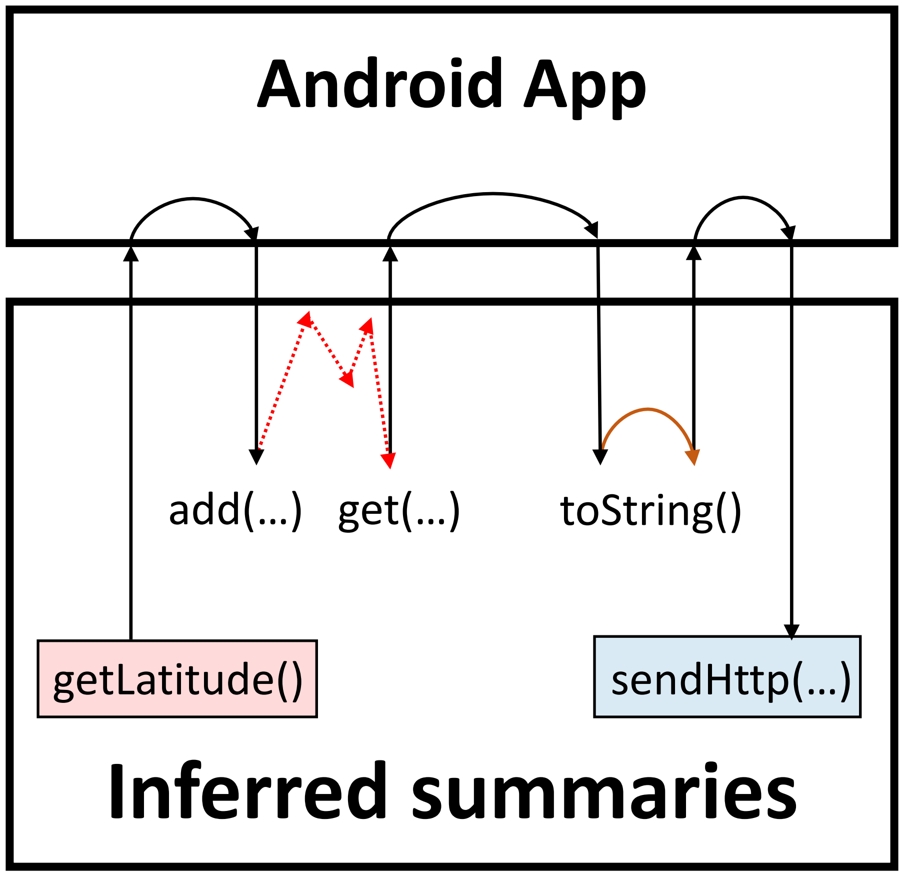
```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



location → Internet

Step 3: Analyst corrections

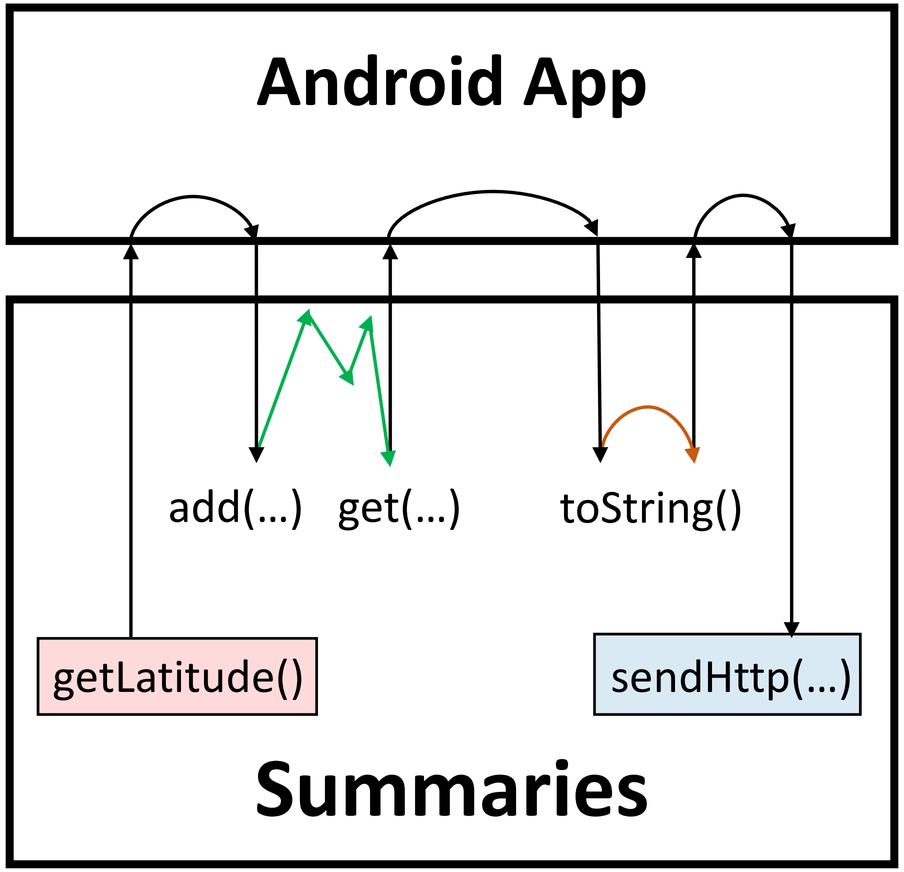
```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



location → Internet

Step 3: Analyst corrections

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



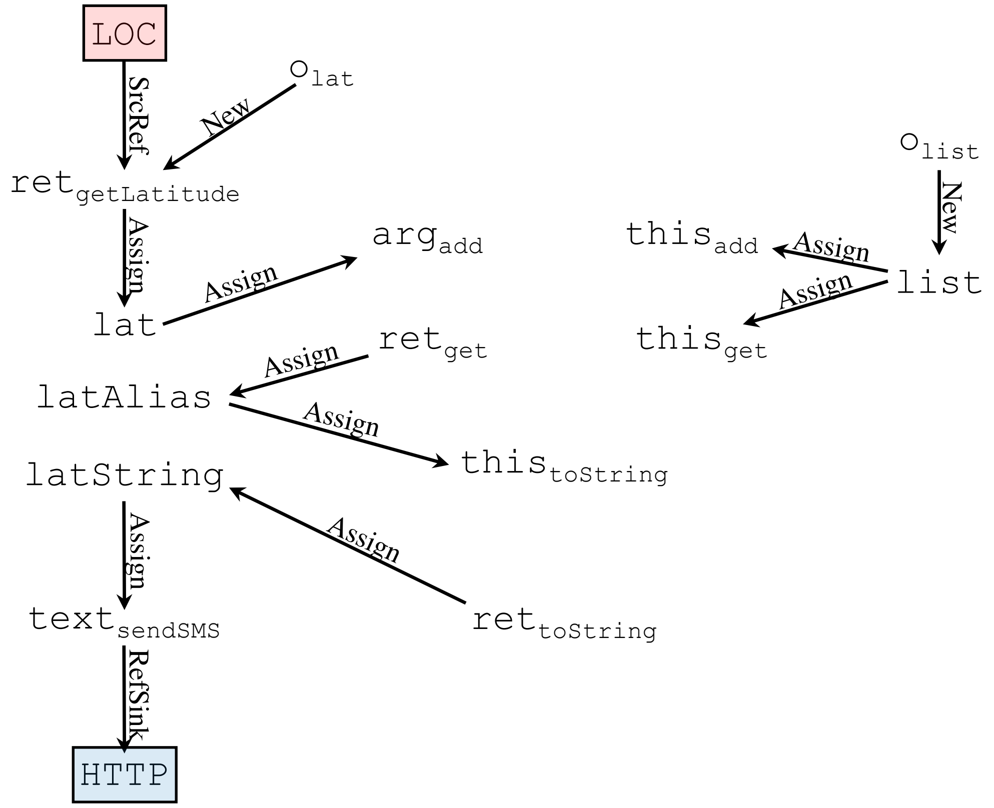
location → Internet

Assume:	The analyst answers correctly
Guarantee:	Results are as if we know all summaries

CFL Reachability

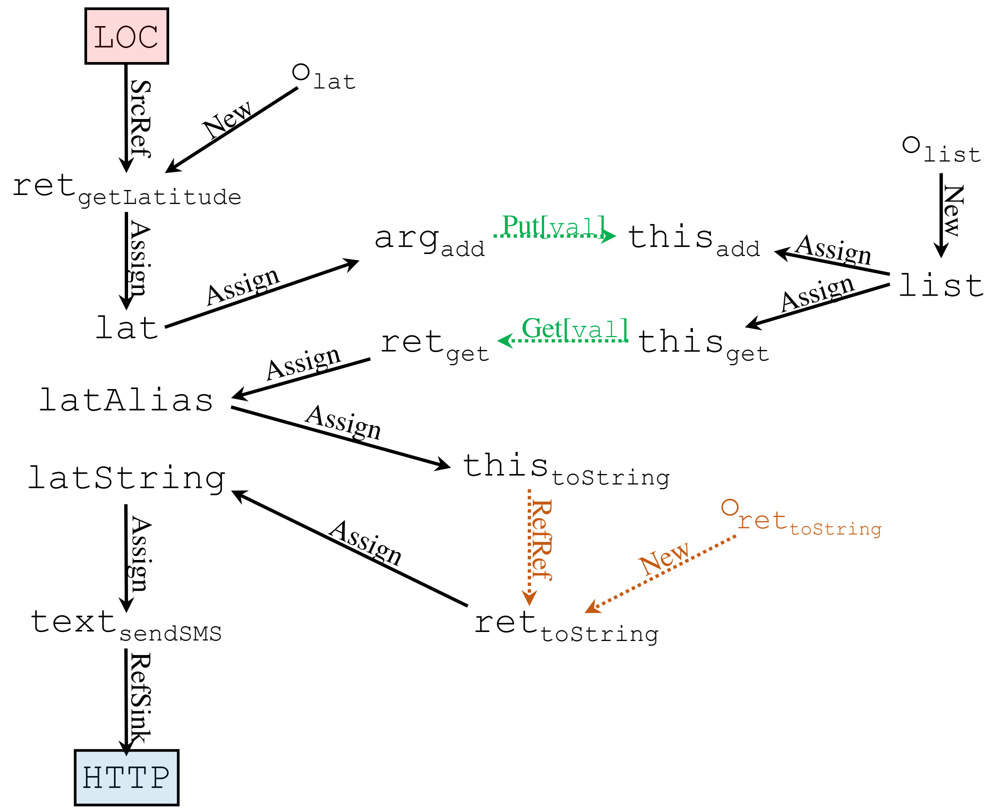
CFL Reachability: Phase 1

1. **Double** lat = getLatitude();
2. **List** list = **new** List();
3. list.add(lat);
4. **Double** latAlias = list.get(0);
5. **String** latStr = latAlias.toString();
6. sendHttp(latStr);

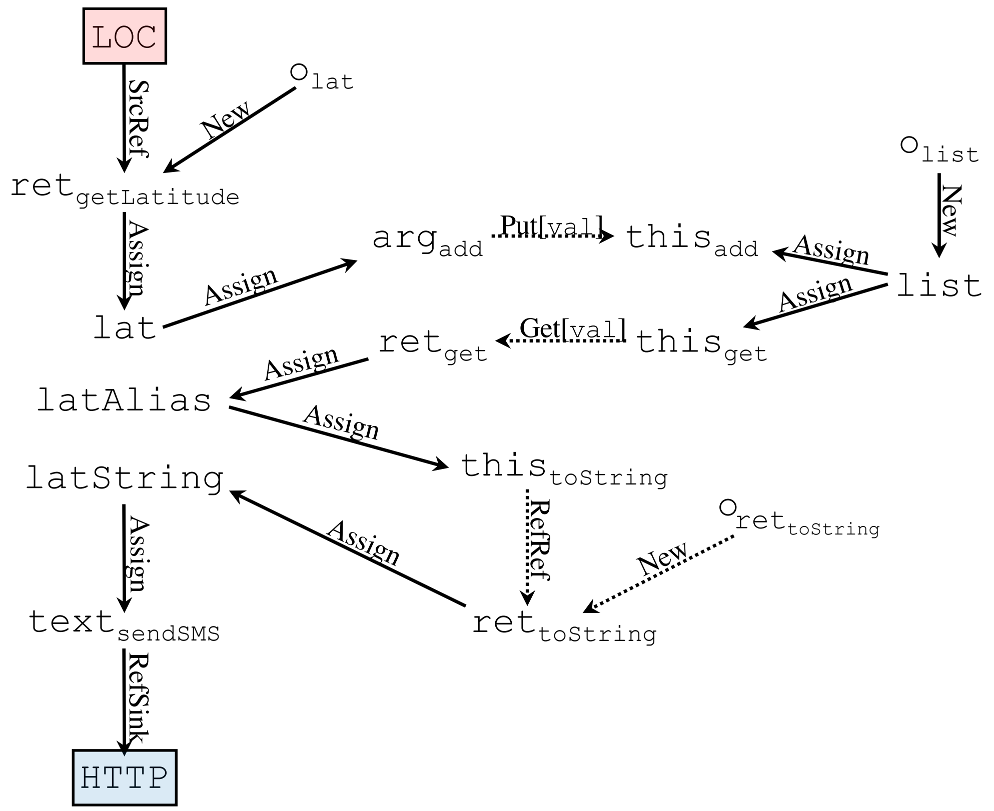


CFL Reachability: Phase 1

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```

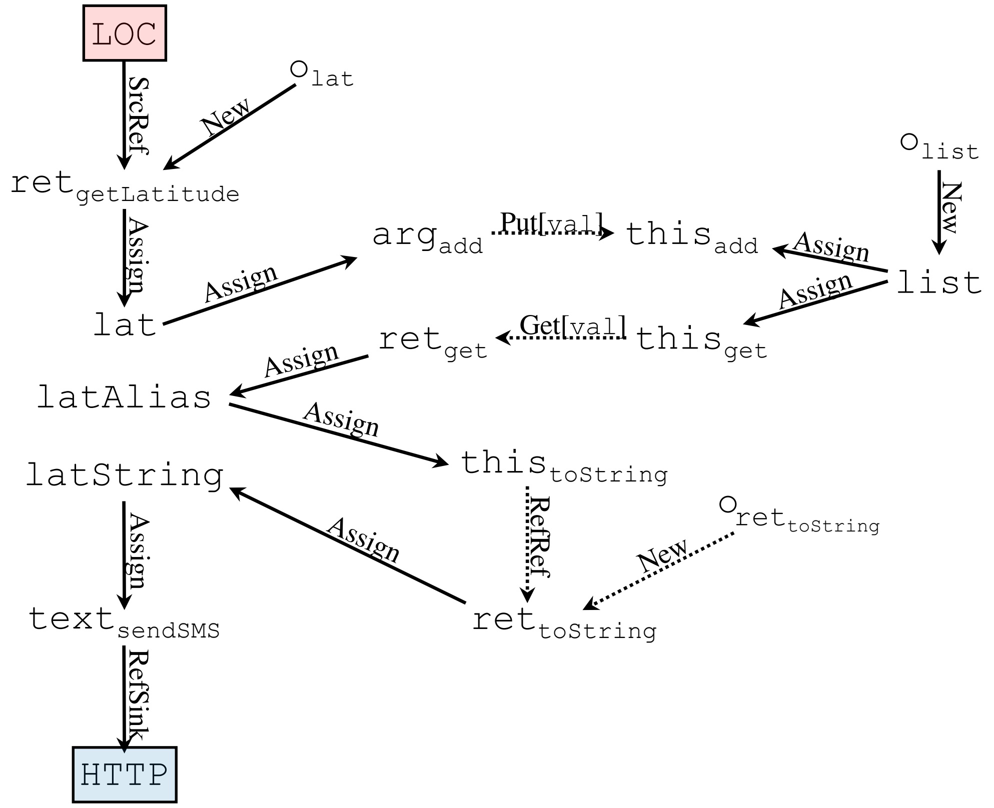


CFL Reachability: Phase 2



CFL Reachability: Phase 2

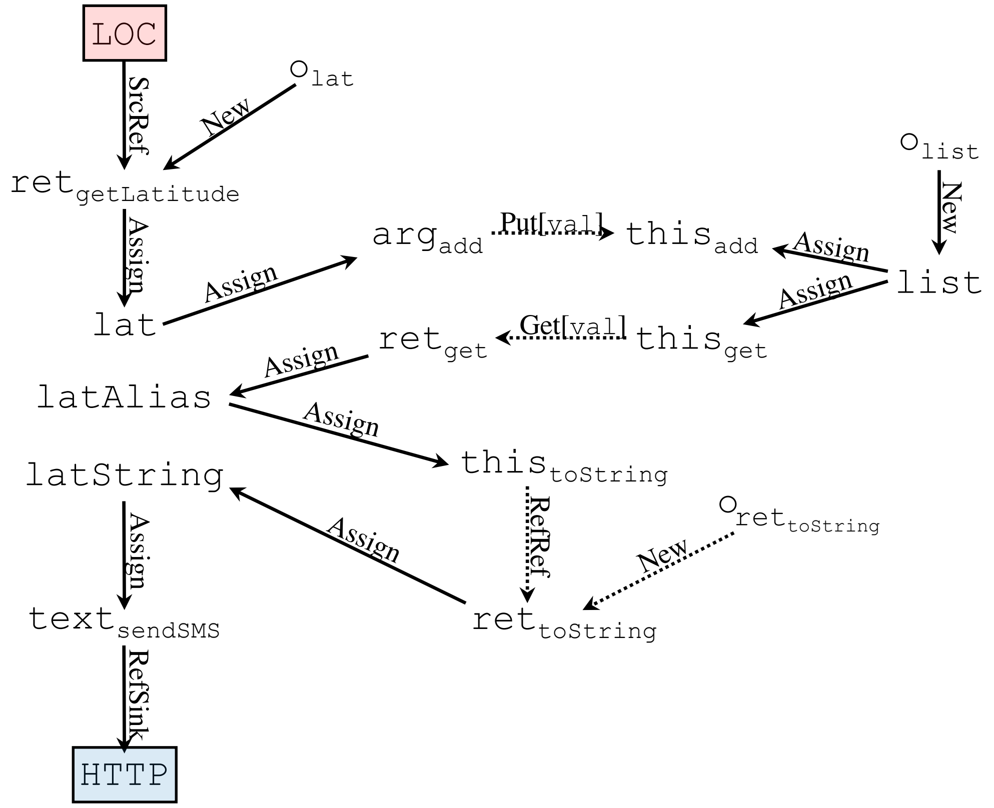
- Source sink paths



CFL Reachability: Phase 2

- Source sink paths
- Labels along path satisfy:

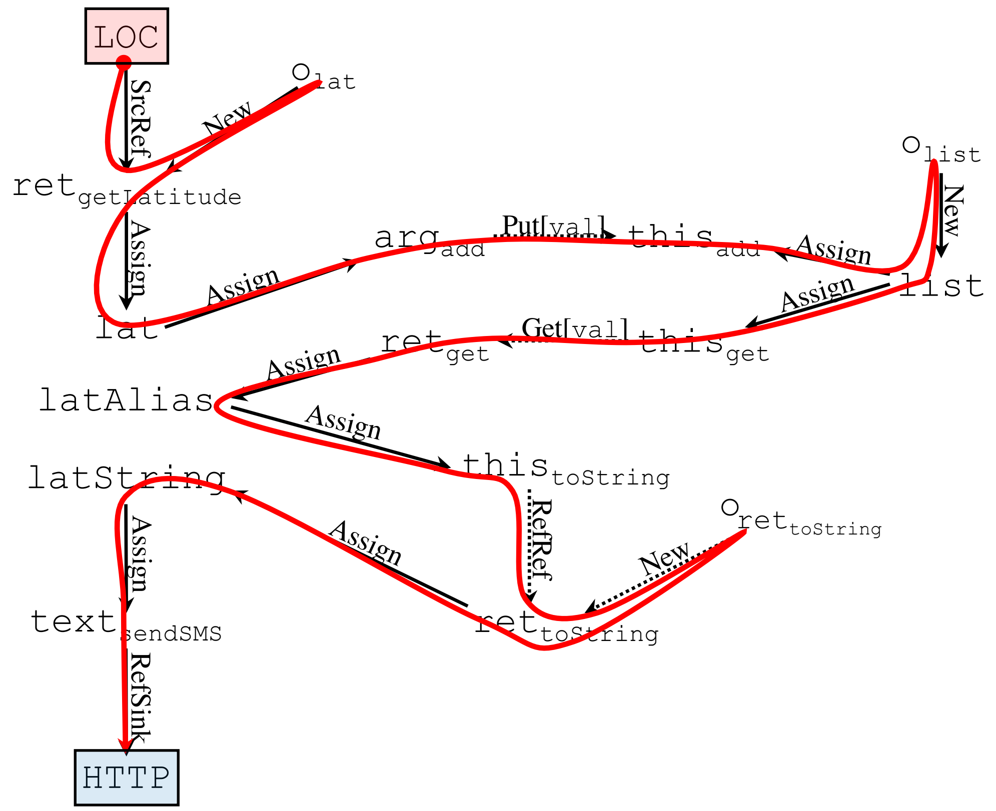
$$\ell_1 \dots \ell_n \in L_{\text{flow}}$$



CFL Reachability: Phase 2

- Source sink paths
- Labels along path satisfy:

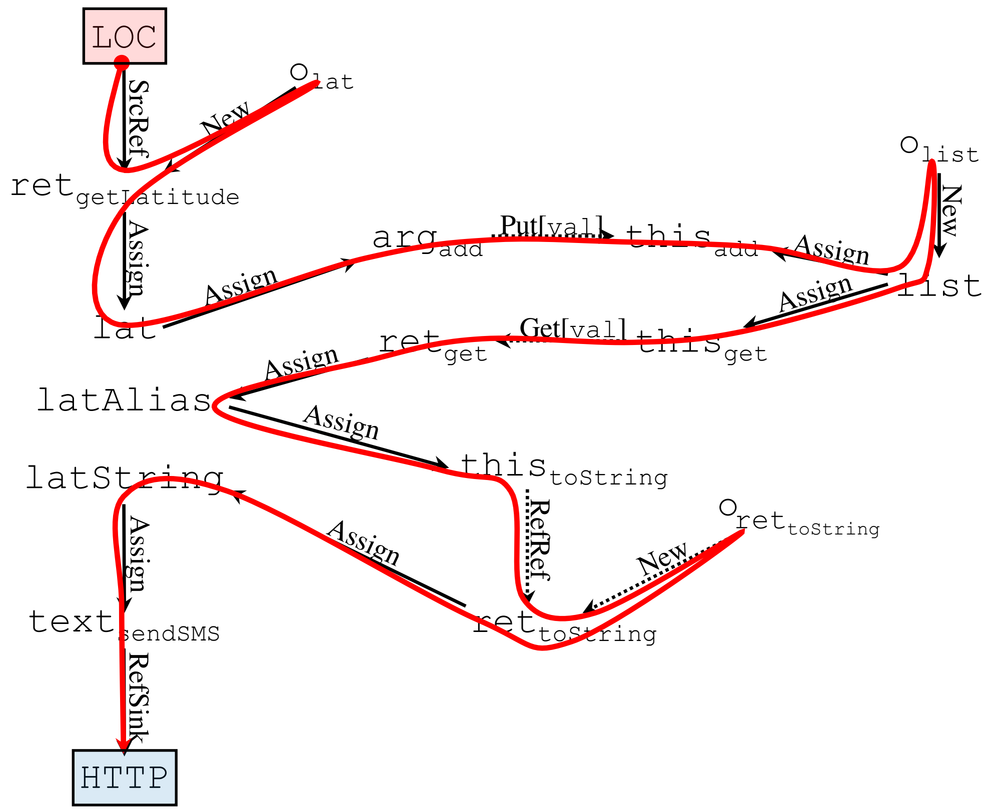
$$\ell_1 \dots \ell_n \in L_{\text{flow}}$$



CFL Reachability: Phase 2

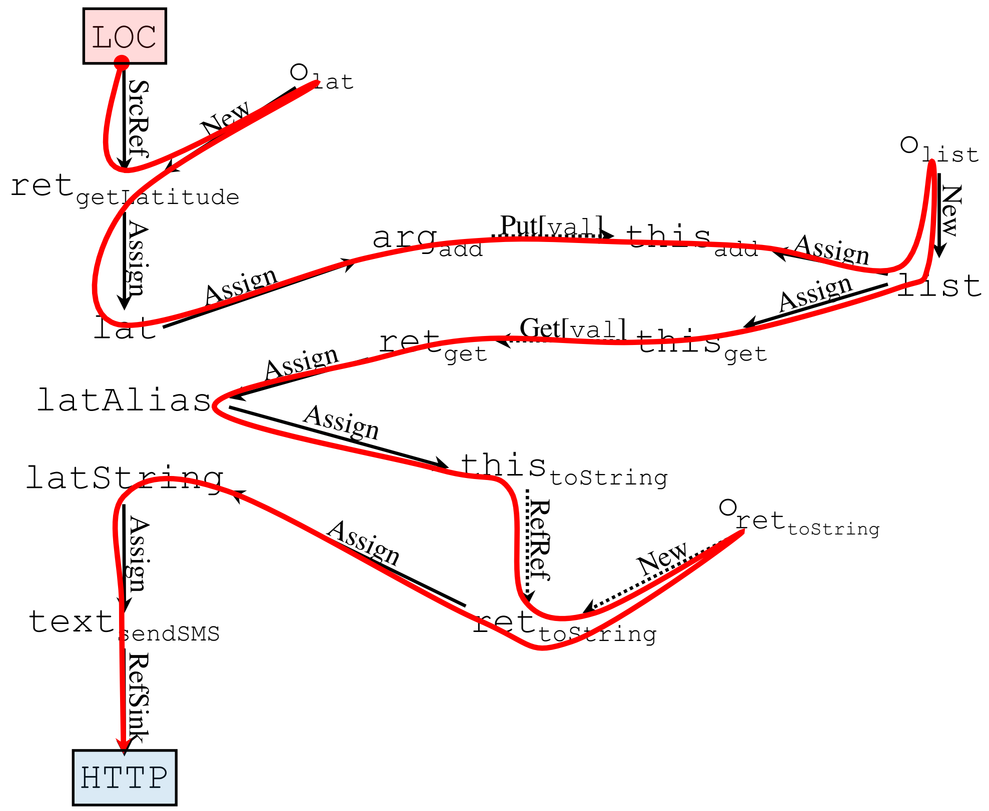
- Source sink paths
- Labels along path satisfy:

$$\ell_1 \ldots \ell_n \in L_{\text{flow}}$$



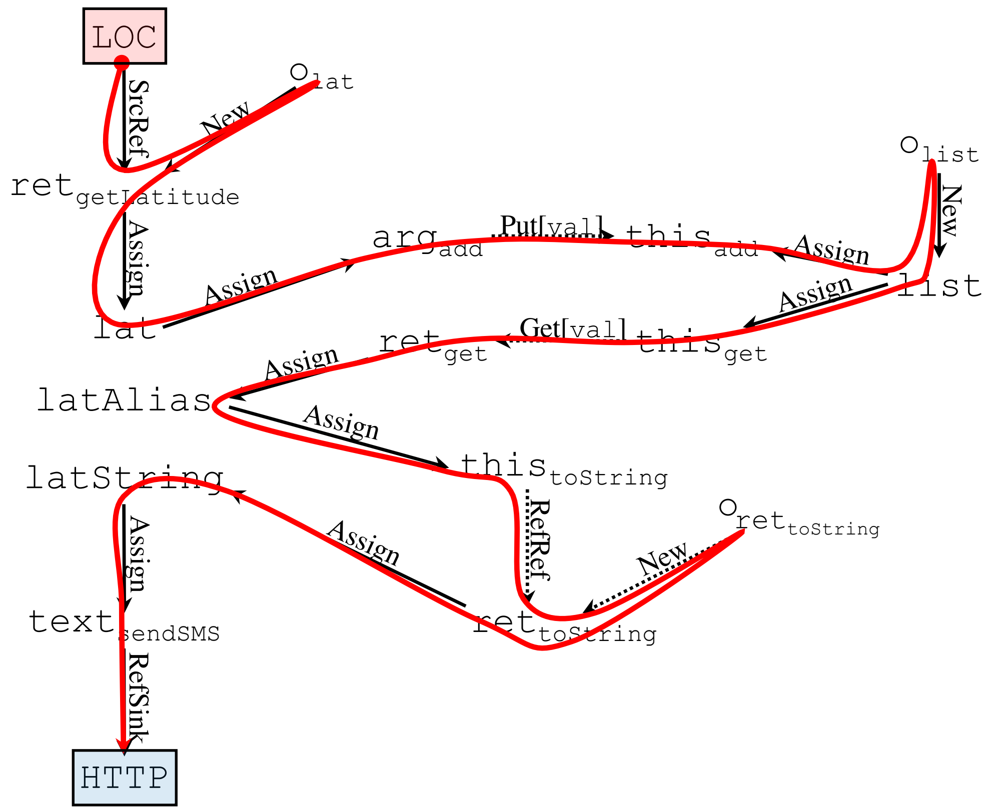
$$\text{SrcRef}\ \overline{\text{New}}\ \text{New Assign Assign Put[val] Assign}\ \overline{\text{New}}\ \text{New Assign Get[Val] Assign Assign RefRef}\ \overline{\text{New}}\ \text{New Assign Assign RefSink} \in L_{\text{flow}}$$

Missing Summaries



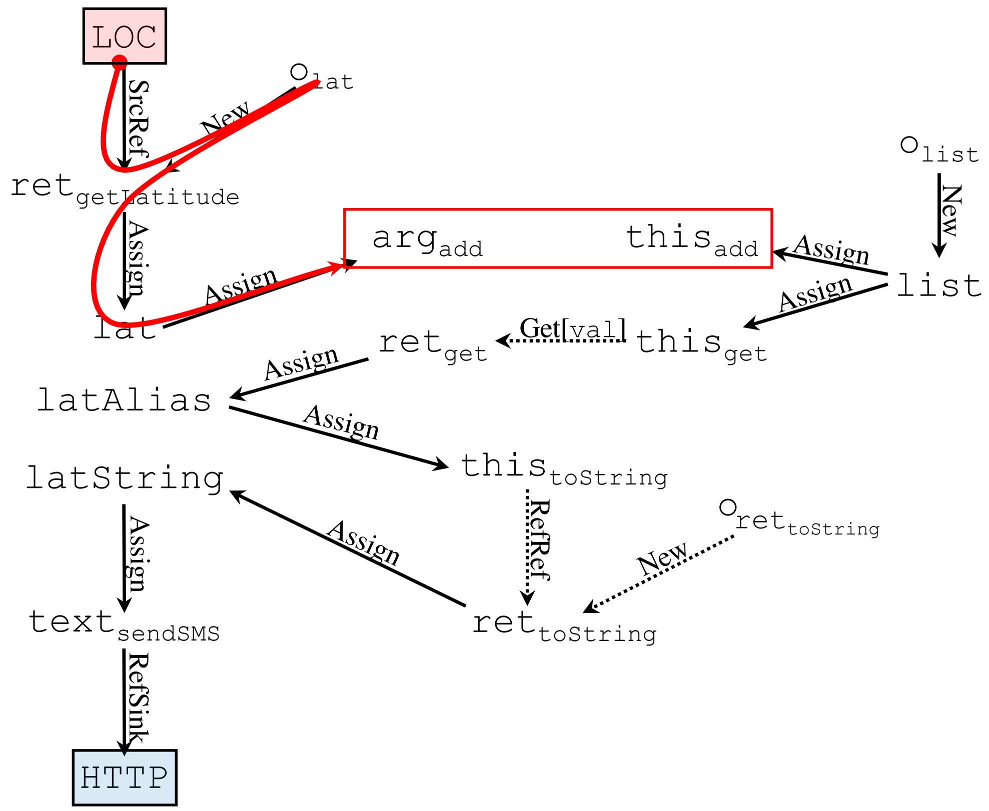
Missing Summaries

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



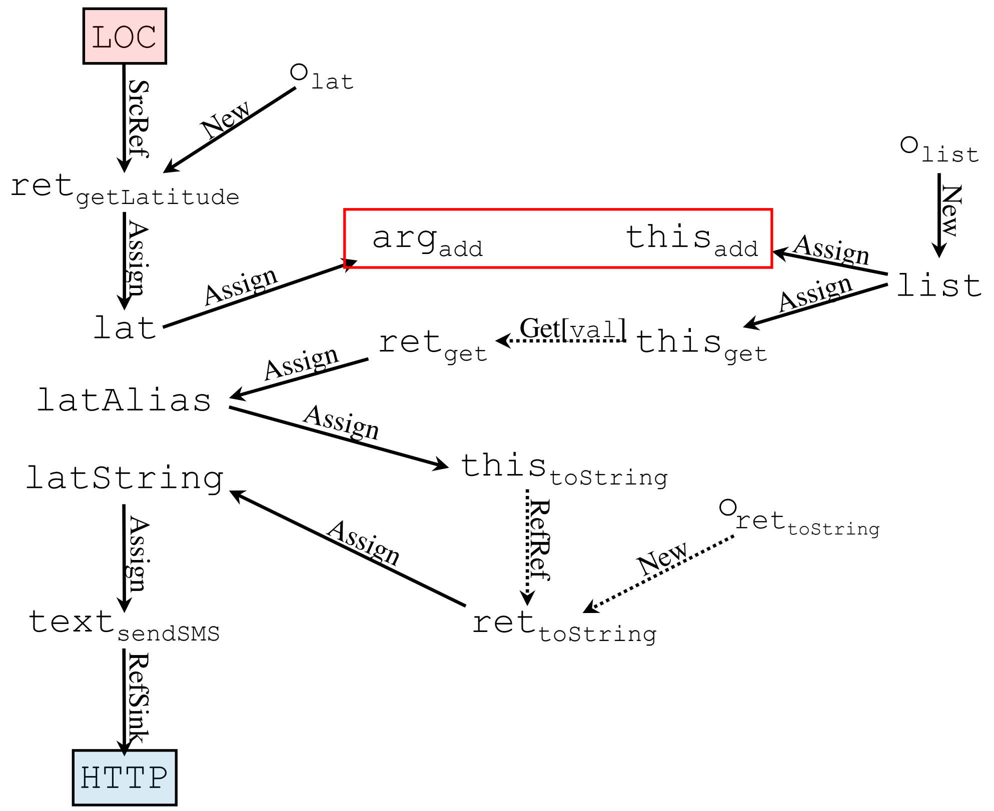
Missing Summaries

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



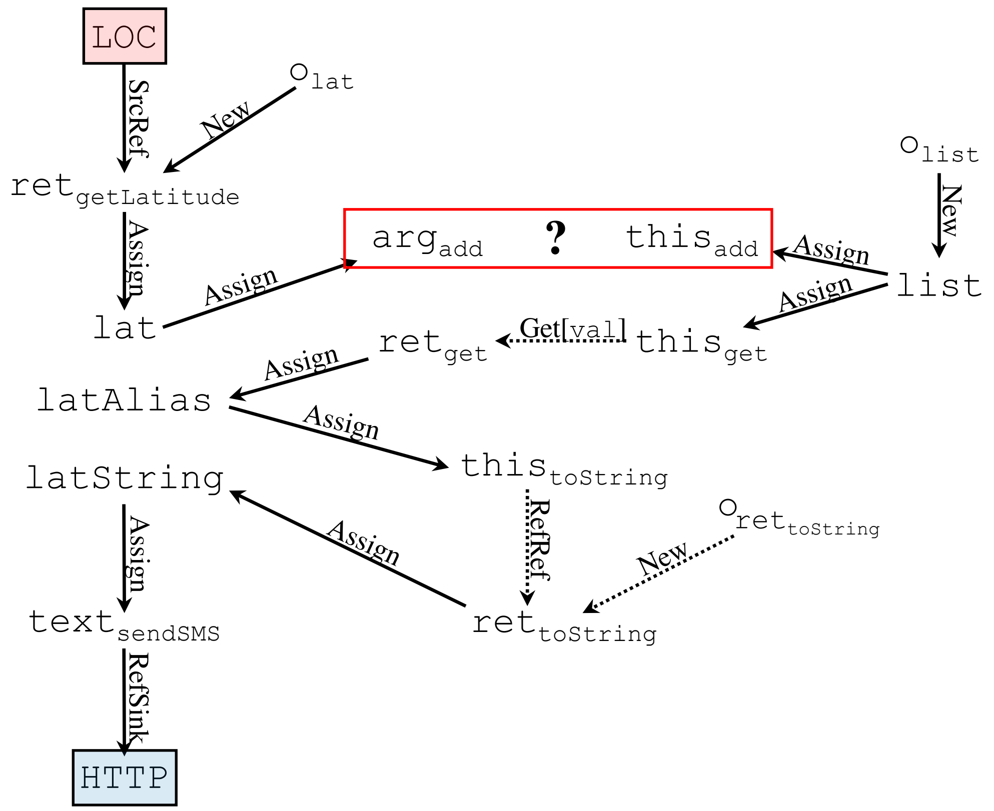
Step 1: Worst-case Analysis

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



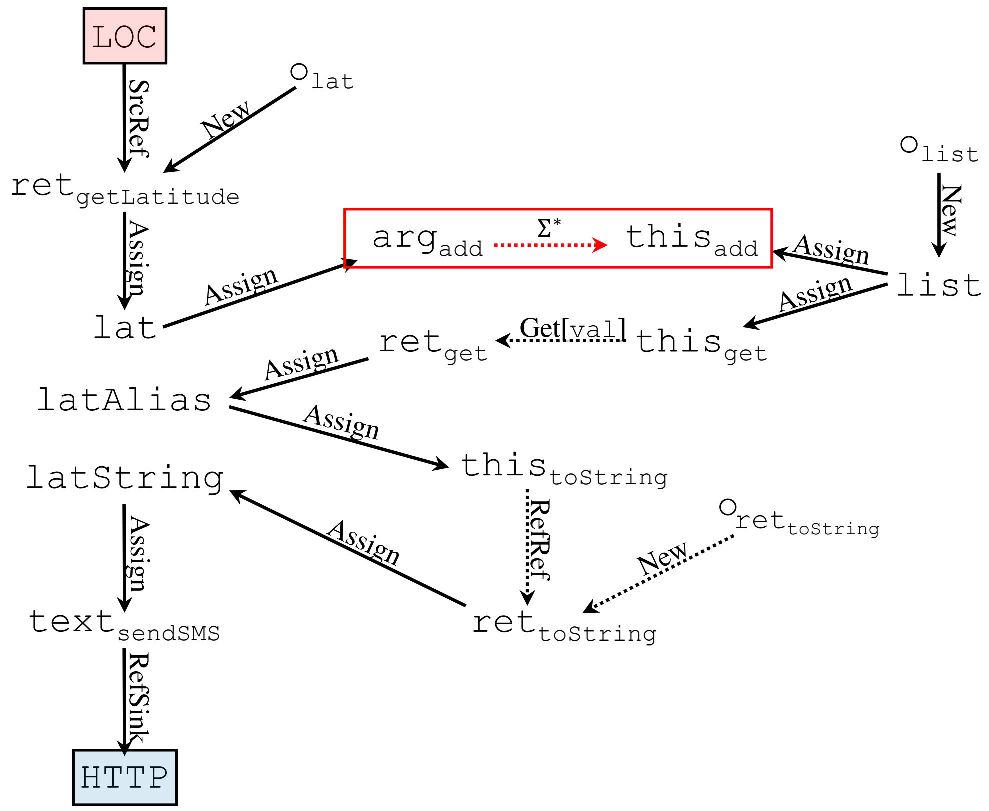
Step 1: Worst-case Analysis

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



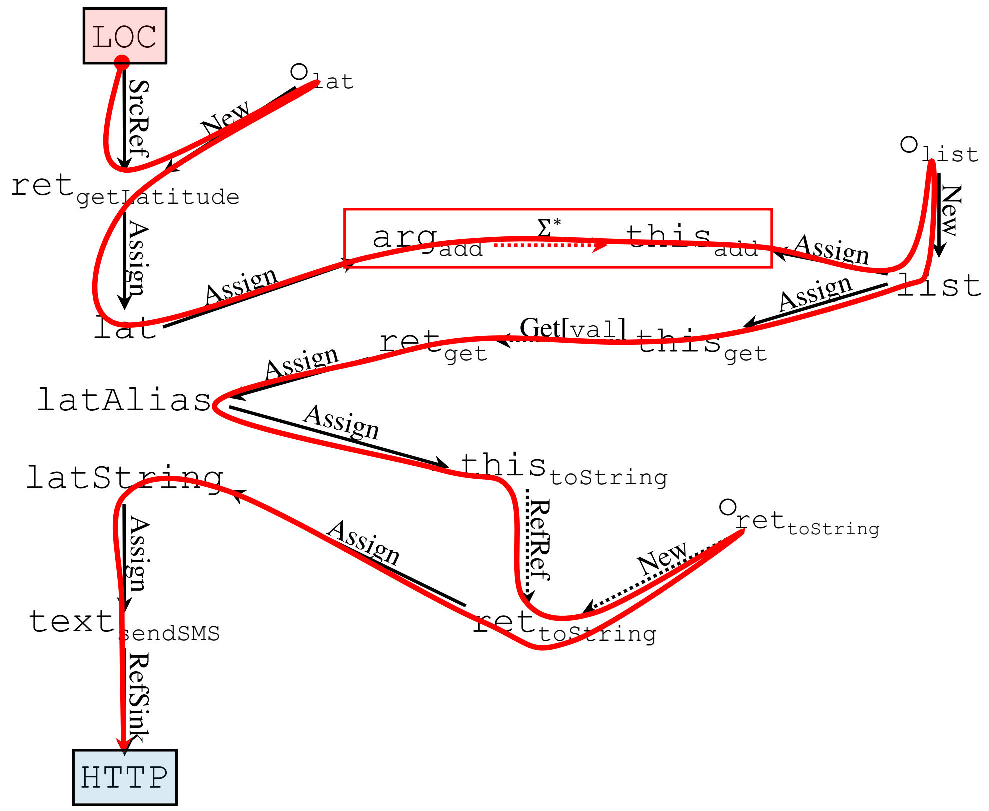
Step 1: Worst-case Analysis

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



Step 1: Worst-case Analysis

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



SrcRef $\overline{\text{New}}$ New Assign Assign (Put[val] $\in \Sigma^*$) Assign $\overline{\text{New}}$ New Assign Get[Val] Assign Assign RefRef $\overline{\text{New}}$ New Assign Assign RefSink $\in L(C_{\text{flow}})$

Key idea: Worst-case Subgraph

Key idea: Worst-case Subgraph

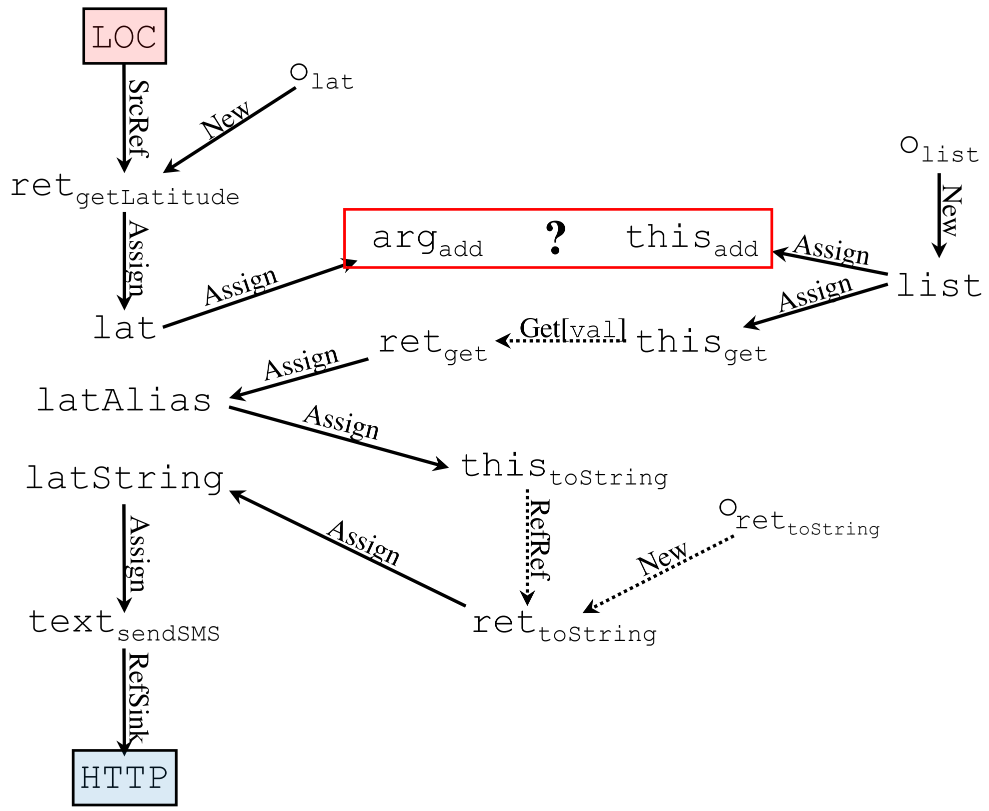
$$\xrightarrow{\Sigma^*}$$

Key idea: Worst-case Subgraph

$$\begin{tikzcd} {} \arrow[r, "\Sigma^*"] & {} \arrow[r, phantom, "="] & {} \arrow[r, "\epsilon"] & t \arrow[r, "\epsilon"] \arrow[loop, out=240, in=260, "\sigma_1"'] \arrow[loop, out=280, in=300, "\sigma_n"] & {} \\ & & & \cdots & \end{tikzcd}$$

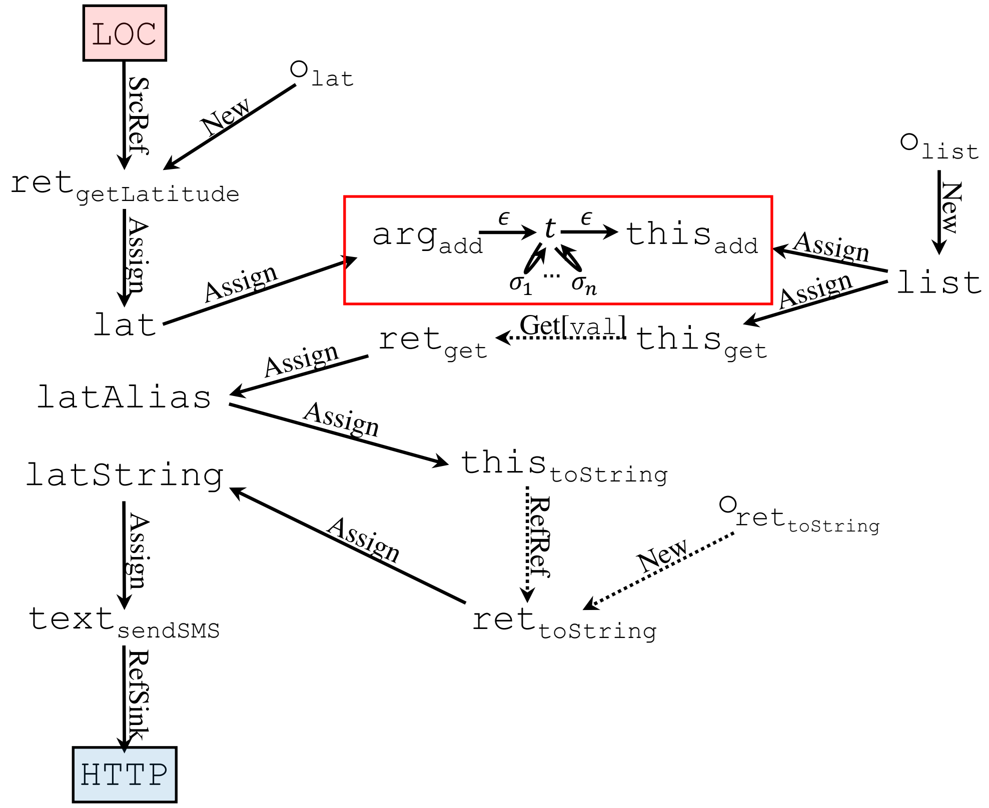
Step 1: Worst-case Analysis

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



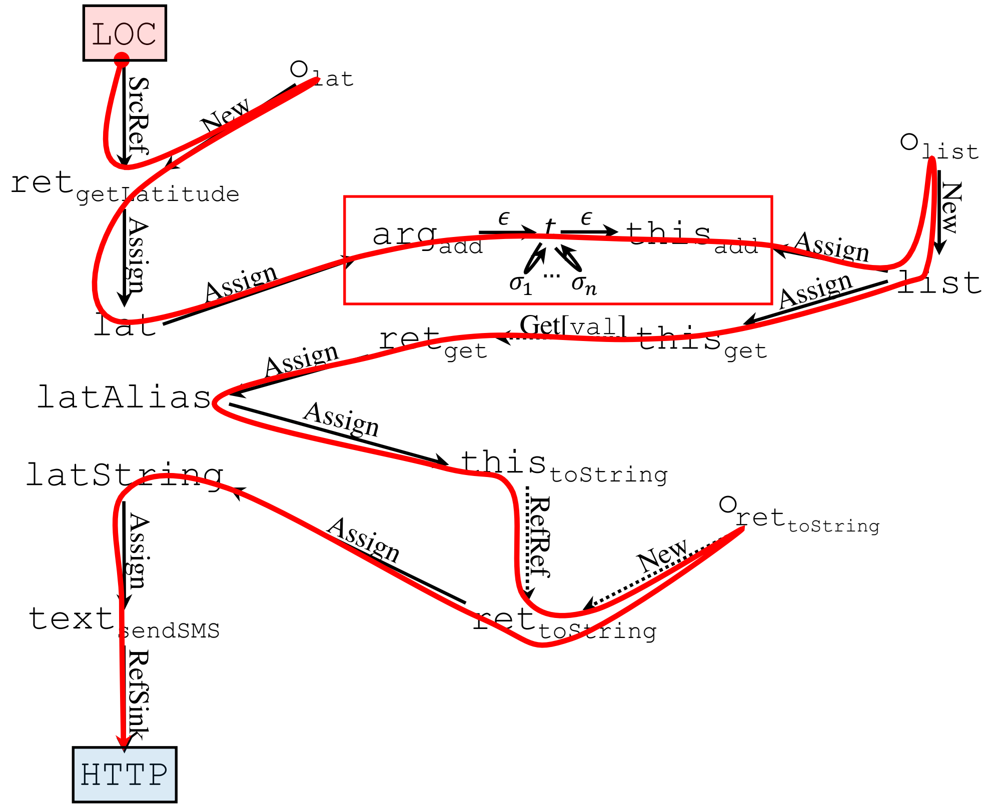
Step 1: Worst-case Analysis

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



Step 1: Worst-case Analysis

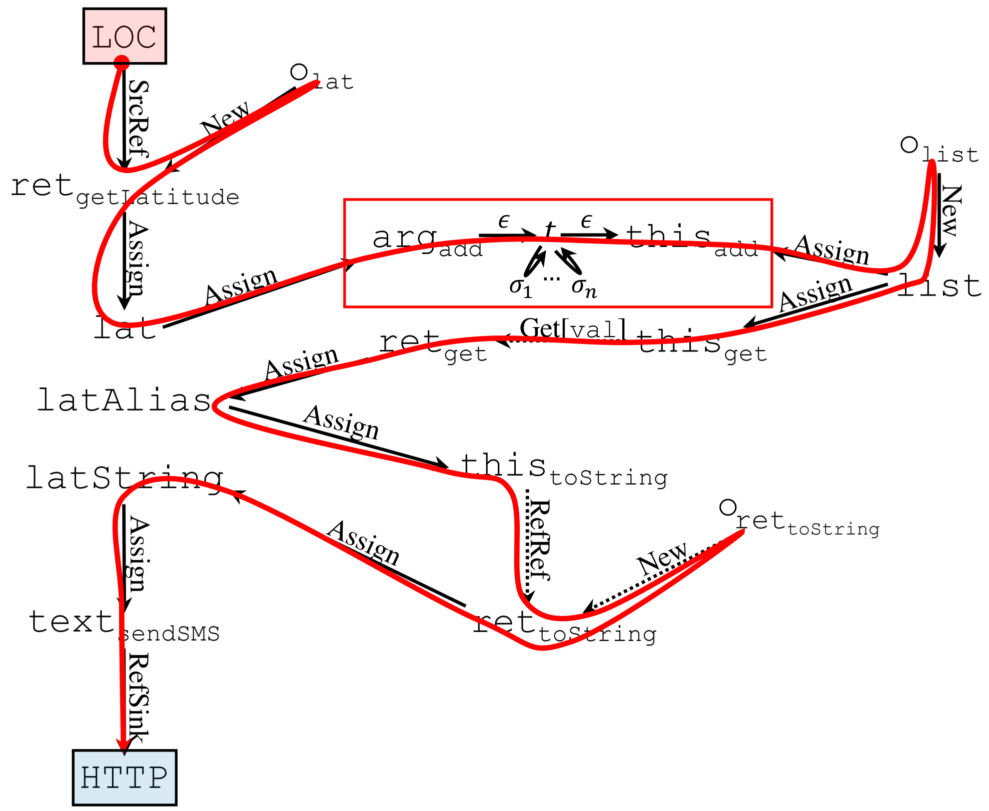
```
1.  class List:
2.      @alias(arg, this.val)
3.      void add(Object arg) {}
4.
5.      @alias(this.val, return)
6.      Object get(Integer index) {}
7.
8.  class Double:
9.      @flow(this, return)
10.     String toString() {}
```



SrcRef $\overline{\text{New}}$ New Assign Assign ϵ Put[val] ϵ Assign $\overline{\text{New}}$ New Assign Get[Val] Assign Assign RefRef $\overline{\text{New}}$ New Assign Assign RefSink $\in L(C_{\text{flow}})$

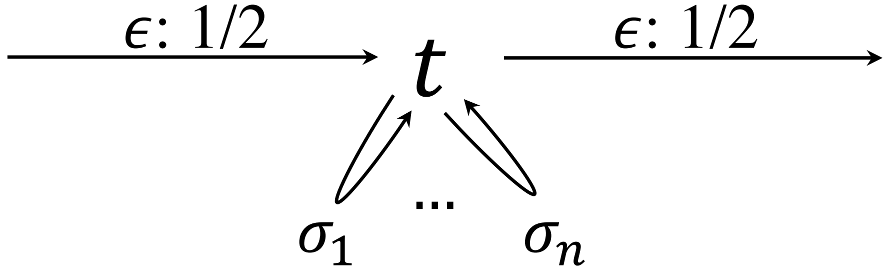
Step 2: Summary Inference

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



Key Idea: Shortest Path

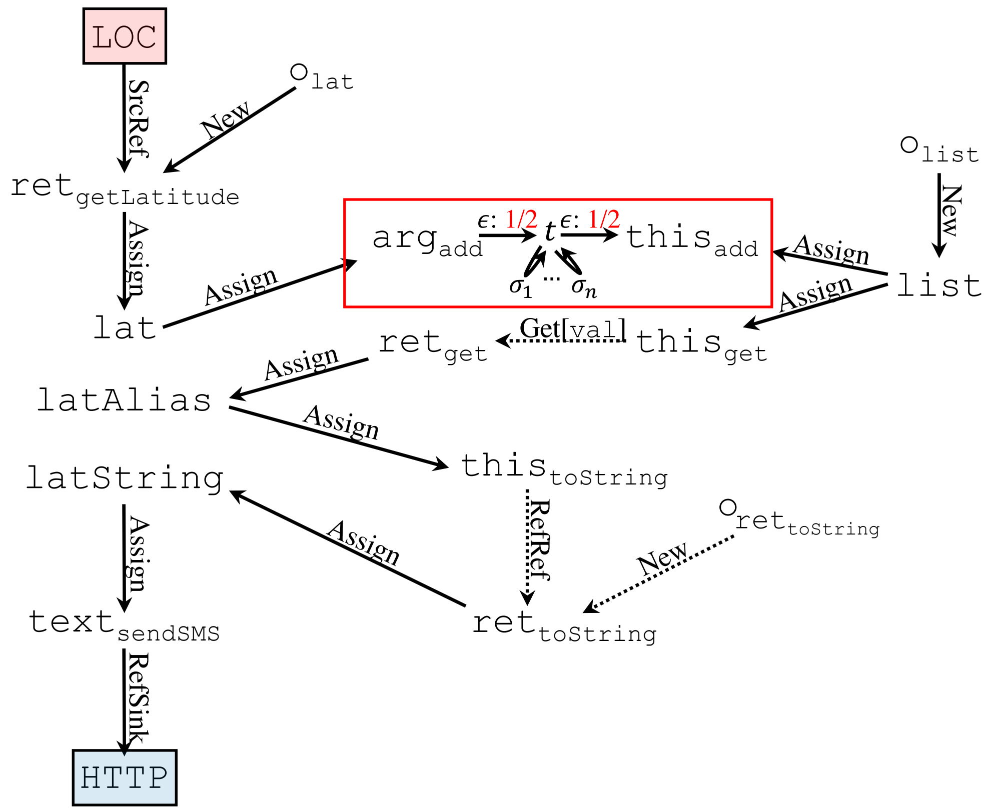
Key Idea: Shortest Path



(Other edges have weight 0)

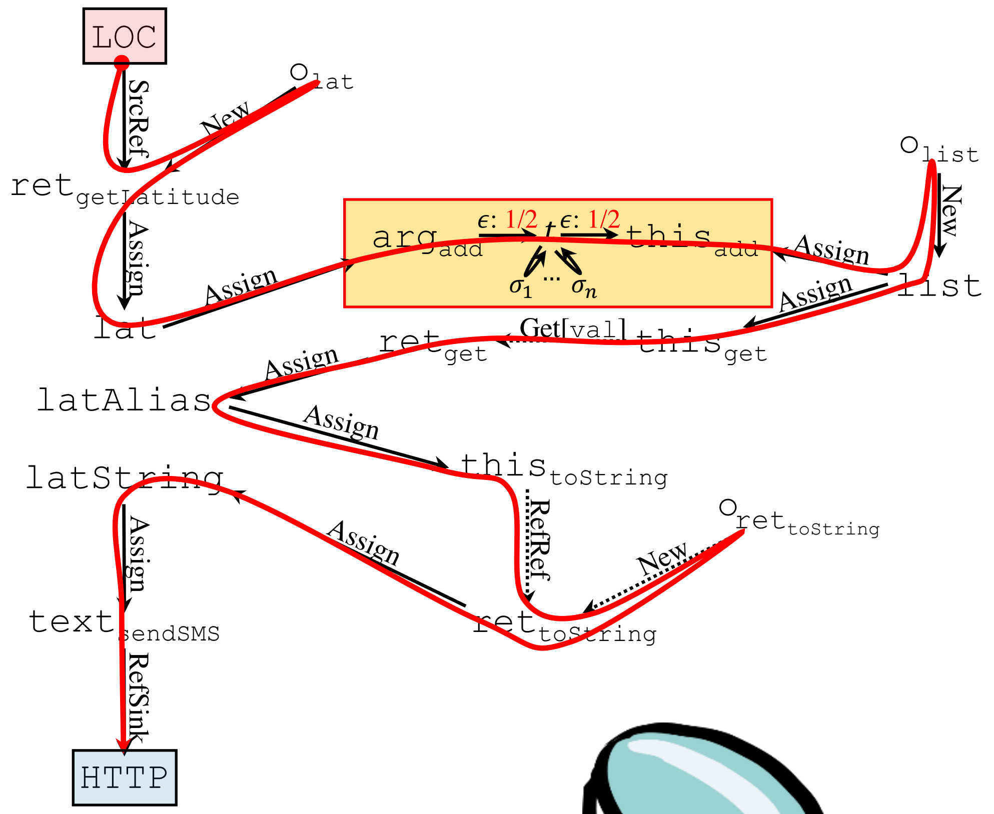
Step 2: Summary Inference

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



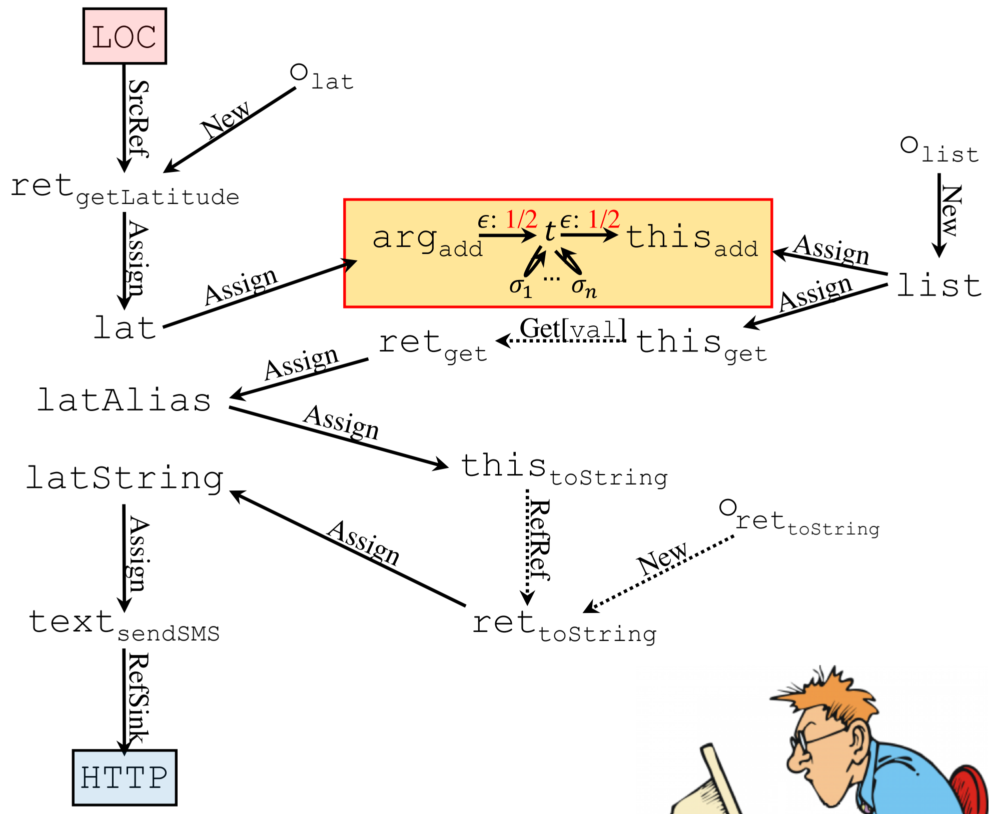
Step 2: Summary Inference

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



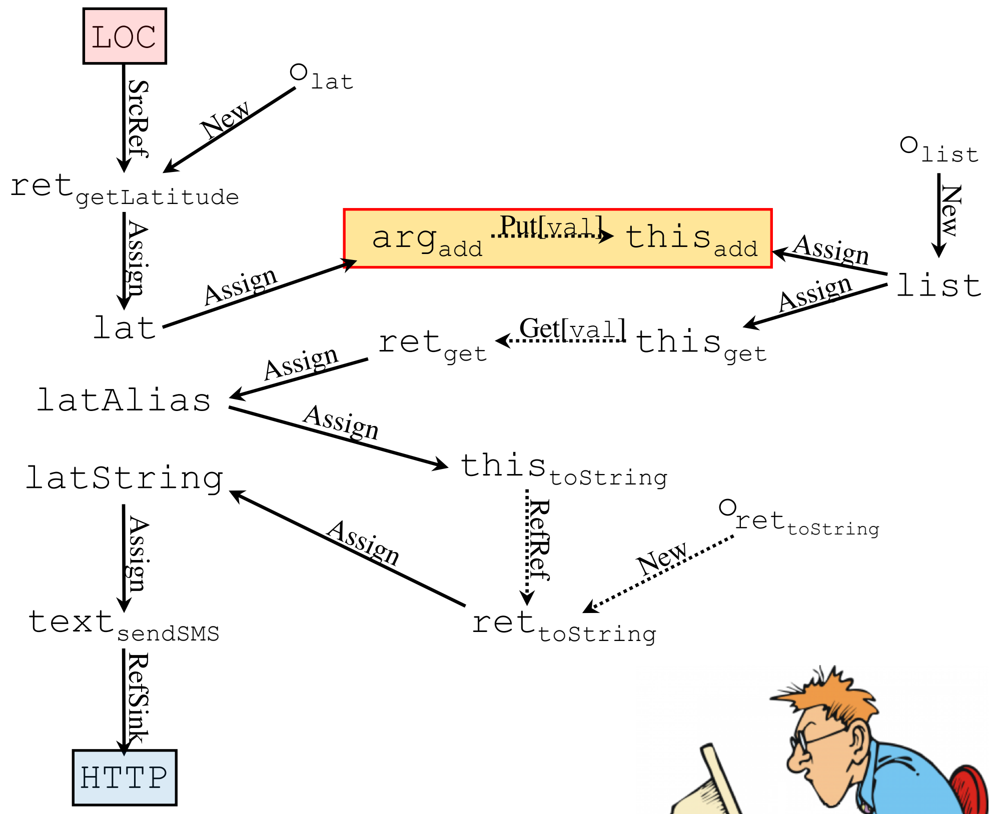
Step 3: Analyst Corrections

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```

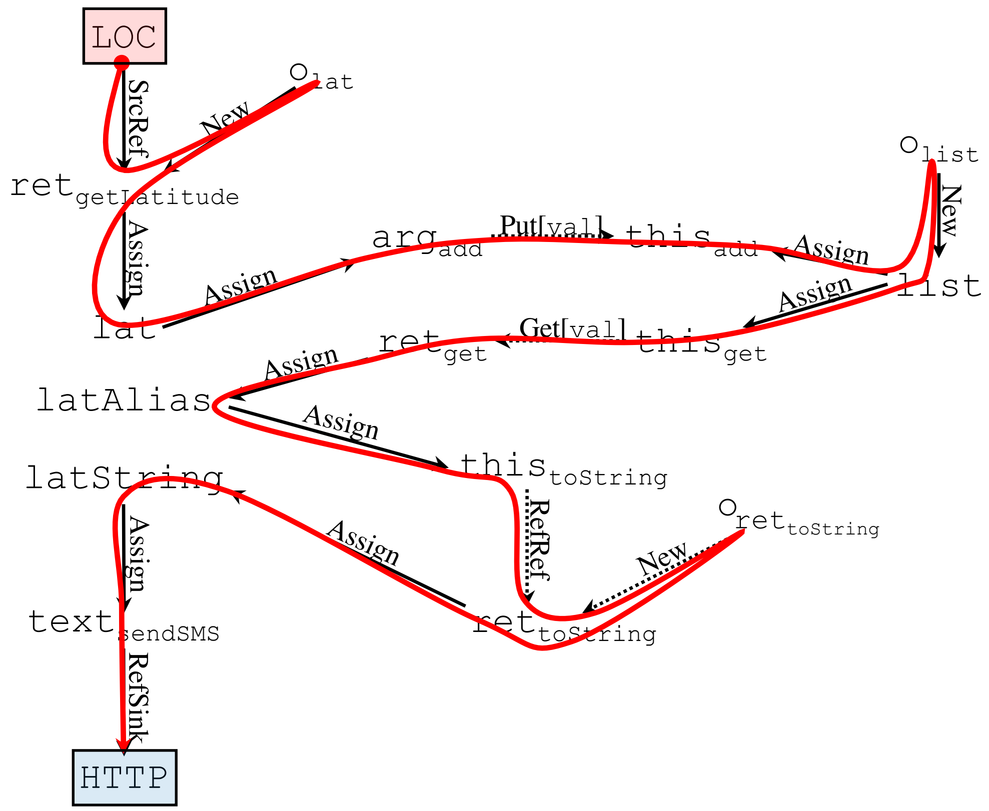


Step 3: Analyst Corrections

```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



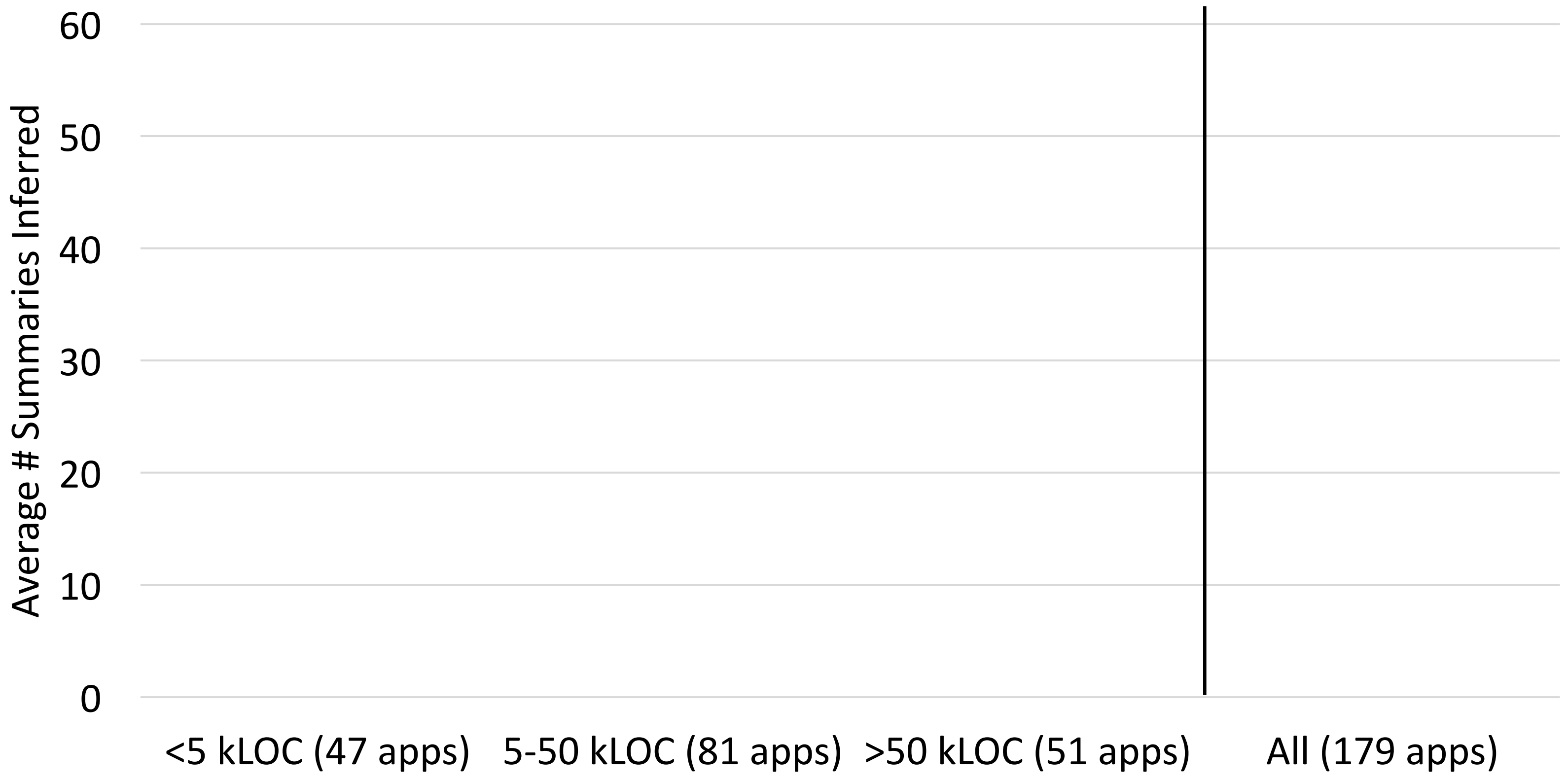
```
class List:
    @alias(arg, this.val)
    void add(Object arg) {}

    @alias(this.val, return)
    Object get(Integer index) {}

class Double:
    @flow(this, return)
    String toString() {}
```



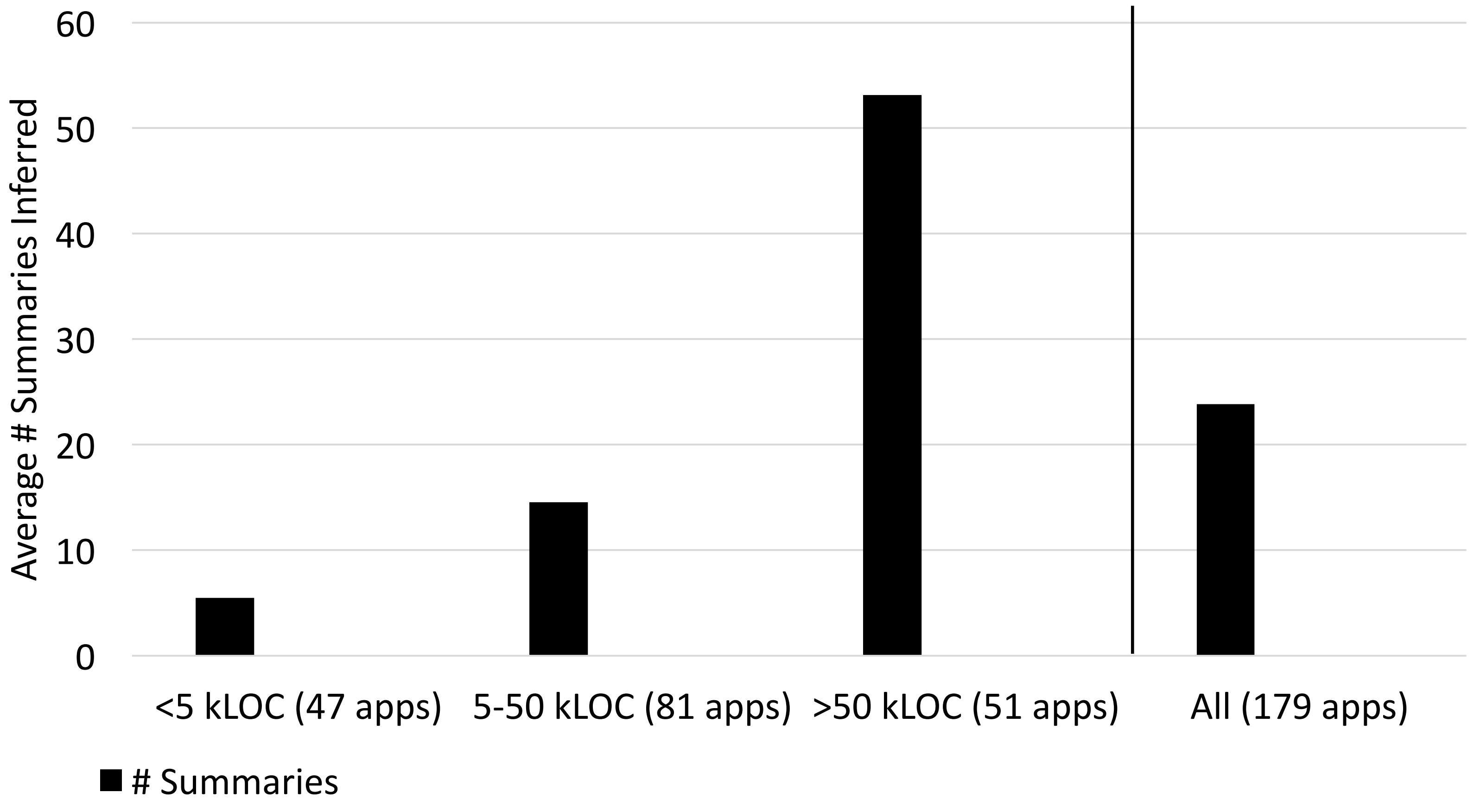
Evaluation

- **Total:** 179 apps (Symantec/Google Play/Darpa)
- **Flow:** 179 apps
- **Alias:** 156 apps

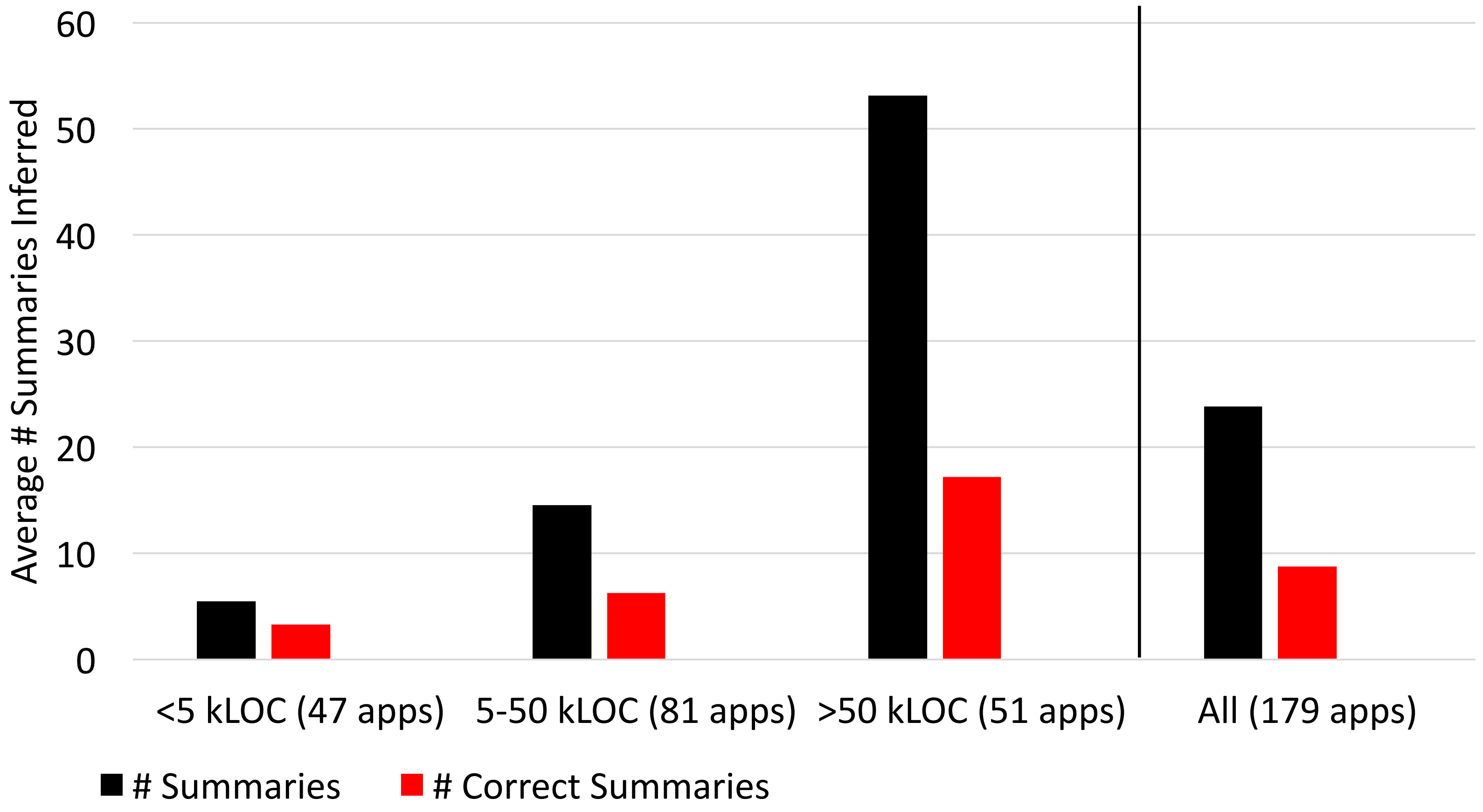
Evaluation: Flow Summaries



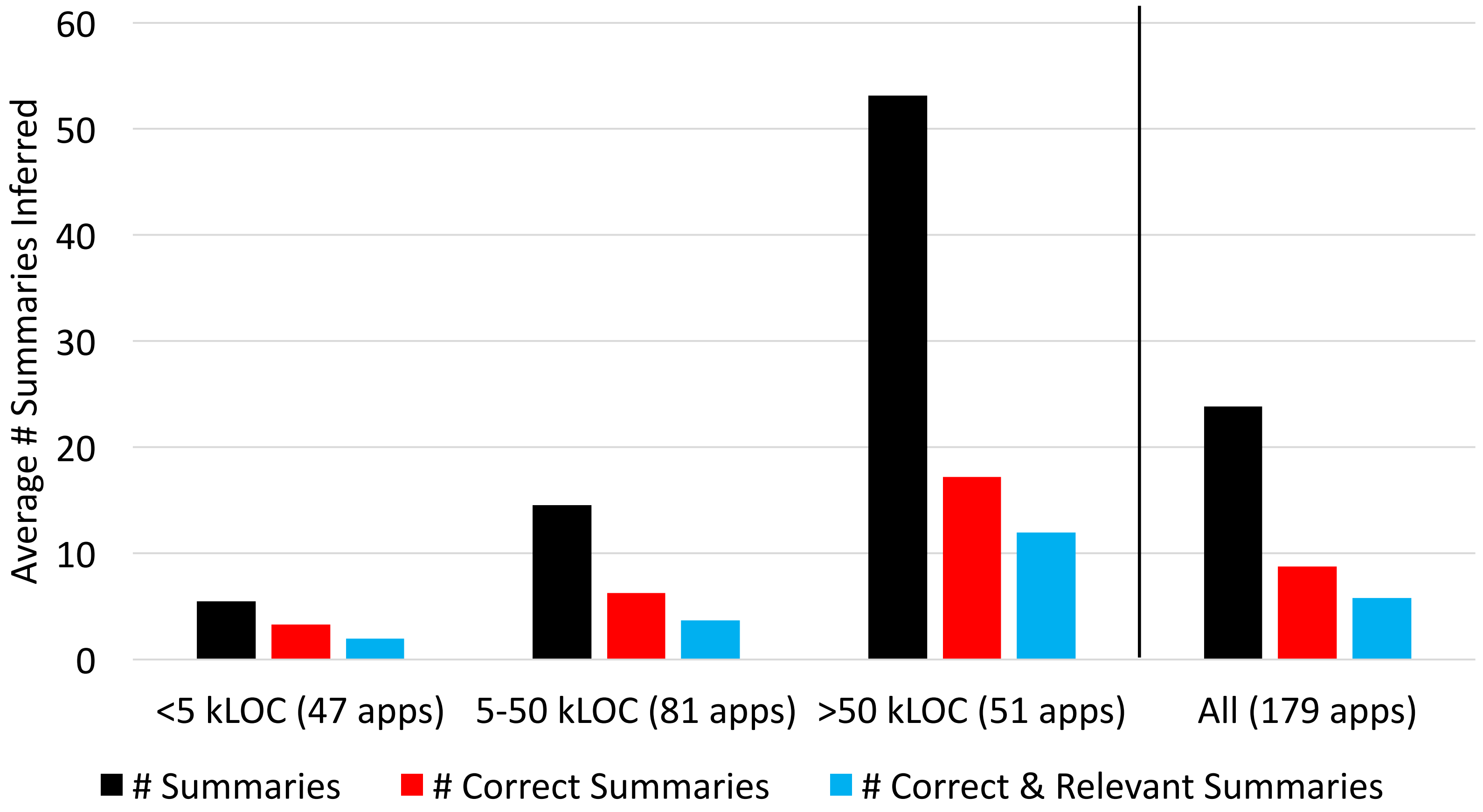
Evaluation: Flow Summaries



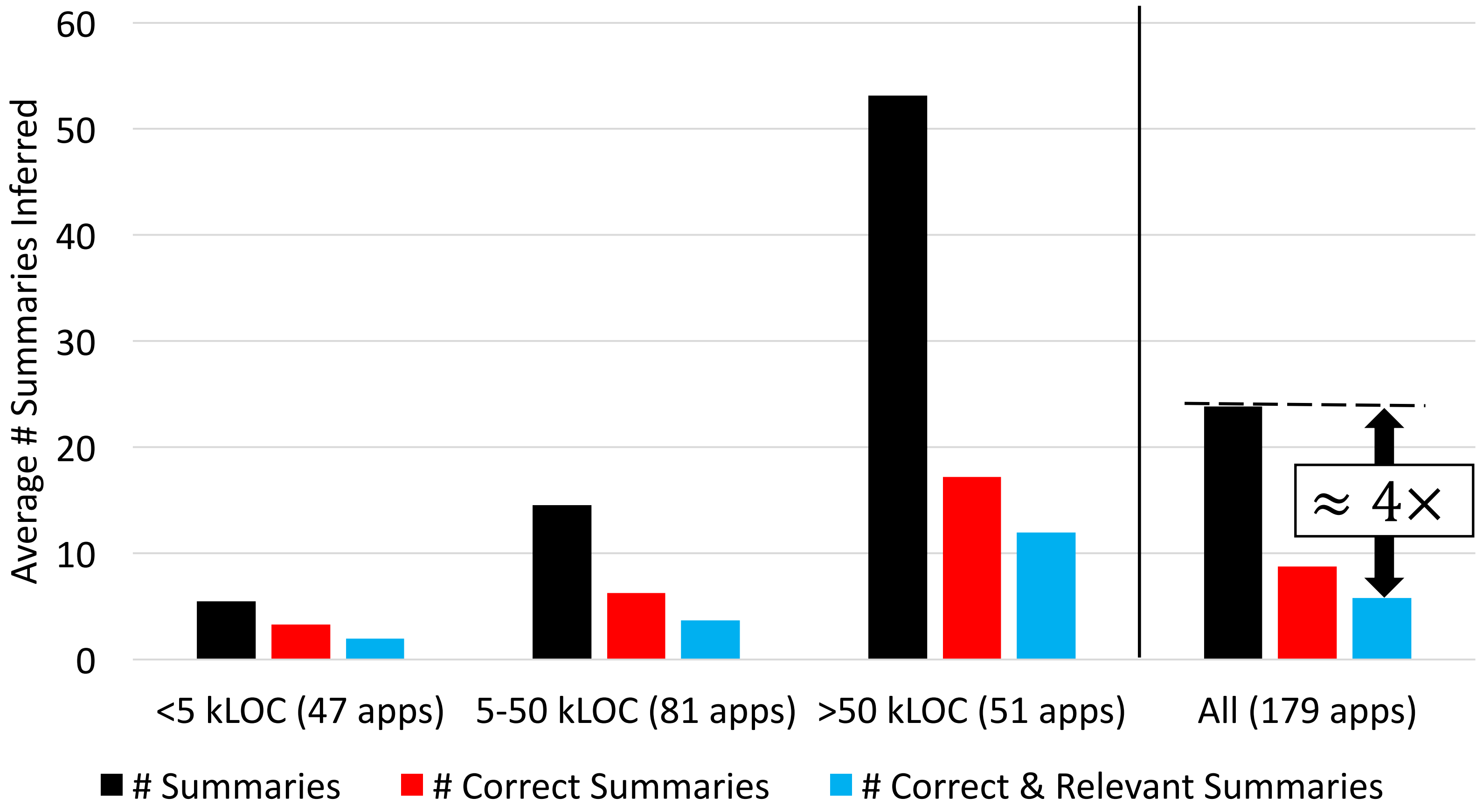
Evaluation: Flow Summaries



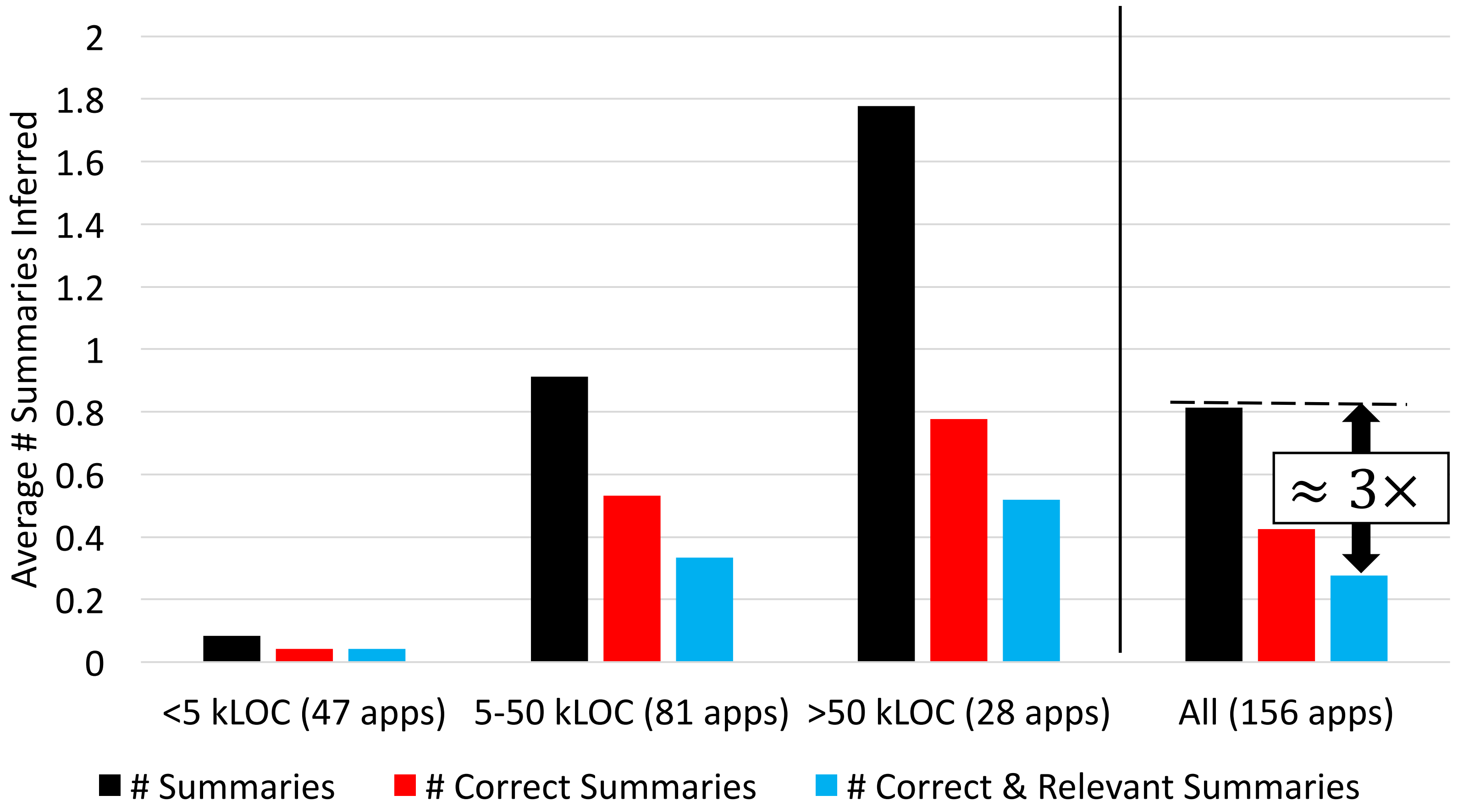
Evaluation: Flow Summaries



Evaluation: Flow Summaries



Evaluation: Alias Summaries



Human users hold valuable knowledge

- A little interaction goes a long way

Related Work

- Interactively inferring program invariants (Dillig 2012)
- Interactively inferring library specifications (Zhu 2013)

Follow-Up Work

Follow-Up Work

Untrusted Responses (OOPSLA 2015)

- Instrumentation to enforce responses

Follow-Up Work

Untrusted Responses (OOPSLA 2015)

- Instrumentation to enforce responses

“Interact” with Executions

- **Step 3:** ~~Analyst corrections~~ → Monitor executions

Follow-Up Work

Untrusted Responses (OOPSLA 2015)

- Instrumentation to enforce responses

“Interact” with Executions

- **Step 3:** ~~Analyst corrections~~ $\rightarrow$ Monitor executions

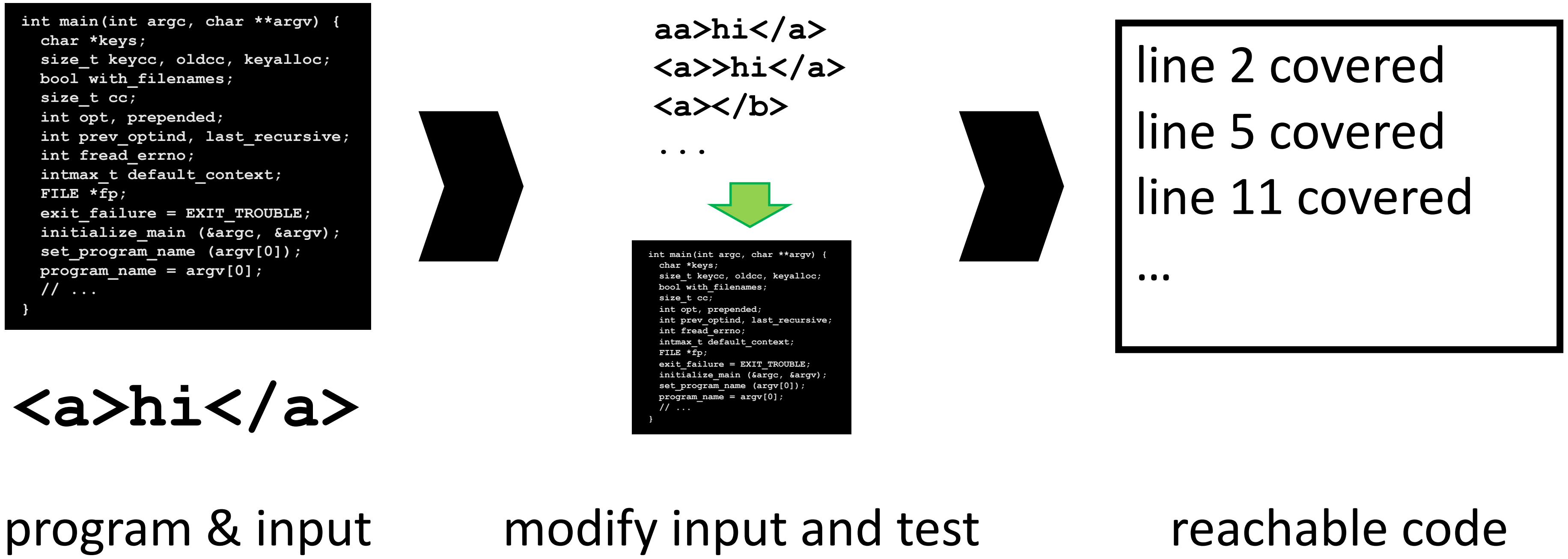
Automatic Test Generation

- **Input:** ~~Android app~~ $\rightarrow$ Synthesized test cases

Inferring Grammars for Fuzz Testing

Osbert Bastani, Rahul Sharma, Alex Aiken, and Percy Liang

PLDI 2017



```
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];
  // ...
}
```

```
<a>hi</a>
```

```
aa>hi</a>
<a>>hi</a>
<a></b>
...
```

```
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];
  // ...
}
```

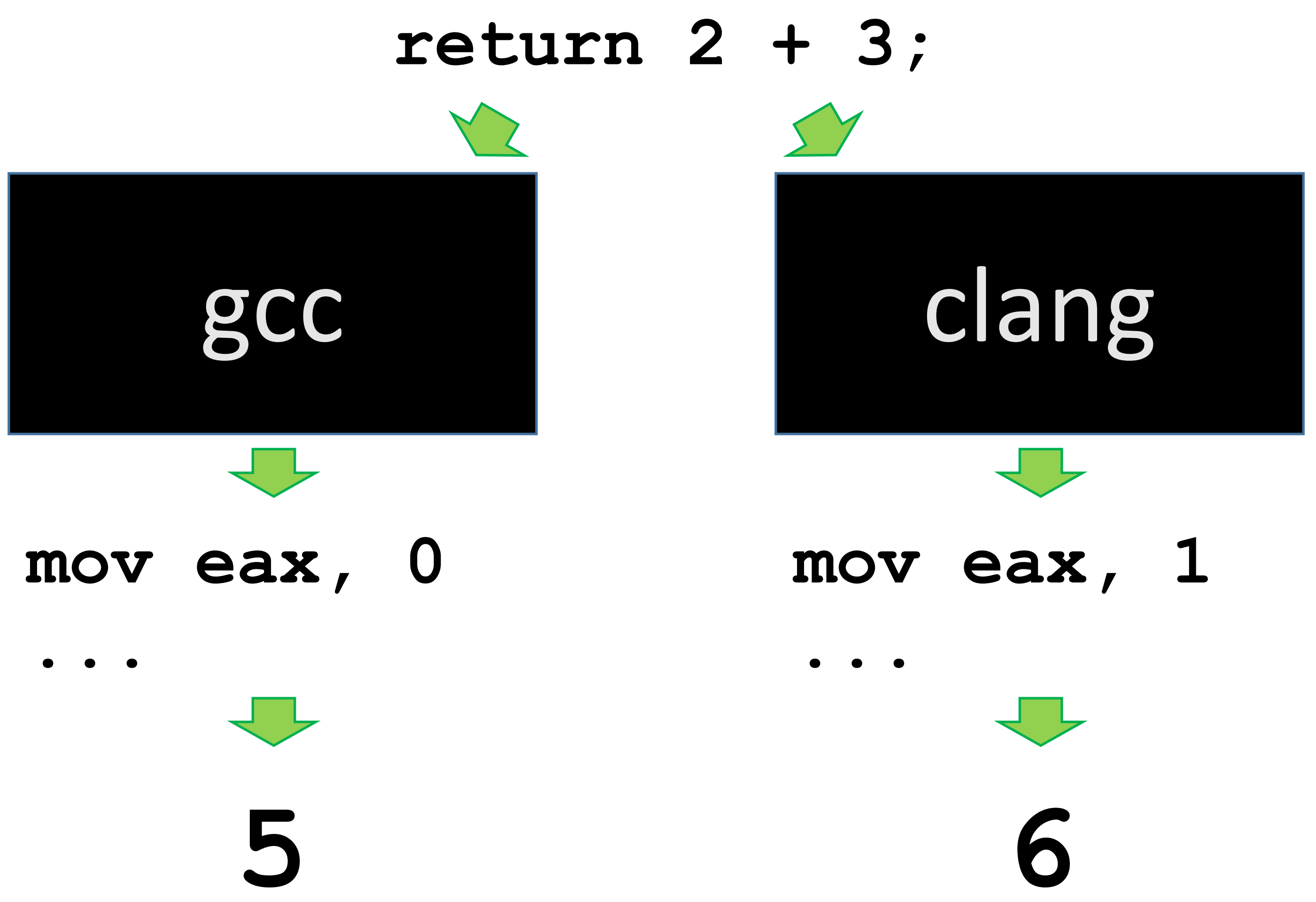
line 2 covered
line 5 covered
line 11 covered
...

Security Vulnerabilities (Miller 1990, …)

```
<<a>ai</a>
```

```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  bool with_filenames;
  size_t cc;
  int op                  ;
  int                              ive;
  int
  int
  FIL
  exit                           E;
  init                          rgv);
  set_program_name (argv[0]);
  program_name = argv[0];
  // ...
}
```

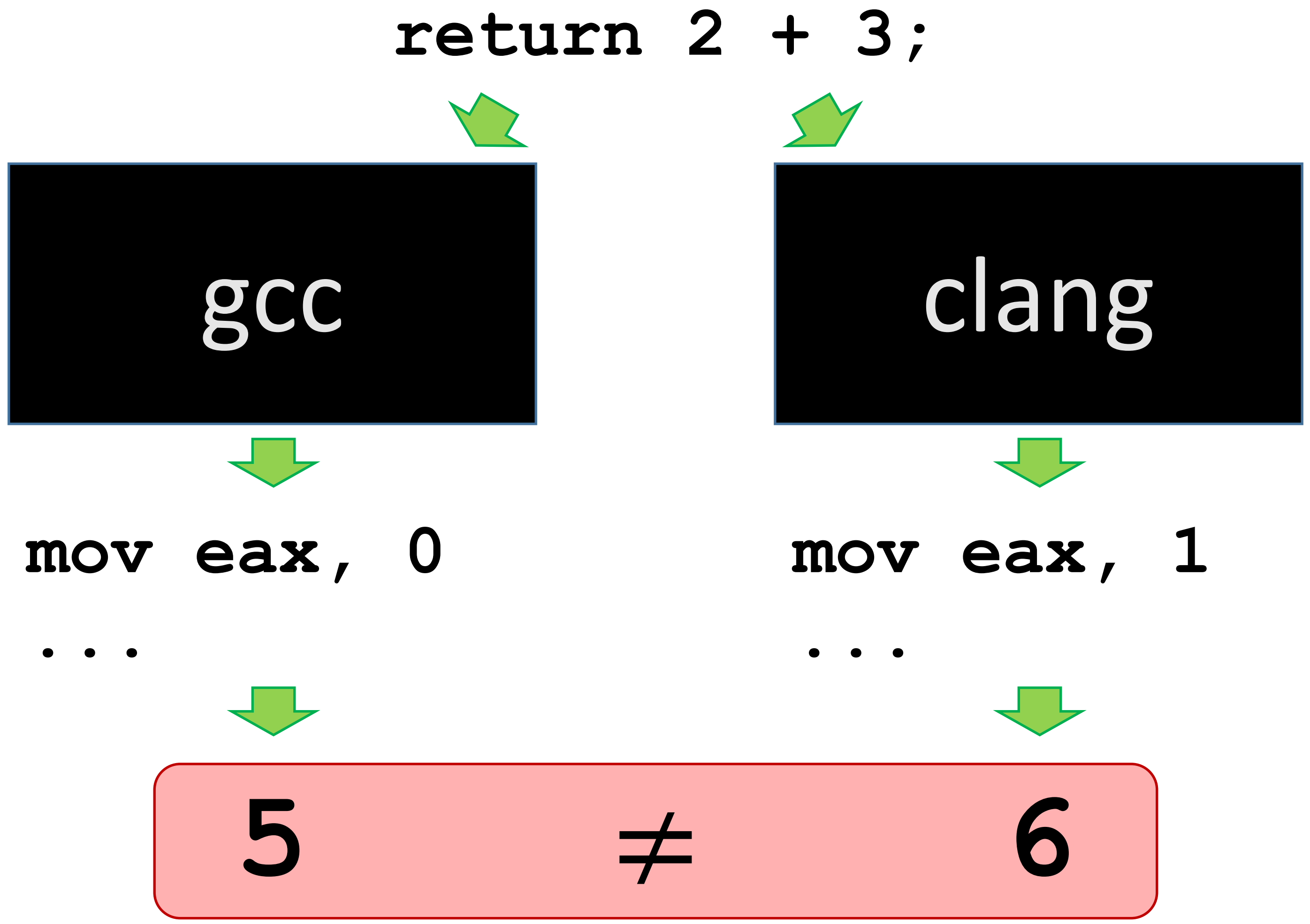
Compiler Bugs (Yang 2011, …)

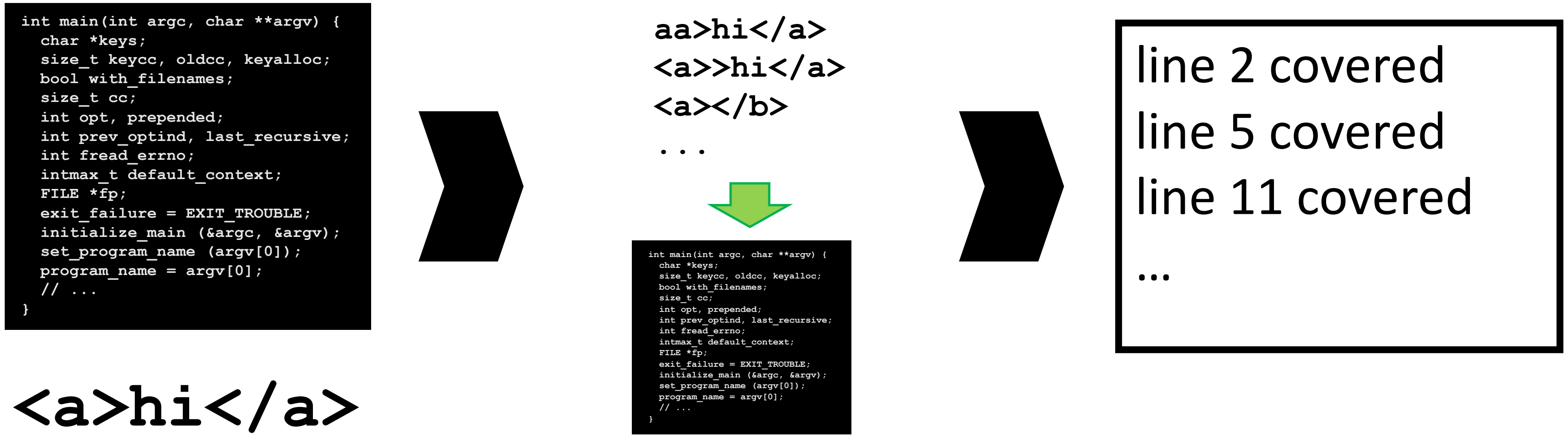

```
return 2 + 3;
```

```
mov eax, 0
...
```

```
mov eax, 1
...
```

Compiler Bugs (Yang 2011, …)





```
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];
  // ...
}
```

```
<a>hi</a>
```

```
aa>hi</a>
<a>>hi</a>
<a></b>
...
```

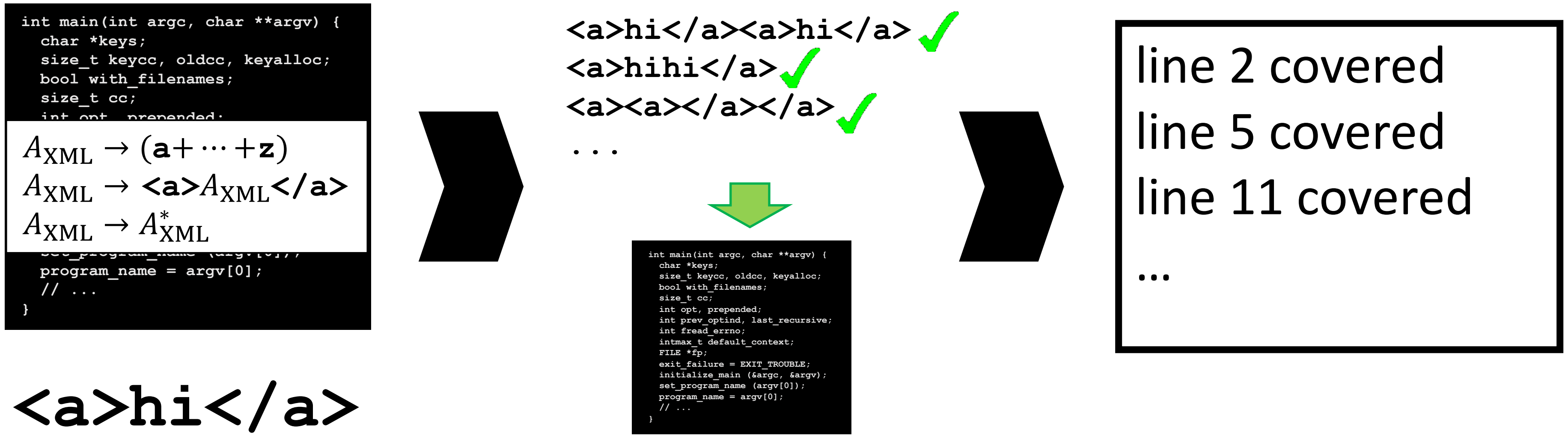
```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  bool with_filenames;
  size_t cc;
  int opt, prepended;
  int prev_optind, last_recursive;
  int fread_errno;
  intmax_t default_context;
  FILE *fp;
  exit_failure = EXIT_TROUBLE;
  initialize_main (&argc, &argv);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

```
line 2 covered
line 5 covered
line 11 covered
...
```

program & input

modify input and test

reachable code



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

$$A_{\text{XML}} \rightarrow (\mathbf{a}+\cdots+\mathbf{z})$$
$$A_{\text{XML}} \rightarrow \texttt{<a>}A_{\text{XML}}\texttt{</a>}$$
$$A_{\text{XML}} \rightarrow A_{\text{XML}}^{*}$$

```
  set_program_name (argv[0]);
  program_name = argv[0];
  // ...
}
```

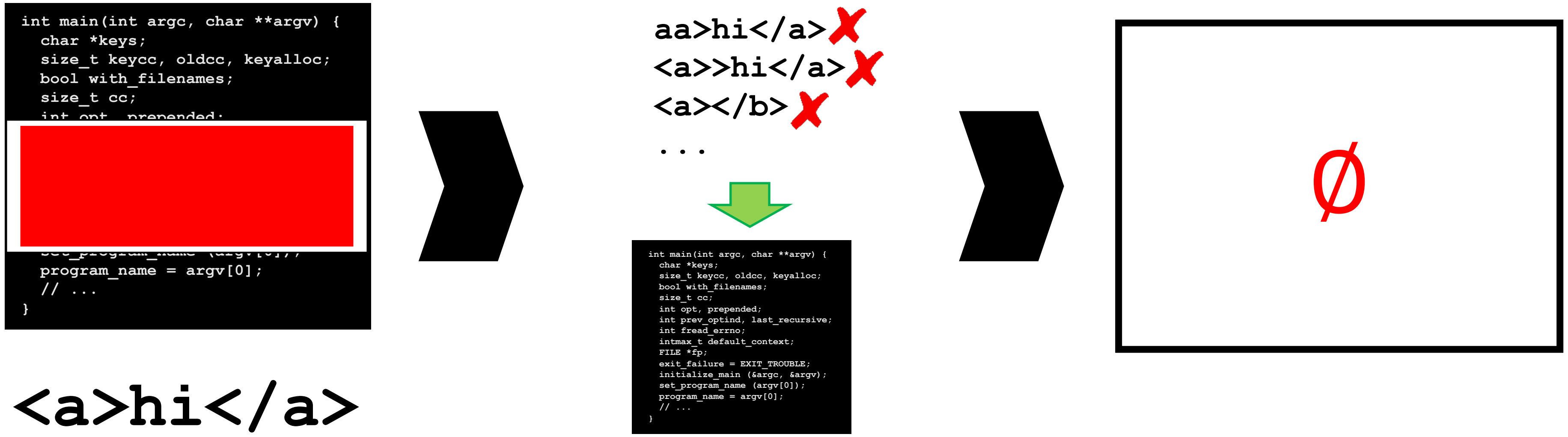
`<a>hi</a>`

```
<a>hi</a><a>hi</a> ✓
<a>hihi</a> ✓
<a><a></a></a> ✓
...
```

```
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];
  // ...
}
```

line 2 covered
line 5 covered
line 11 covered
...

program & input   modify input and test   reachable code



```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  bool with_filenames;
  size_t cc;
  int opt, prepended;
[redacted]
  program_name = argv[0];
  // ...
}
```

`<a>hi</a>`

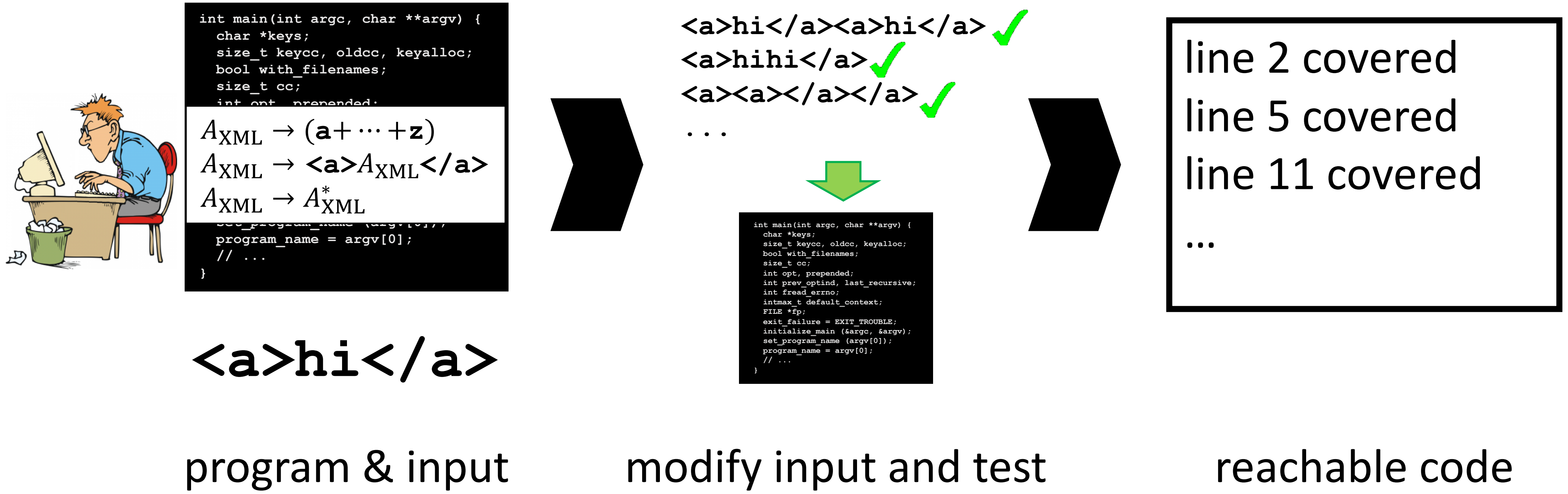
```
aa>hi</a> ✗
<a>>hi</a> ✗
<a></b> ✗
...
```

```
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];
  // ...
}
```

program & input

modify input and test

reachable code



```
int main(int argc, char **argv) {
  char *keys;
  size_t keycc, oldcc, keyalloc;
  bool with_filenames;
  size_t cc;
  int opt, prepended;
  set_program_name (argv[0]);
  program_name = argv[0];
  // ...
}
```

$A_{\text{XML}} \rightarrow (\mathtt{a} + \cdots + \mathtt{z})$

$A_{\text{XML}} \rightarrow \mathtt{<a>}A_{\text{XML}}\mathtt{</a>}$

$A_{\text{XML}} \rightarrow A_{\text{XML}}^{*}$

`<a>hi</a>`

program & input

```
<a>hi</a><a>hi</a> ✓
<a>hihi</a> ✓
<a><a></a></a> ✓
. . .
```

```
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];
  // ...
}
```

modify input and test

```
line 2 covered
line 5 covered
line 11 covered
...
```

reachable code

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

$A_{\mathrm{XML}} \rightarrow (\mathtt{a} + \cdots + \mathtt{z})$

$A_{\mathrm{XML}} \rightarrow \mathtt{<a>}A_{\mathrm{XML}}\mathtt{</a>}$

$A_{\mathrm{XML}} \rightarrow A_{\mathrm{XML}}^{*}$

```
  program_name = argv[0];
  // ...
}
```

`<a>hi</a>`

program & input

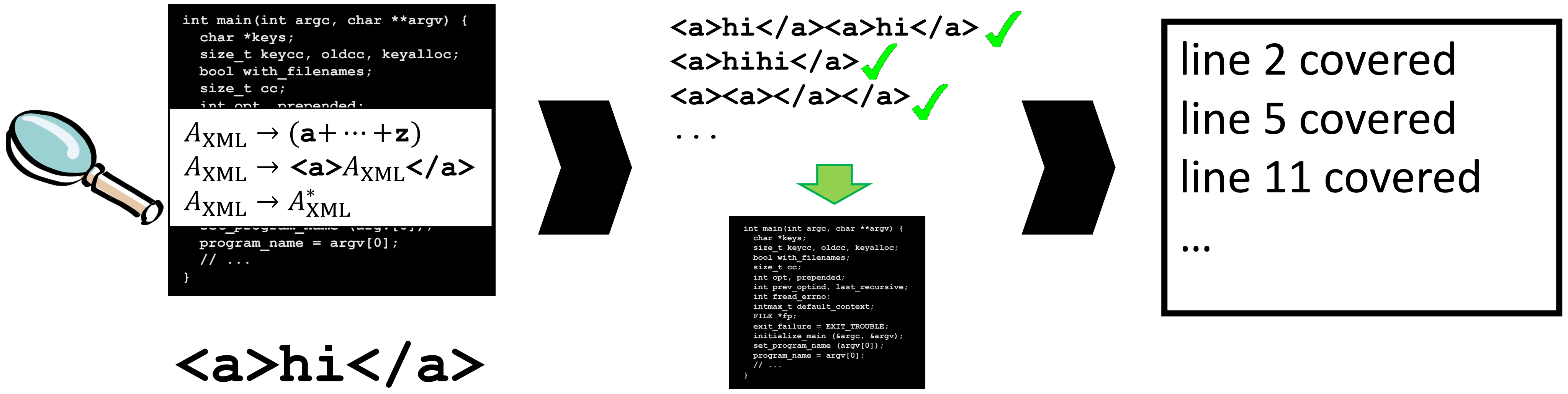
```
<a>hi</a><a>hi</a> ✓
<a>hihi</a> ✓
<a><a></a></a> ✓
. . .
```

```
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];
  // ...
}
```

modify input and test

line 2 covered
line 5 covered
line 11 covered
...

reachable code



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

$A_{\mathrm{XML}} \to (\mathtt{a}+\cdots+\mathtt{z})$

$A_{\mathrm{XML}} \to \mathtt{<a>}A_{\mathrm{XML}}\mathtt{</a>}$

$A_{\mathrm{XML}} \to A_{\mathrm{XML}}^{*}$

```
  set_program_name (argv[0]);
  program_name = argv[0];
  // ...
}
```

`<a>hi</a>`

```
<a>hi</a><a>hi</a> ✓
<a>hihi</a> ✓
<a><a></a></a> ✓
. . .
```

```
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];
  // ...
}
```

line 2 covered
line 5 covered
line 11 covered
...

program & input

modify input and test

reachable code

Grammar Synthesis Algorithm

$\alpha_{\text{in}} = \texttt{<a>hi</a>}$

$$\mathcal{O}_{\text{XML}}(\alpha) = \begin{cases} 1 & \text{if } \alpha \in L_{\text{XML}} \\ 0 & \text{otherwise} \end{cases}$$

input example & membership oracle

$A_{\text{XML}} \to (\texttt{a}+\cdots+\texttt{z})$
$A_{\text{XML}} \to \texttt{<a>}A_{\text{XML}}\texttt{</a>}$
$A_{\text{XML}} \to A_{\text{XML}}^*$

grammar approximating **target language** L_{XML}

Idea: Construct a series of increasingly general languages

Idea: Construct a series of increasingly general languages

$\alpha_{\text{in}} =$ **`<a>hi</a>`**

Idea: Construct a series of increasingly general languages

$\alpha_{\text{in}} = \texttt{<a>hi</a>} \subseteq (\texttt{<a>hi</a>})^*$

Idea: Construct a series of increasingly general languages

$$\alpha_{\text{in}} = \texttt{<a>hi</a>} \subseteq (\texttt{<a>hi</a>})^* \subseteq (\texttt{<a>}(\texttt{hi})^*\texttt{</a>})^*$$

Idea: Construct a series of increasingly general languages

$$\alpha_{\text{in}} = \texttt{<a>hi</a>} \subseteq (\texttt{<a>hi</a>})^* \subseteq (\texttt{<a>}(\texttt{hi})^*\texttt{</a>})^* \subseteq \ldots$$

Idea: Construct a series of increasingly general languages

$$\begin{tikzcd}
 & & \text{generalization steps} \arrow[dl] \arrow[dr] \arrow[drrr] & & & & \\
\alpha_{\text{in}} = \texttt{<a>hi</a>} & \subseteq & (\texttt{<a>hi</a>})^* & \subseteq & (\texttt{<a>}(\texttt{hi})^*\texttt{</a>})^* & \subseteq & \ldots
\end{tikzcd}$$

Idea: Construct a series of increasingly general languages

generalization steps

$$\alpha_{\text{in}} = \texttt{<a>hi</a>} \subseteq (\texttt{<a>hi</a>})^* \subseteq (\texttt{<a>}(\texttt{hi})^*\texttt{</a>})^* \subseteq \ldots$$

Idea: Construct a series of increasingly general languages

generalization steps

$$\alpha_{\text{in}} = \boxed{\texttt{<a>hi</a>} \subseteq (\texttt{<a>hi</a>})^*} \subseteq (\texttt{<a>}(\texttt{hi})^*\texttt{</a>})^* \subseteq \dots$$

current language L_i

Idea: Construct a series of increasingly general languages

generalization steps

$$\alpha_{\text{in}} = \boxed{\texttt{<a>hi</a>} \subseteq (\texttt{<a>hi</a>})^*} \subseteq (\texttt{<a>}(\texttt{hi})^*\texttt{</a>})^* \subseteq \dots$$

current language L_i

generalized language L_{i+1}

Idea: Construct a series of increasingly general languages

generalization steps

$$\alpha_{\text{in}} = \boxed{\texttt{<a>hi</a>} \subseteq (\texttt{<a>hi</a>})^*} \subseteq (\texttt{<a>}(\texttt{hi})^*\texttt{</a>})^* \subseteq \ldots$$

current language L_i generalized language L_{i+1}

Monotone: $L_{i+1} \supseteq L_i$

Idea: Construct a series of increasingly general languages

generalization steps

$\alpha_{\text{in}} = \texttt{<a>hi</a>} \subseteq (\texttt{<a>hi</a>})^* \subseteq (\texttt{<a>}(\texttt{hi})^*\texttt{</a>})^* \subseteq \ldots$

current language L_i generalized language L_{i+1}

Monotone: $L_{i+1} \supseteq L_i$

Precise: $L_{i+1} \subseteq L_{\text{XML}}$

Idea: Construct a series of increasingly general languages

generalization steps

$$\alpha_{\text{in}} = \boxed{\texttt{<a>hi</a>} \subseteq (\texttt{<a>hi</a>})^*} \subseteq (\texttt{<a>}(\texttt{hi})^*\texttt{</a>})^* \subseteq \dots$$

current language L_i generalized language L_{i+1}

Monotone: $L_{i+1} \supseteq L_i$

Precise: $L_{i+1} \setminus L_i \subseteq L_{\text{XML}}$

Generalization Step

Generalization Step

```
<a>hi</a>
```

current
language

Generalization Step

`<a>hi</a>`

$\rightarrow$

$($**`<a>hi</a>`**$)^*$

$\vdots$

$($**`<`**$)^*$**`a>hi</a>`**

$\vdots$

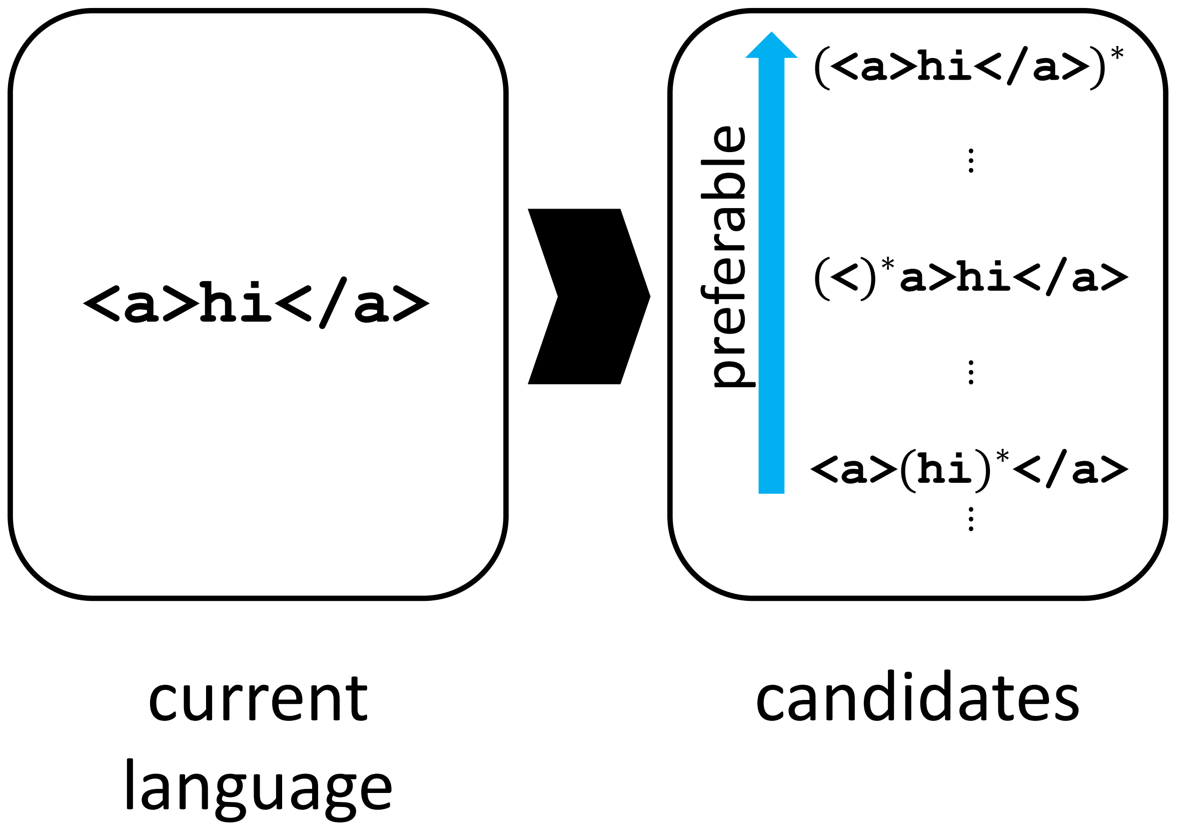
`<a>`$($**`hi`**$)^*$**`</a>`**

$\vdots$

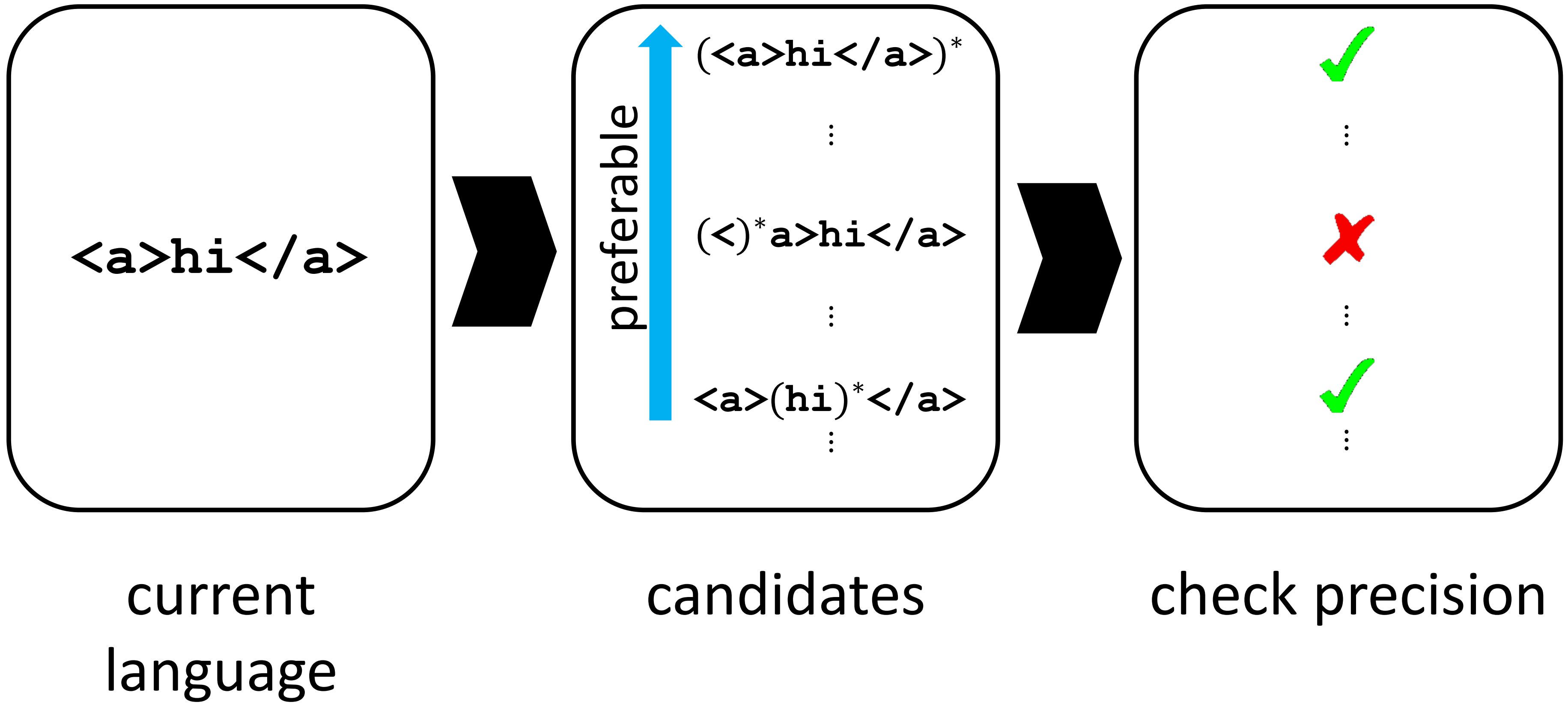
current language

candidates

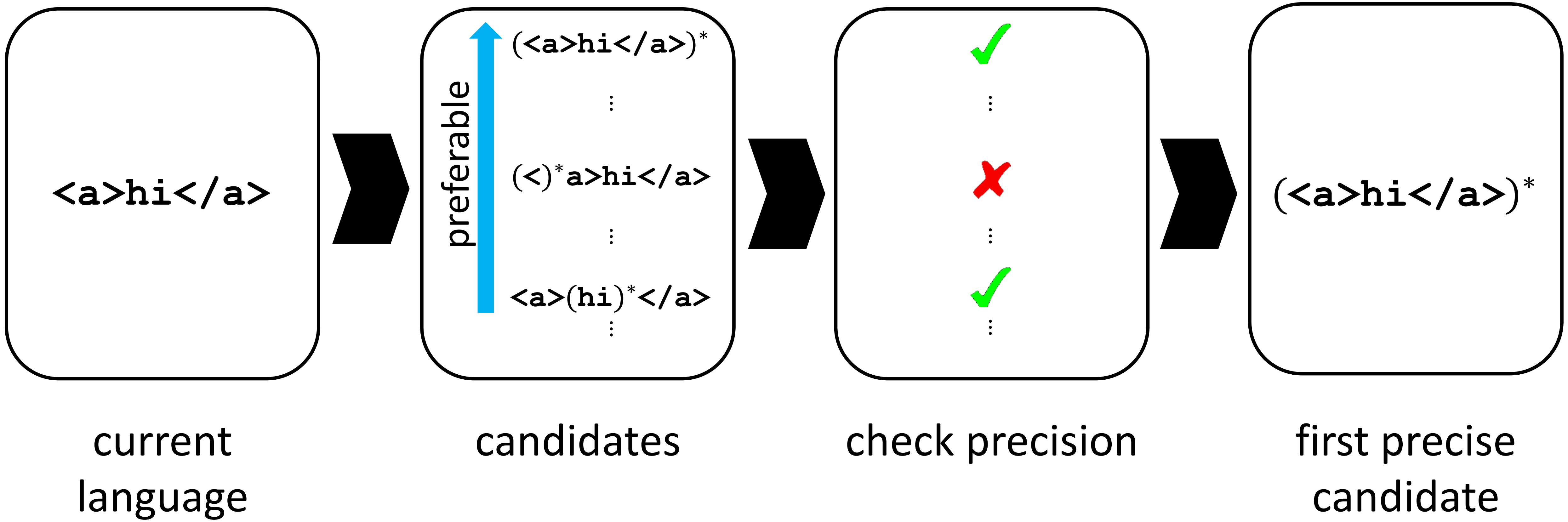
Generalization Step



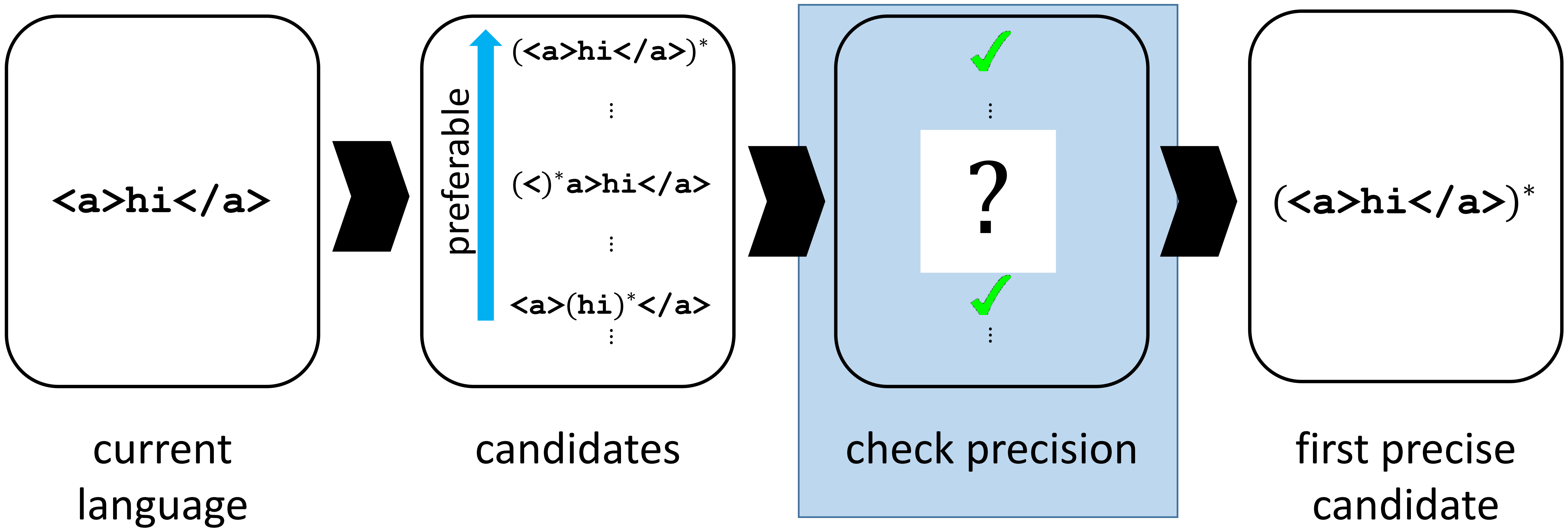
Generalization Step



Generalization Step



Generalization Step



Check Precision

For every $\alpha \in (\texttt{<a>hi</a>})^* \setminus \texttt{<a>hi</a>}$:

$$\alpha \in L_{\text{XML}}$$

Check Precision

For every $\alpha \in (\texttt{<a>hi</a>})^* \setminus \texttt{<a>hi</a>}$:

$$\mathcal{O}_{\mathrm{XML}}(\alpha) = 1$$

Check Precision

infinite!

For every $\alpha \in (\texttt{<a>hi</a>})^* \setminus \texttt{<a>hi</a>}$:

$$\mathcal{O}_{\mathrm{XML}}(\alpha) = 1$$

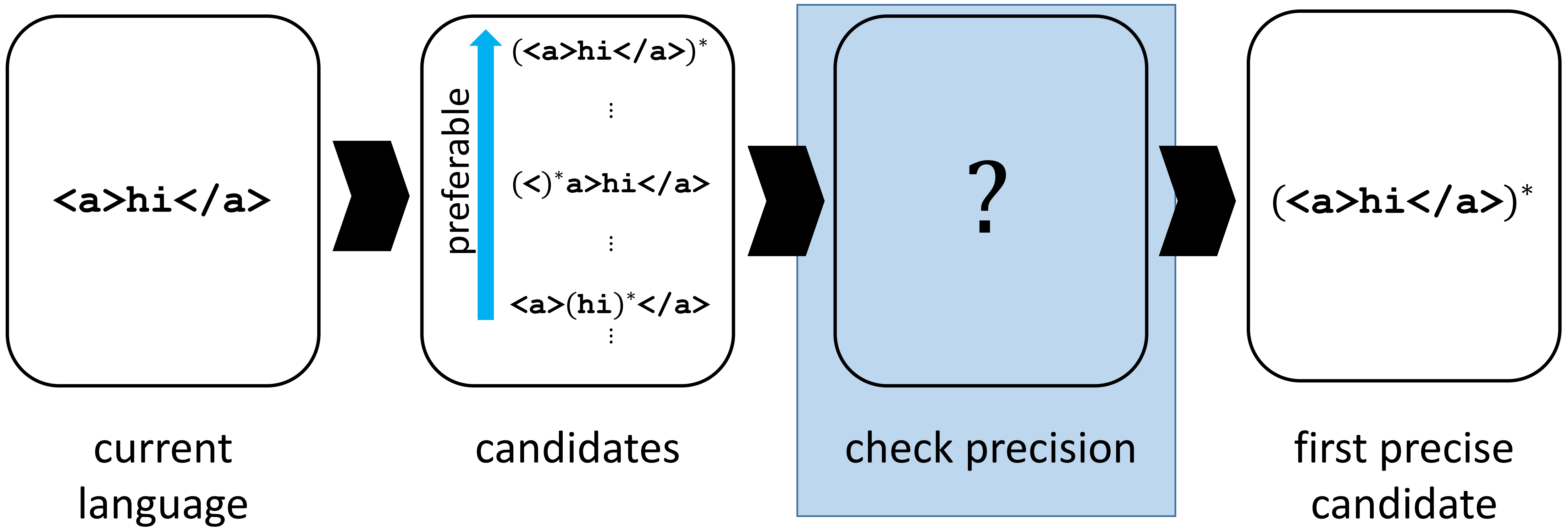
Check Potential Precision

finite subset of **checks**

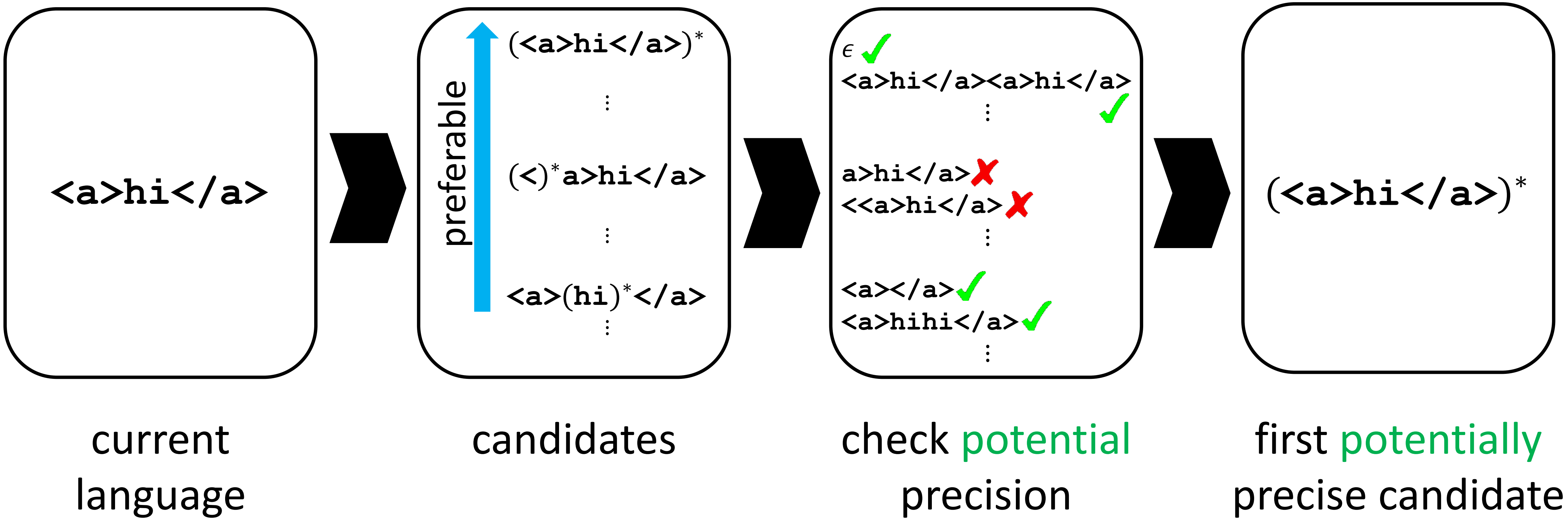
For every $\alpha \in \boldsymbol{S} \subseteq (\texttt{<a>hi</a>})^* \setminus \texttt{<a>hi</a>}$:

$$\mathcal{O}_{\mathrm{XML}}(\alpha) = 1$$

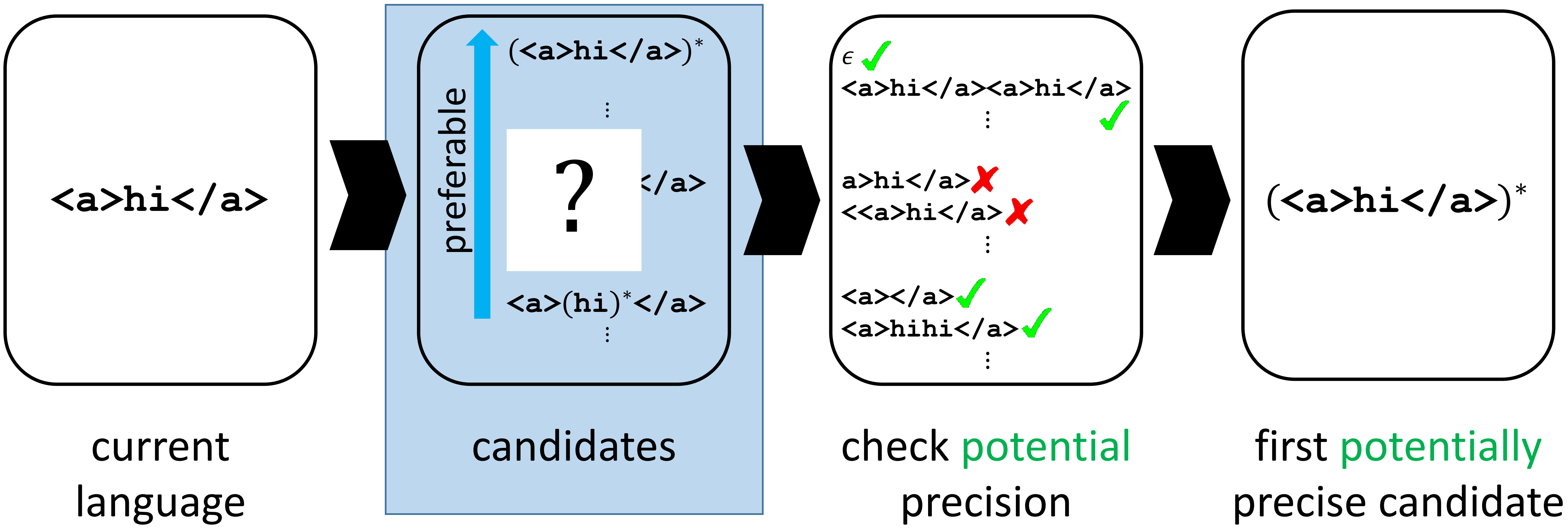
Generalization Step



Generalization Step



Generalization Step



```
<a>hi</a>
```

$\texttt{<a>hi</a>}$

$\Rightarrow (\texttt{<a>hi</a>})^*$

$\Rightarrow (\texttt{<a>}(\texttt{hi})^*\texttt{</a>})^*$

$\Rightarrow (\texttt{<a>}(\texttt{h}+\texttt{i})^*\texttt{</a>})^*$

$$\texttt{<a>hi</a>}$$
$$\Rightarrow (\texttt{<a>hi</a>})^*$$
$$\Rightarrow (\texttt{<a>}(\texttt{hi})^*\texttt{</a>})^*$$
$$\Rightarrow (\texttt{<a>}(\texttt{h}+\texttt{i})^*\texttt{</a>})^*$$
$$\Rightarrow \boxed{\begin{array}{l} A \to (\texttt{h}+\texttt{i})^* \\ B \to (\texttt{<a>}A\texttt{</a>})^* \end{array}}$$

$\texttt{<a>hi</a>}$

$\Rightarrow (\texttt{<a>hi</a>})^*$

$\Rightarrow (\texttt{<a>}(\texttt{hi})^*\texttt{</a>})^*$

$\Rightarrow (\texttt{<a>}(\texttt{h}+\texttt{i})^*\texttt{</a>})^*$

$$\Rightarrow \boxed{\begin{array}{l} A \to (\texttt{h}+\texttt{i})^* \\ B \to (\texttt{<a>}A\texttt{</a>})^* \end{array}}$$

$$\Rightarrow \boxed{\begin{array}{l} A \to (\texttt{h}+\texttt{i})^* \\ A \to (\texttt{<a>}A\texttt{</a>})^* \end{array}}$$

$\texttt{<a>hi</a>}$

$\Rightarrow (\texttt{<a>hi</a>})^*$

$\Rightarrow (\texttt{<a>}(\texttt{hi})^*\texttt{</a>})^*$

$\Rightarrow (\texttt{<a>}(\texttt{h}+\texttt{i})^*\texttt{</a>})^*$

$\Rightarrow \boxed{\begin{array}{l} A \rightarrow (\texttt{h}+\texttt{i})^* \\ B \rightarrow (\texttt{<a>}A\texttt{</a>})^* \end{array}}$

$\Rightarrow \boxed{\begin{array}{l} A \rightarrow (\texttt{h}+\texttt{i})^* \\ A \rightarrow (\texttt{<a>}A\texttt{</a>})^* \end{array}}$

$\Rightarrow \boxed{\begin{array}{l} A \rightarrow (\texttt{a}+\cdots+\texttt{z})^* \\ A \rightarrow (\texttt{<a>}A\texttt{</a>})^* \end{array}}$

Evaluation

Grammar learning:	Compare to existing algorithms
Fuzz testing:	Compare to existing fuzzers

Evaluation: Grammar Learning

Baselines:	L-Star, RPNI
Grammars:	URL, Grep, LISP, XML
Inputs:	membership oracle $\mathcal{O}$ 50 random strings $E_{\mathrm{in}} \subseteq L_*$

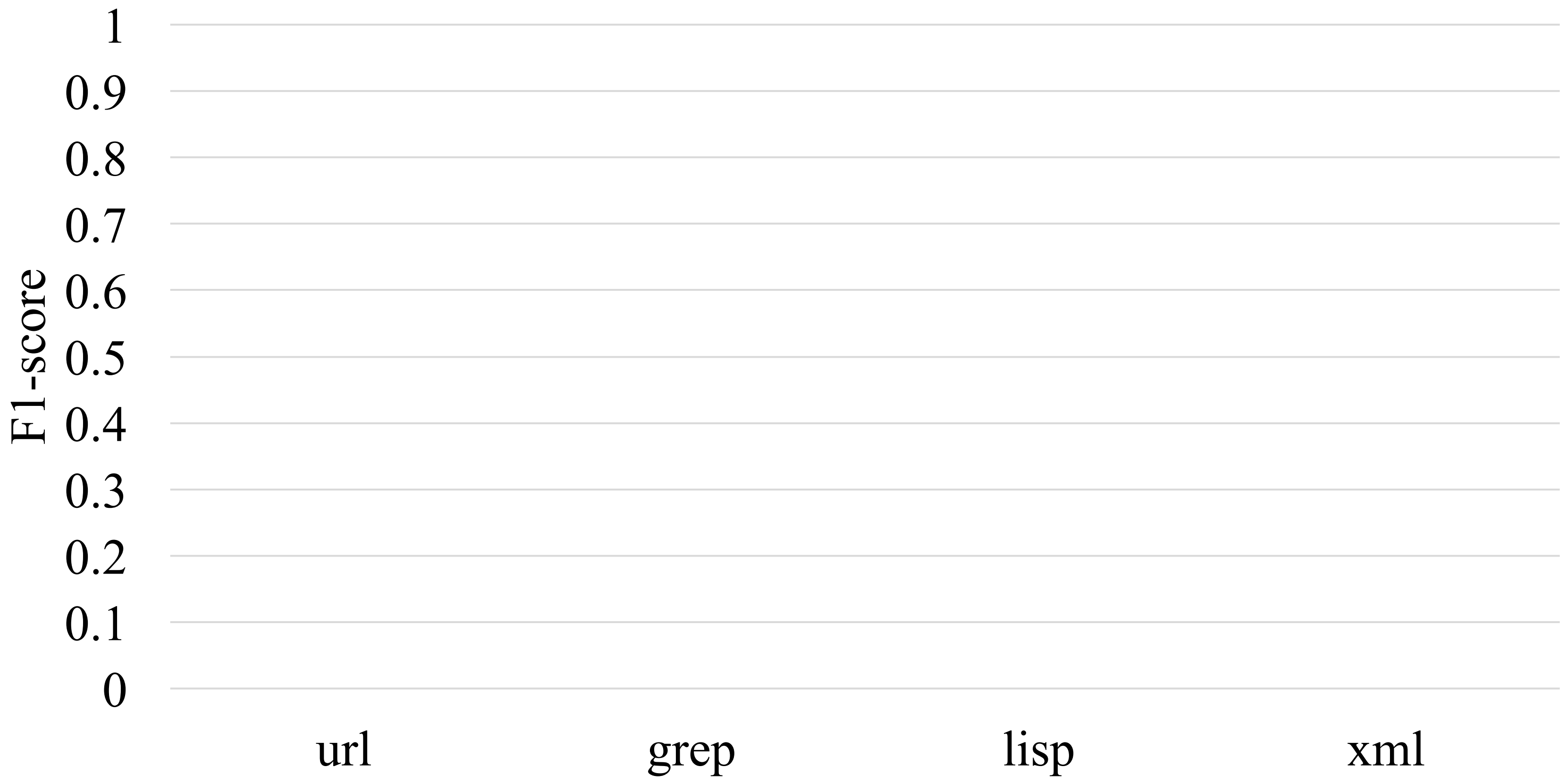
Evaluation: Grammar Learning

Precision: $$\frac{\#\text{ valid sampled inputs}}{\#\text{ sampled inputs}}$$

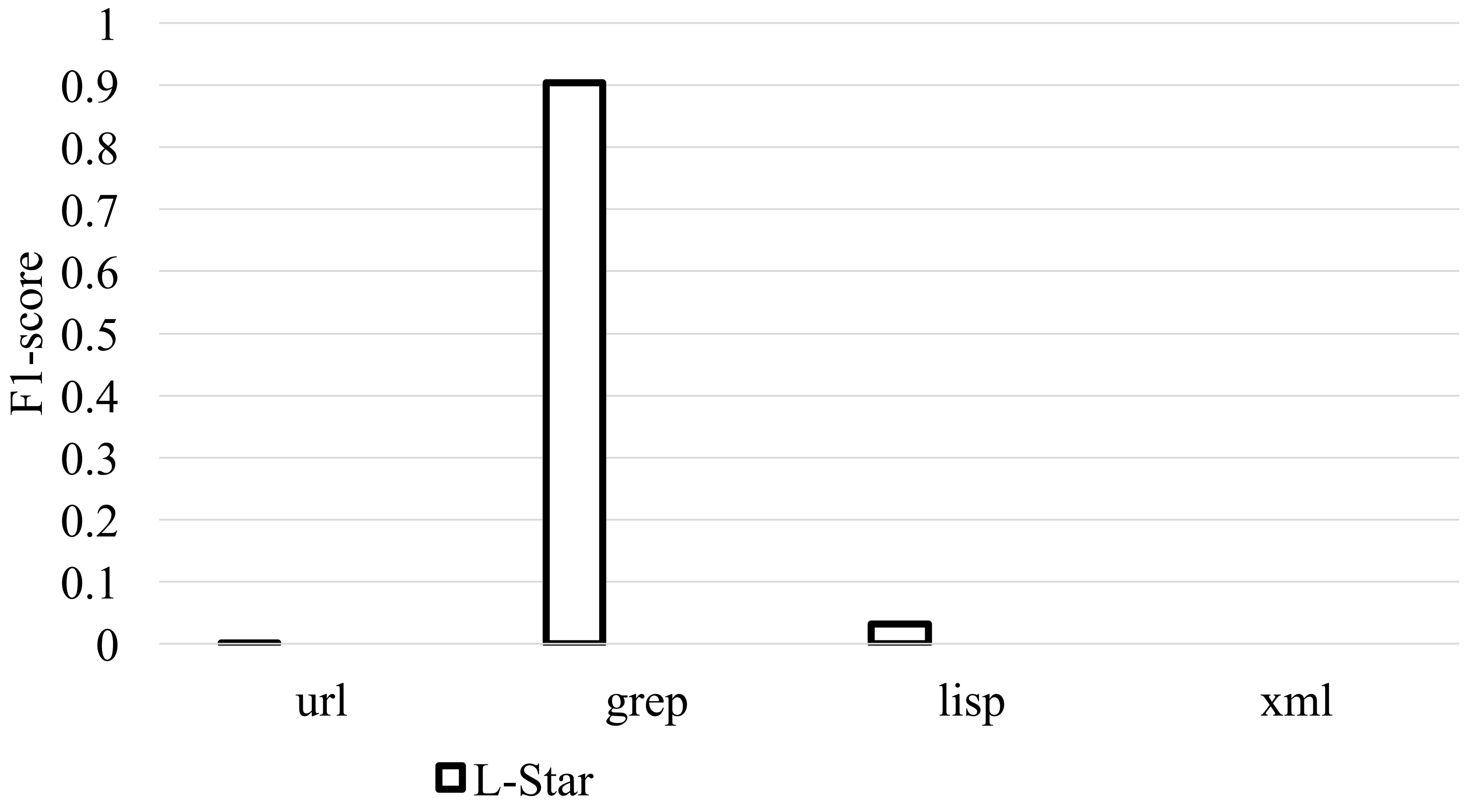
Recall: $$\frac{\#\text{ true inputs that might be sampled}}{\#\text{ true inputs}}$$

F_1-Score: $$\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

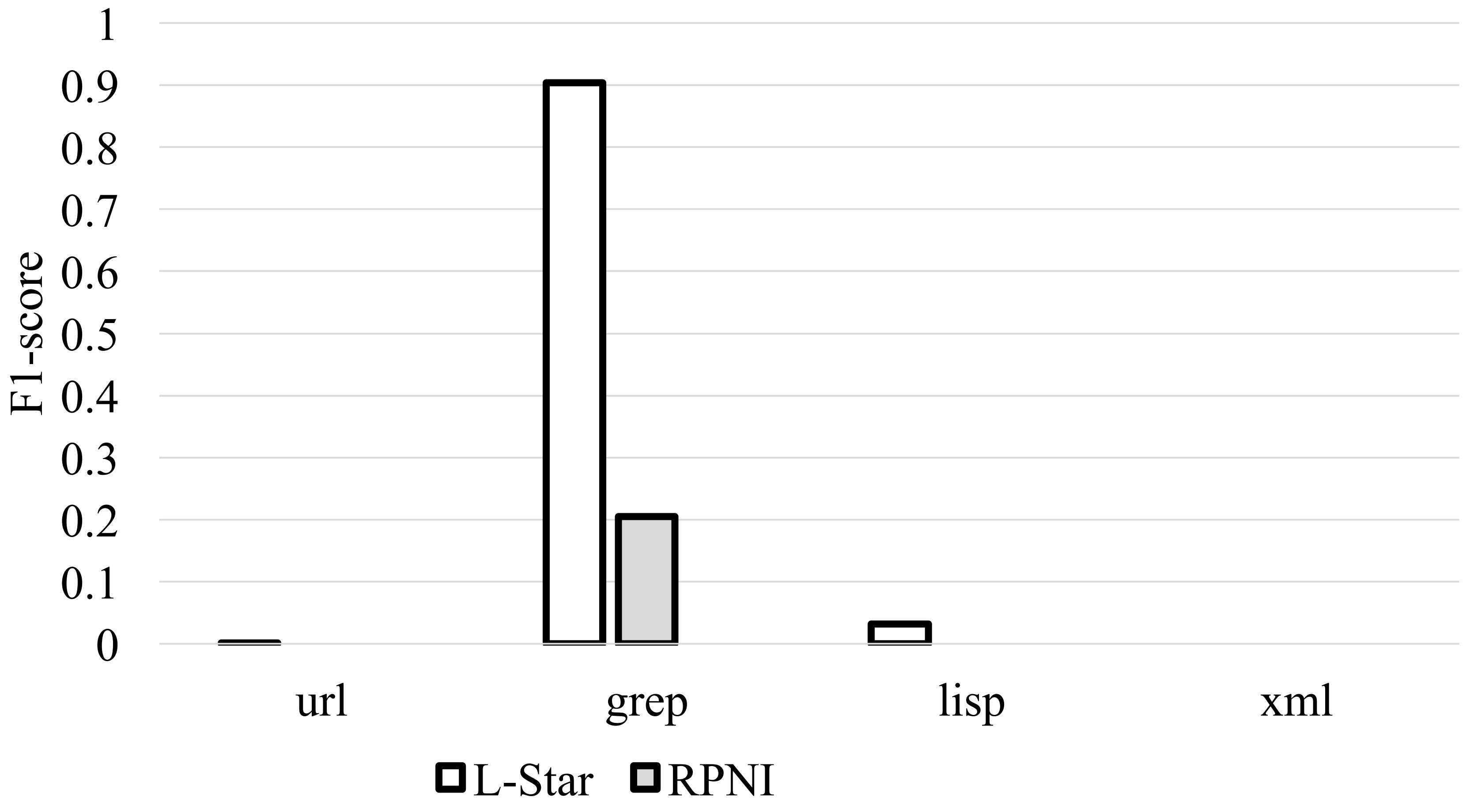
Evaluation: Grammar Learning



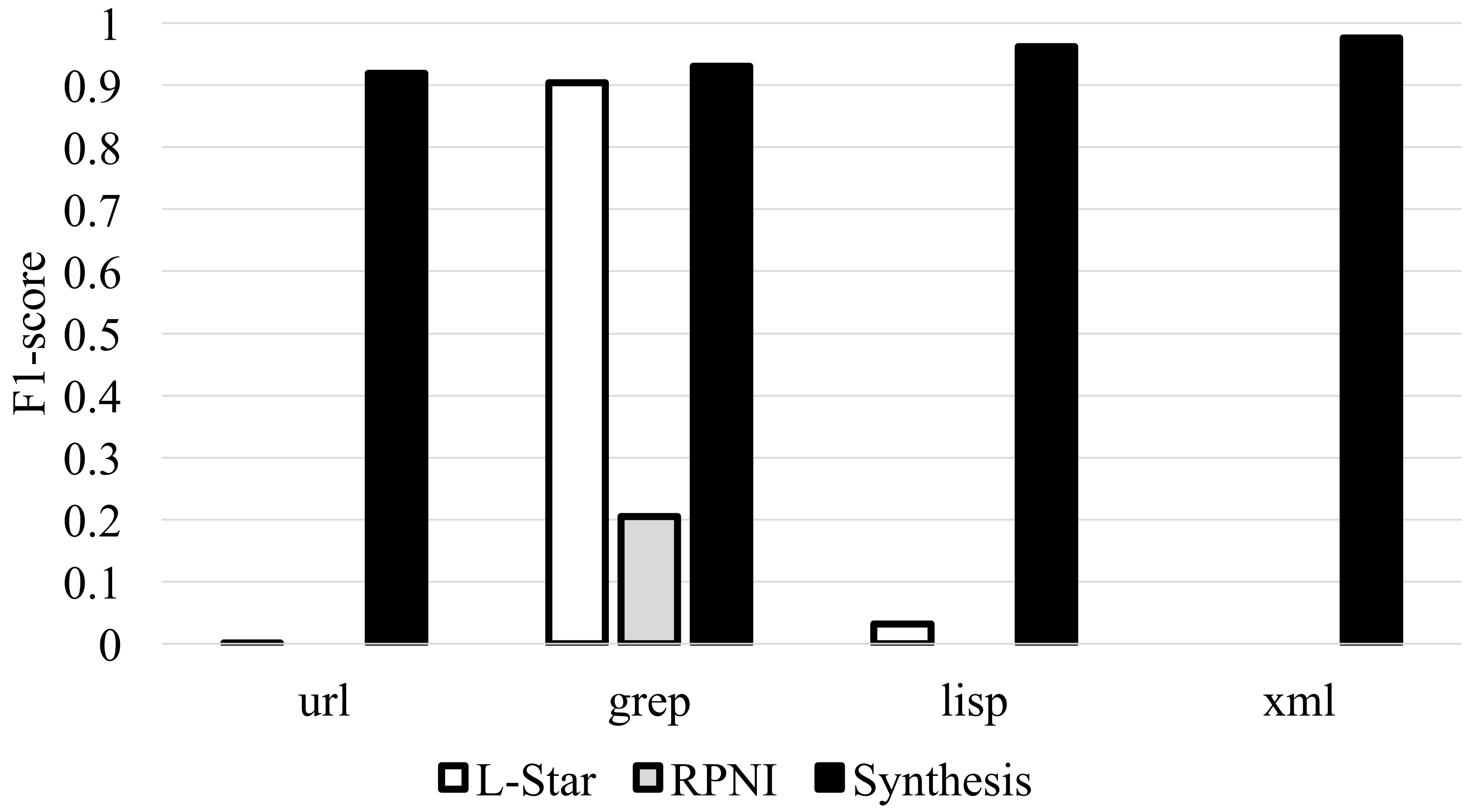
Evaluation: Grammar Learning



Evaluation: Grammar Learning



Evaluation: Grammar Learning



Evaluation: Fuzz Testing

Fuzzer:	synthesize grammar, randomly resample subtrees of parse tree
Baselines:	naïve (random insertions/deletions) afl-fuzz (production fuzzer)
Programs:	Grep, Sed, Flex, Bison, XML Parser Python, Ruby, SpiderMonkey (parser only)

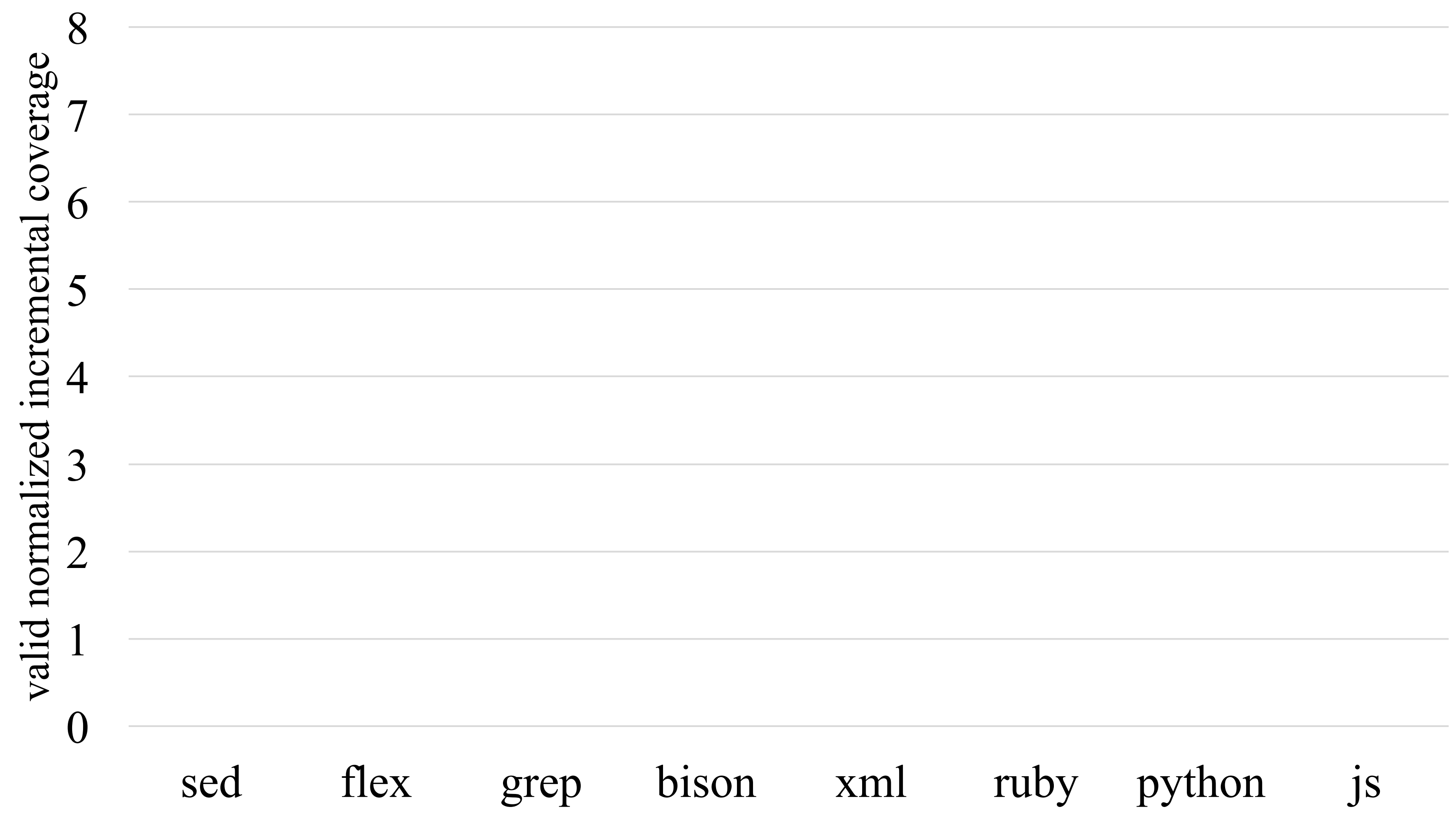
Evaluation: Fuzz Testing

Valid coverage: $\mathrm{Cov}(E) = \#(\text{lines covered by } E \cap L_*)$

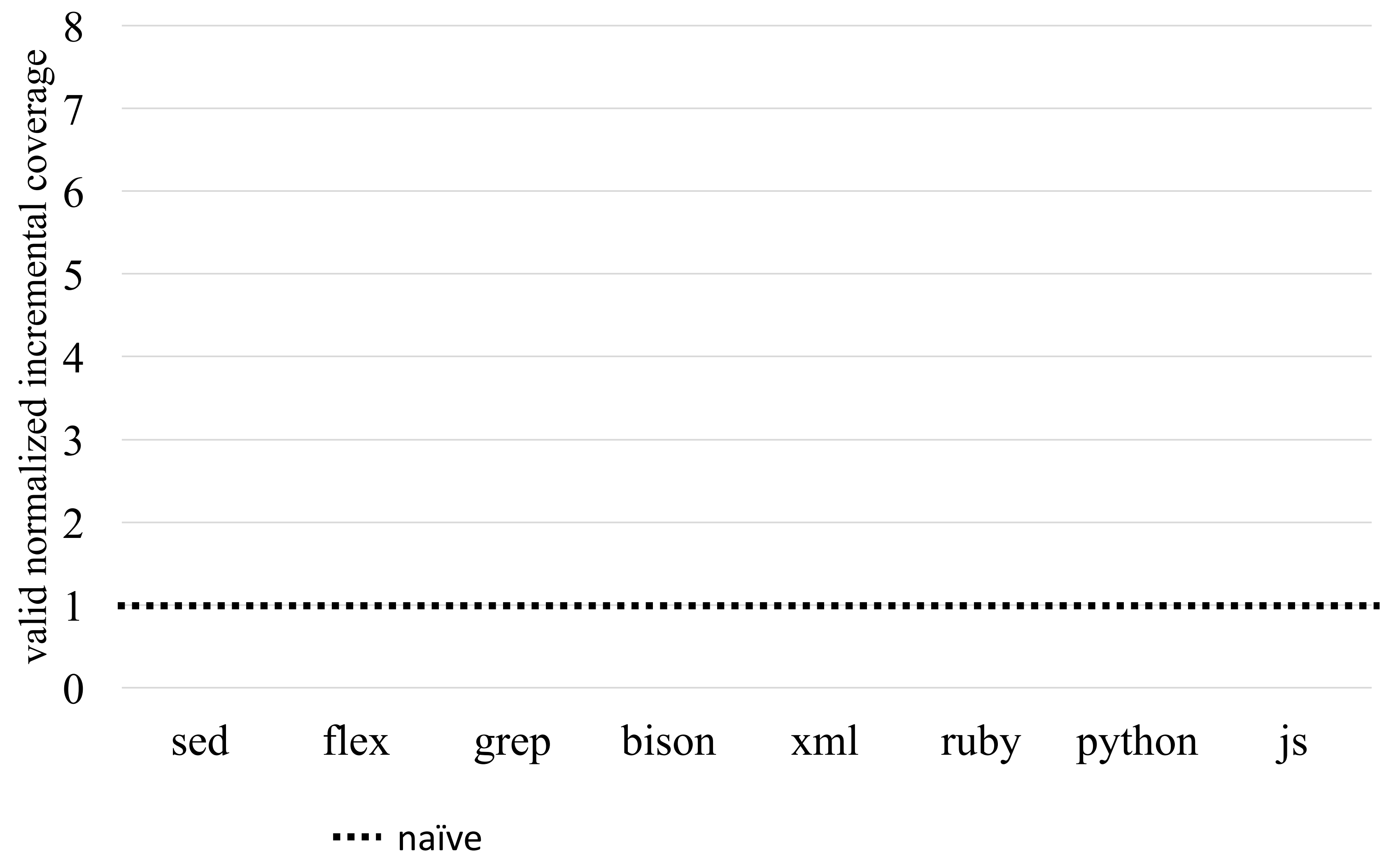
Incremental coverage: $\mathrm{IncCov}(E) = \mathrm{Cov}(E) - \mathrm{Cov}(\alpha_{\mathrm{in}})$

Normalized: $\mathrm{NormIncCov}(E) = \dfrac{\mathrm{IncCov}(E)}{\mathrm{IncCov}(E_{\text{naïve}})}$

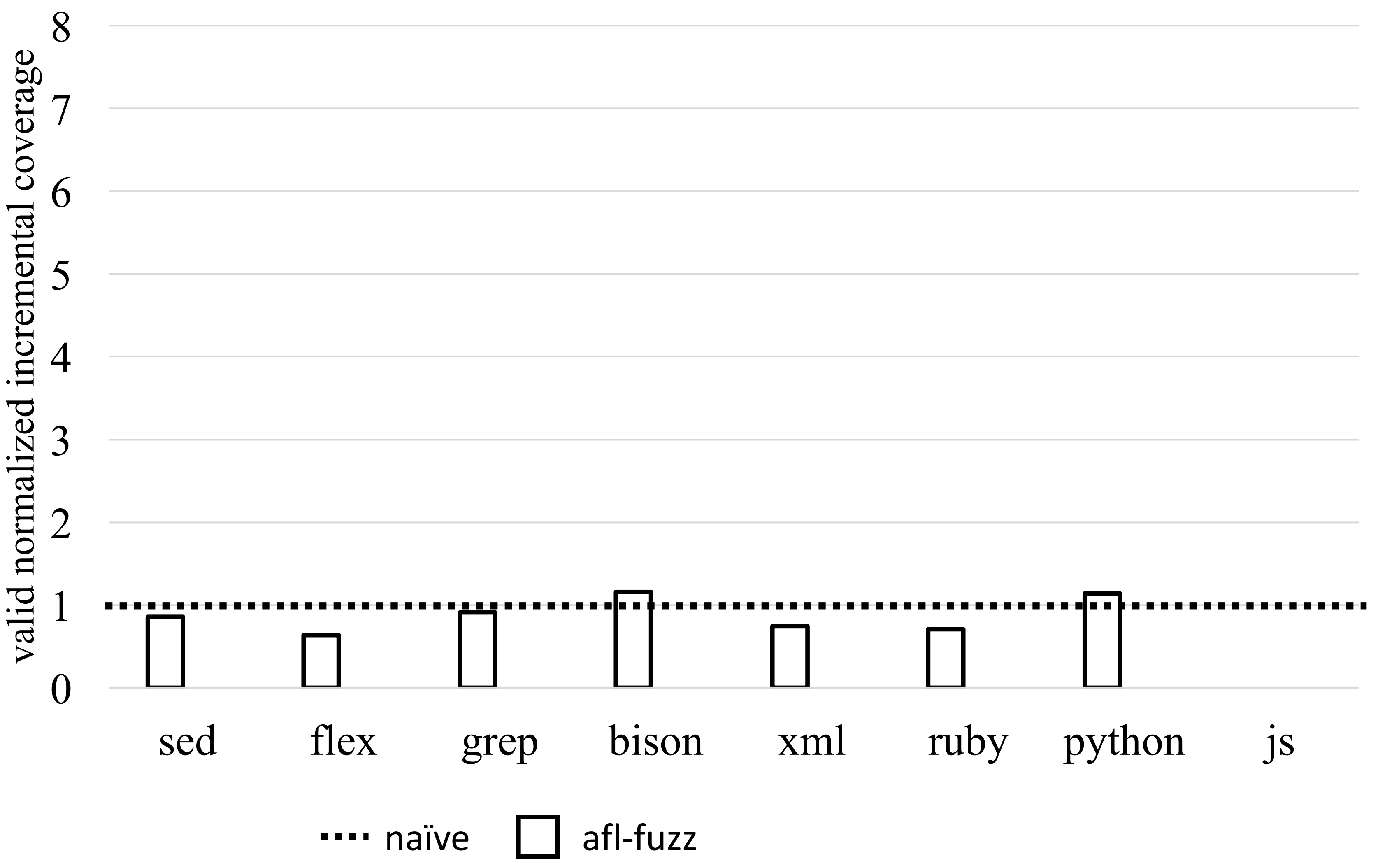
Evaluation: Fuzz Testing



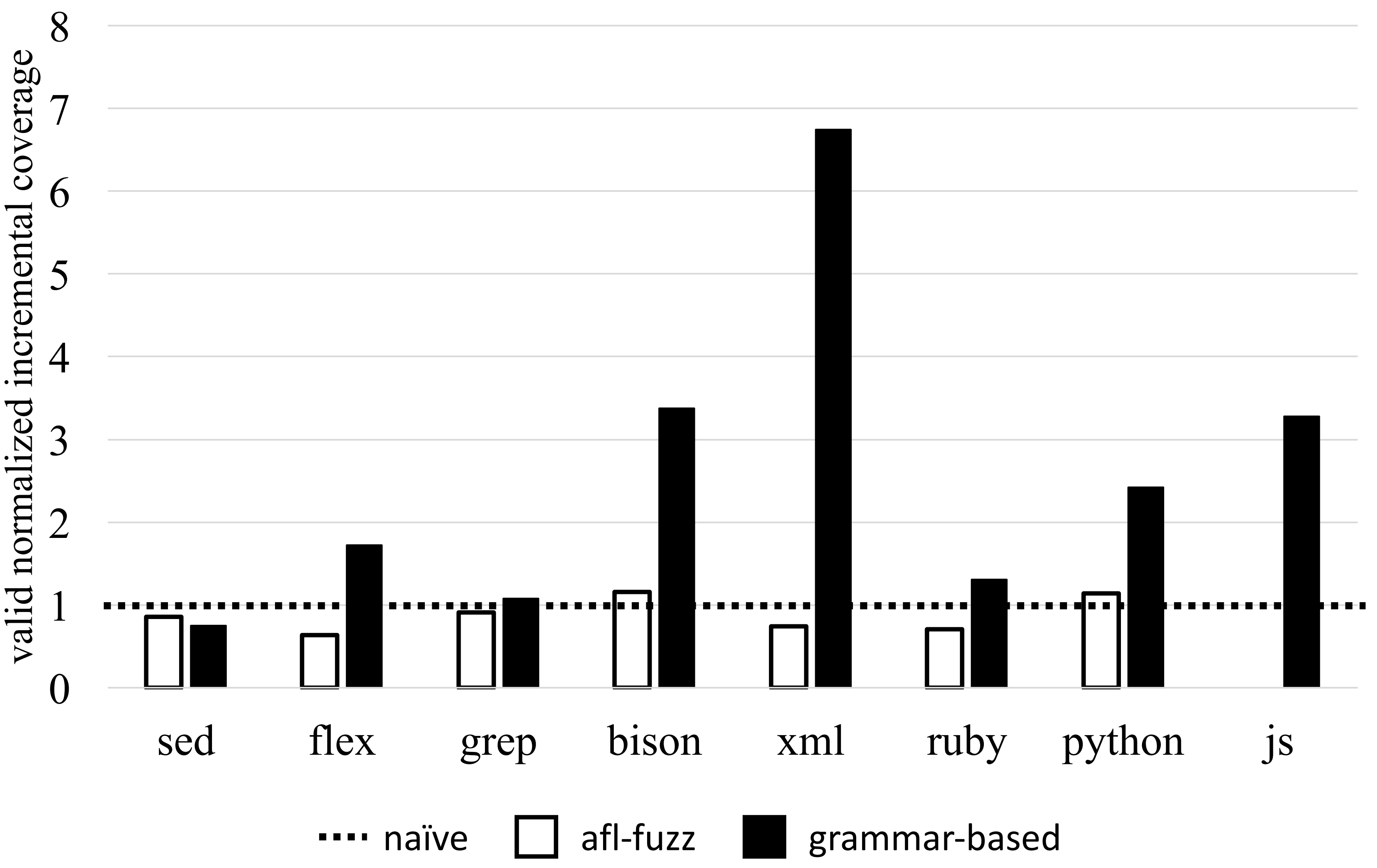
Evaluation: Fuzz Testing



Evaluation: Fuzz Testing



Evaluation: Fuzz Testing



median 2× improvement in incremental coverage

Learn program properties from input-output examples

- “Extreme” form of active learning

Related Work

- Infer program invariants from executions (Ernst 2007)
- Infer program input grammar using dynamic taint analysis (Höschele 2016)

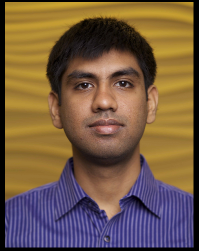
Conclusions & Related Work

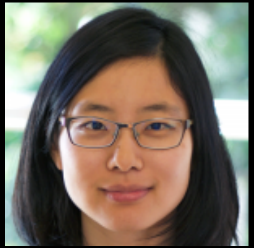
Infer…

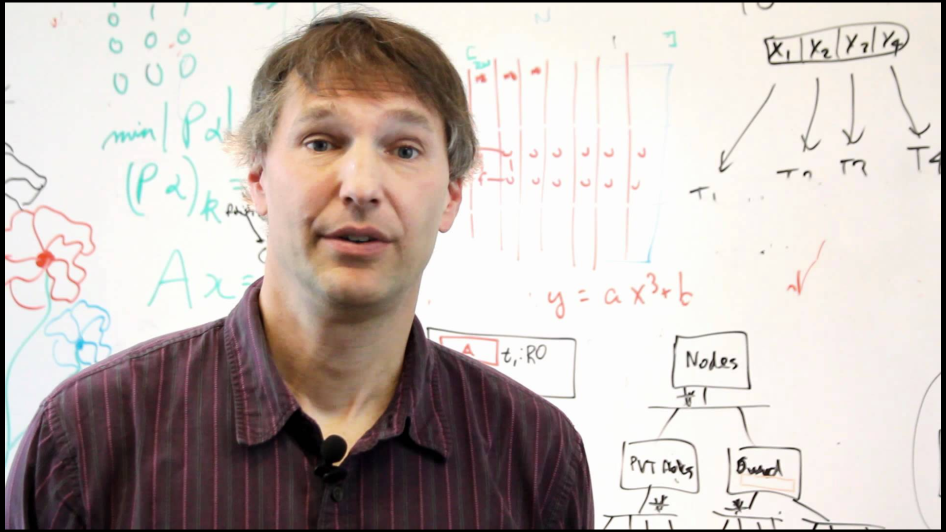
- Flow summaries (Zhu 2013)
- Input formats (Höschele 2016)
- Sources/sinks (Livshits 2009)
- Typestate specifications (Beckman 2011)
- Program invariants (Ernst 2007, Dillig 2012)
- …

Conclusions & Related Work

Inference can substantially improve
the cost-effectiveness of program analysis tools





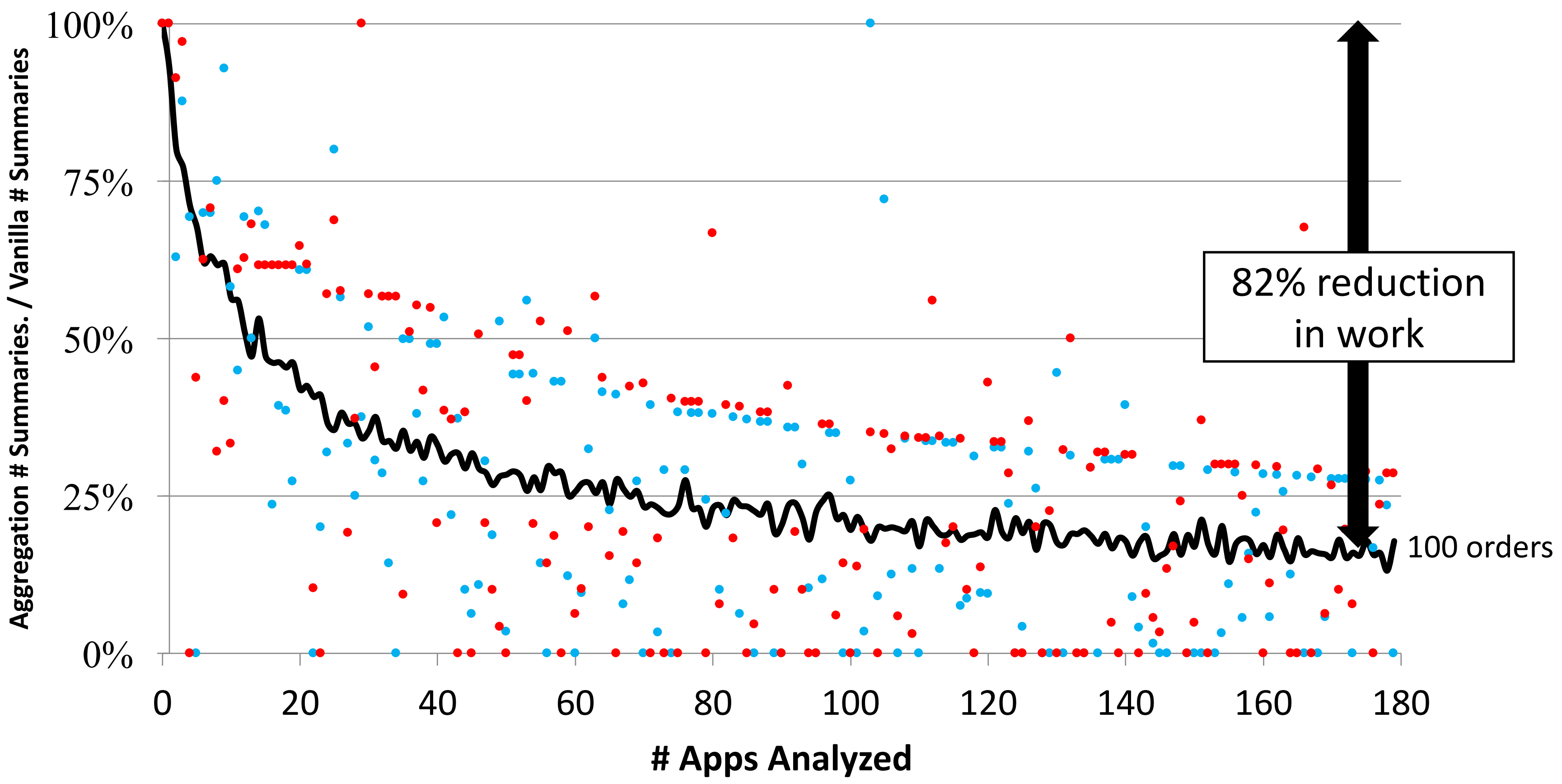

Nodes
$y = ax^3+b$

Questions?

Backup Slides

$$
\begin{array}{lcl}
\text{SrcSink} & \rightarrow & \text{SrcObj FlowsTo RefSink} \\
\text{SrcObj} & \rightarrow & \text{SrcObj FlowsTo RefRef } \overline{\text{FlowsTo}} \\
\text{SrcObj} & \rightarrow & \text{SrcRef } \overline{\text{FlowsTo}} \\
\text{FlowsTo} & \rightarrow & \text{New} \\
\text{FlowsTo} & \rightarrow & \text{FlowsTo Assign} \\
\text{FlowsTo} & \rightarrow & \text{FlowsTo Put[f] FlowsTo } \overline{\text{FlowsTo}} \text{ Get[f]}
\end{array}
$$

Aggregating Summaries over Time



Untrusted Responses (OOPSLA 2015)

Untrusted Responses (OOPSLA 2015)

Step 1: Worst-case analysis

location → Internet
SMS → Internet
device ID → SMS
…

Step 2: Infer summaries

Step 3: Analyst corrections

specification inference

Untrusted Responses (OOPSLA 2015)

Step 1: Worst-case analysis

location → Internet
SMS → Internet
device ID → SMS
...

Step 2: Infer summaries

Step 3: Analyst corrections

specification inference

```
int main(int argc, char **argv) {
  char *keys;
  size_t key      olde    eyalloc;
  bool with     ilenames;
  size_t c
  int opt,     epended;
  int prev              ecursive;
  int frea
  intmax_
  FILE *
  exit_f                 E;
  initi                  rgv);
  set_pr                 ;
  progra
  // ...
}
```

deploy app with instrumentation

Untrusted Responses (OOPSLA 2015)

Step 1: Worst-case analysis

location → Internet
SMS → Internet
device ID → SMS
…

Step 2: Infer summaries

Step 3: Analyst corrections

specification inference

```
int main(int argc, char **argv) {
  char *keys;
  size_t key      olde    eyalloc;
  bool with     ilenames;
  size_t c
  int opt,    epended;
  int prev  ot       st    ecursive;
  int frea
  intmax_
  FILE *
  exit_                      E;
  initi                     rgv);
  set_pr                     ;
  progra
  // ...
}
```

deploy app with instrumentation

inferred (reachability) summaries eliminated 92% of false positives compared to using worst-case summaries

“Interact” with Executions

“Interact” with Executions

Step 1: Worst-case analysis

location → Internet
SMS → Internet
device ID → SMS
…

Step 2: Infer summaries

Step 3: Monitor executions

“Interact” with Executions

Step 1: Worst-case analysis

location → Internet
SMS → Internet
device ID → SMS
…

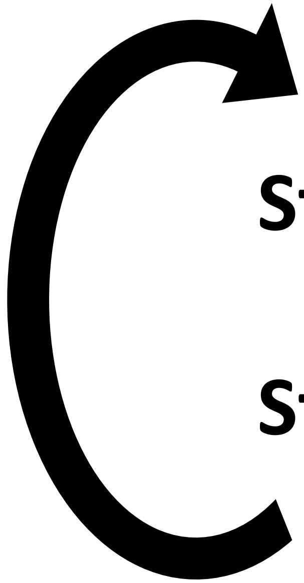
Step 2: Infer summaries

Step 3: Monitor executions

inferred 432 true positives (with 48 false positives)

Automatic Test Generation

Automatic Test Generation



Step 1: Synthesize tests

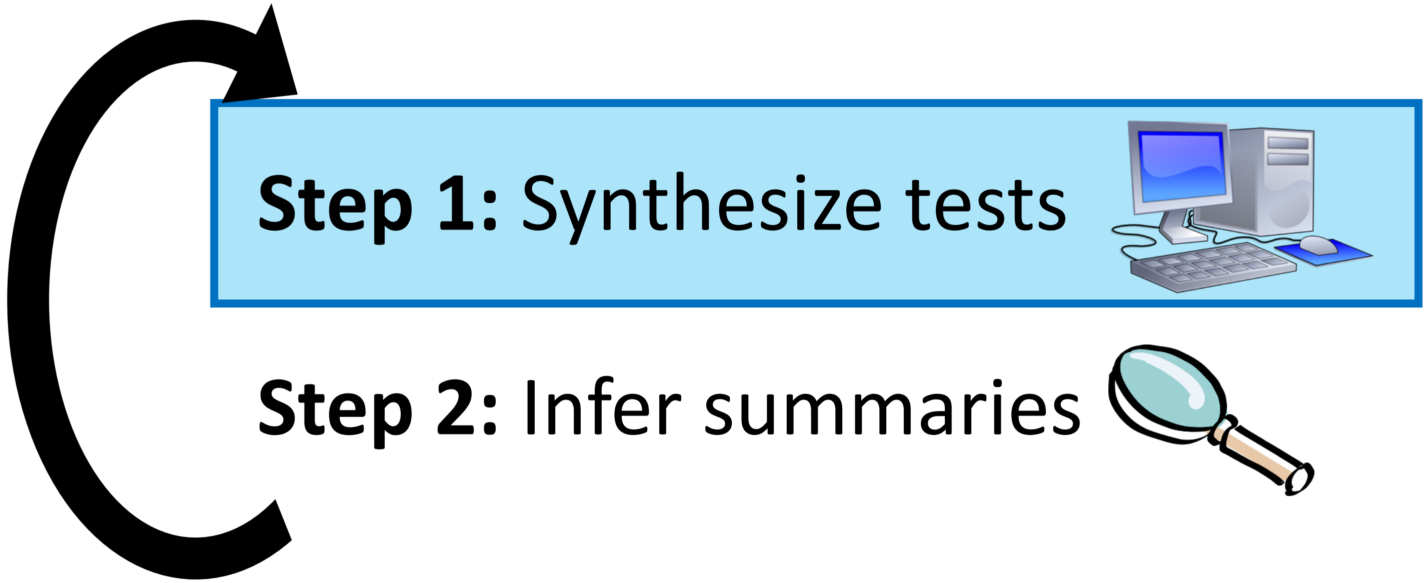
Step 2: Infer summaries

Automatic Test Generation

Step 1: Synthesize tests

Step 2: Infer summaries

Automatic Test Generation



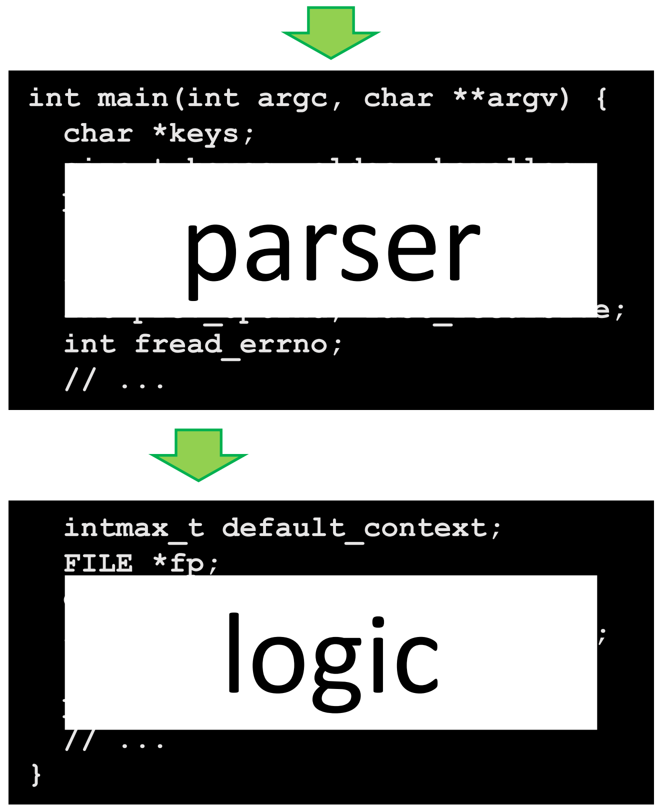
Step 1: Synthesize tests

Step 2: Infer summaries

Java Collections API summaries: 733 inferred versus 58 existing

$$A_{\text{XML}} \to (\mathtt{a} + \cdots + \mathtt{z})$$
$$A_{\text{XML}} \to \mathtt{<a>} A_{\text{XML}} \mathtt{</a>}$$
$$A_{\text{XML}} \to A_{\text{XML}}^{*}$$

`<a>hi</a>`

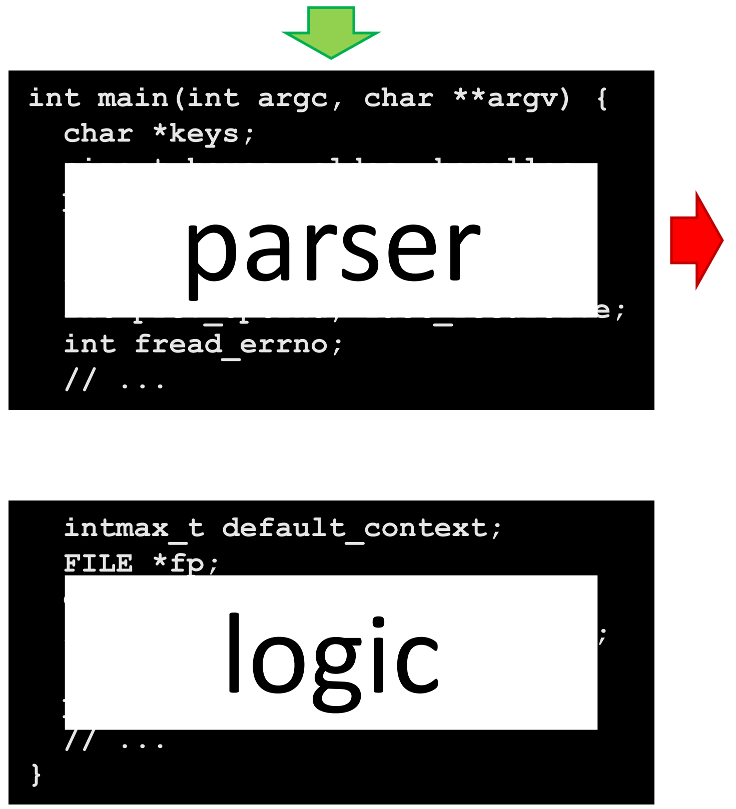


```
int main(int argc, char **argv) {
  char *keys;
  int fread_errno;
  // ...
```

```
  intmax_t default_context;
  FILE *fp;
  // ...
}
```

aa>hi</a>

$$A_{\text{XML}} \to (\mathtt{a}+\cdots+\mathtt{z})$$
$$A_{\text{XML}} \to \mathtt{<a>}A_{\text{XML}}\mathtt{</a>}$$
$$A_{\text{XML}} \to A_{\text{XML}}^{*}$$

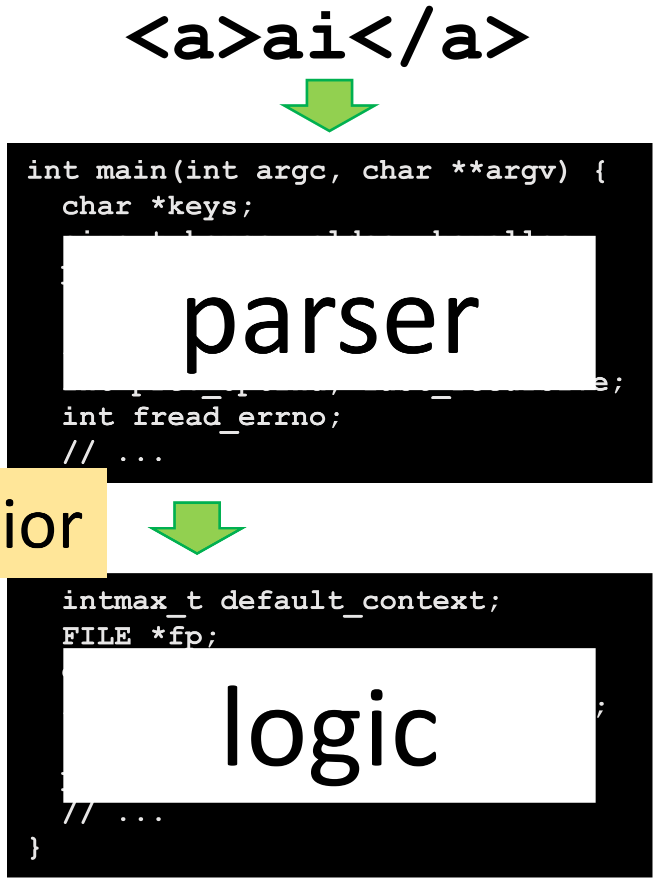


```
int main(int argc, char **argv) {
  char *keys;

  int fread_errno;
  // ...
```

```
  intmax_t default_context;
  FILE *fp;

  // ...
}
```

unmatched '>'!

$$
\begin{aligned}
A_{\mathrm{XML}} &\rightarrow (\mathtt{a}+\cdots+\mathtt{z}) \\
A_{\mathrm{XML}} &\rightarrow \mathtt{<a>}A_{\mathrm{XML}}\mathtt{</a>} \\
A_{\mathrm{XML}} &\rightarrow A_{\mathrm{XML}}^{*}
\end{aligned}
$$



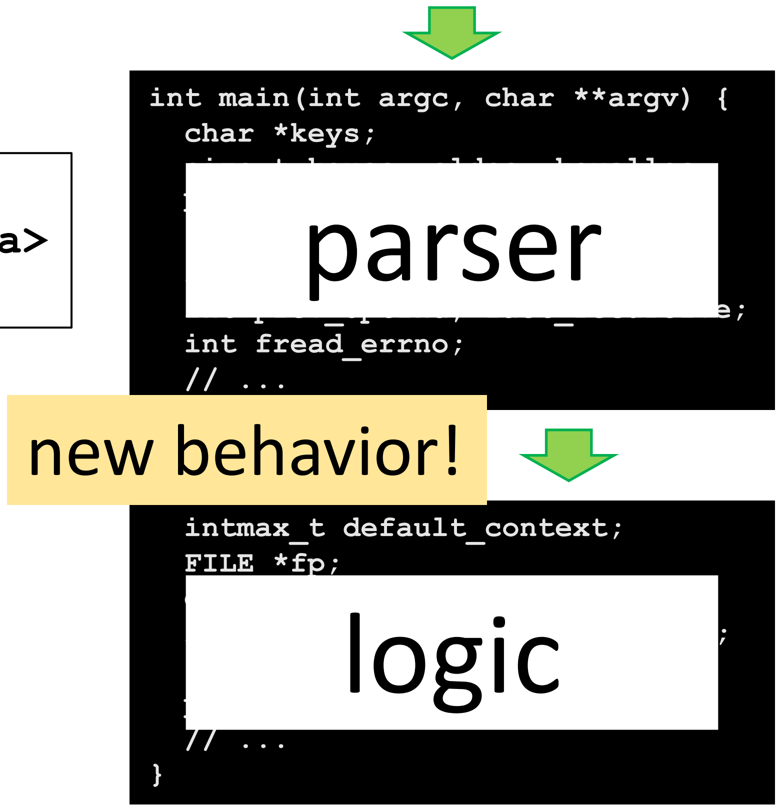
```
int main(int argc, char **argv) {
  char *keys;
  int fread_errno;
  // ...
```

```
  intmax_t default_context;
  FILE *fp;
  // ...
}
```

same behavior

`<a><a>hi</a></a>`

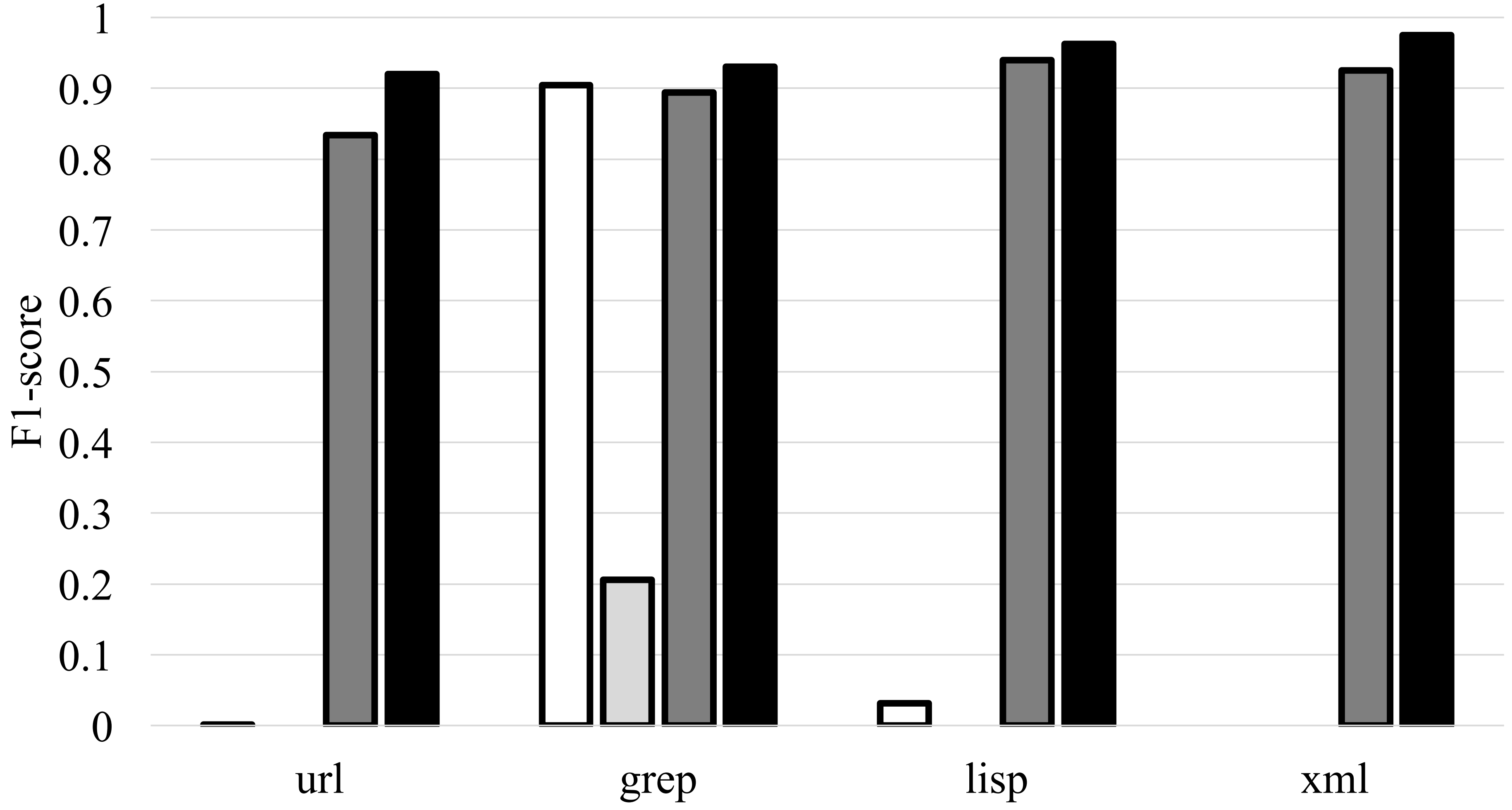
$$A_{\text{XML}} \to (\mathtt{a}+\cdots+\mathtt{z})$$
$$A_{\text{XML}} \to \mathtt{<a>}A_{\text{XML}}\mathtt{</a>}$$
$$A_{\text{XML}} \to A_{\text{XML}}^{*}$$



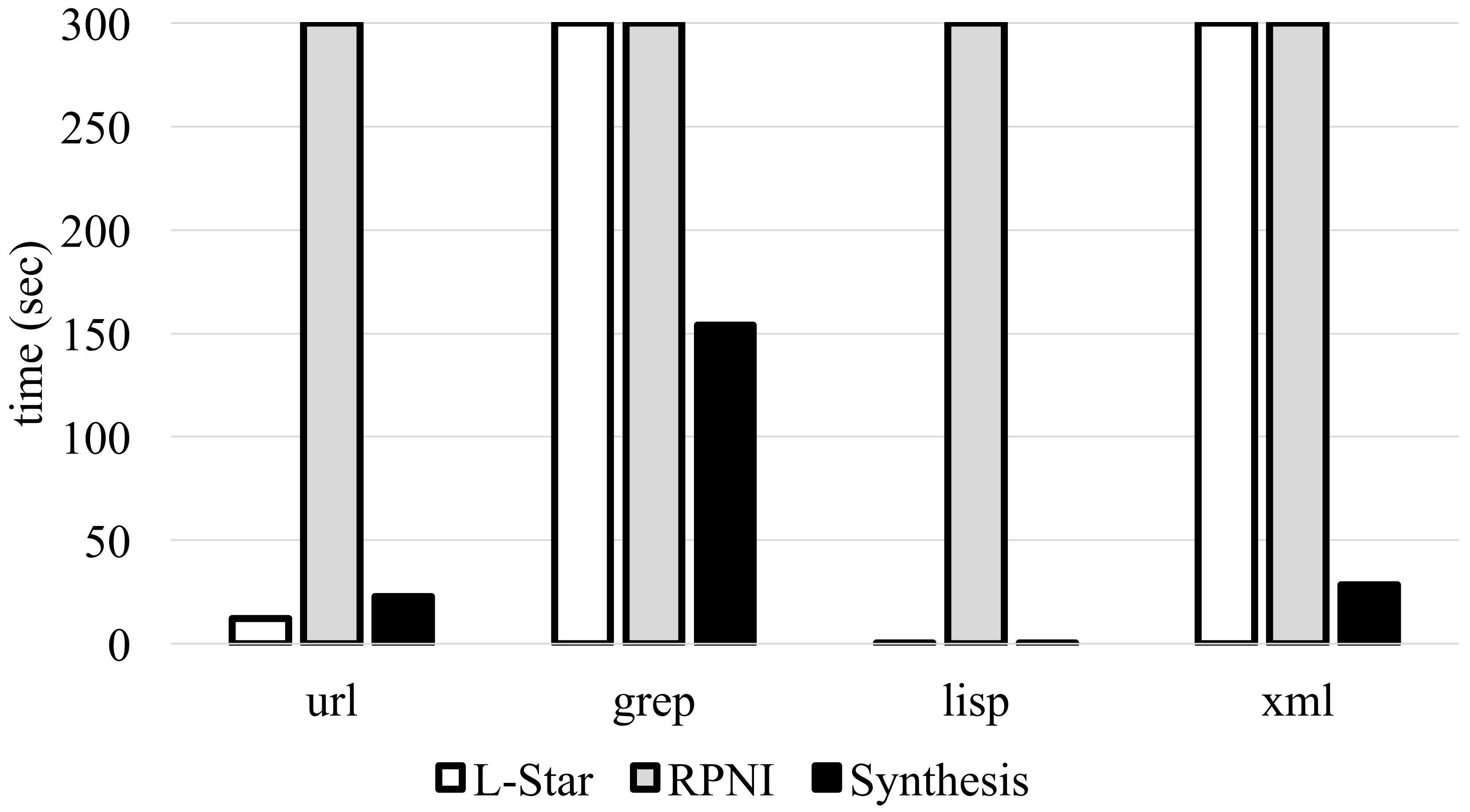
```
int main(int argc, char **argv) {
  char *keys;
  int fread_errno;
  // ...
```

```
  intmax_t default_context;
  FILE *fp;
  // ...
}
```

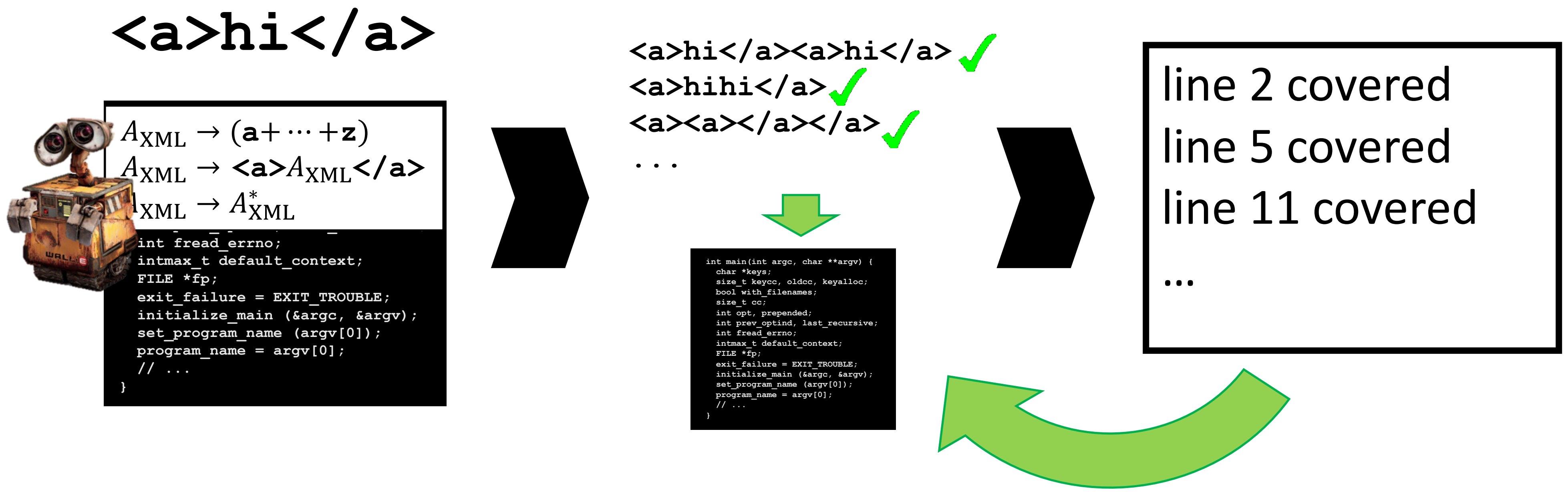
F_1 Score



Running Time



Coverage-Guided Fuzz Testing



`<a>hi</a>`

$A_{\text{XML}} \to (\texttt{a}+\cdots+\texttt{z})$
$A_{\text{XML}} \to \texttt{<a>}A_{\text{XML}}\texttt{</a>}$
$A_{\text{XML}} \to A^{*}_{\text{XML}}$

```
  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];
  // ...
}
```

```
<a>hi</a><a>hi</a> ✓
<a>hihi</a> ✓
<a><a></a></a> ✓
...
```

```
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];
  // ...
}
```

line 2 covered
line 5 covered
line 11 covered
...

40% improvement in incremental coverage on XML (preliminary)

Lexer Synthesis

`<a>hi</a>`

$$\mathcal{O}_{\text{XML}}(\alpha) = \begin{cases} 1 & \text{if } \alpha \in L_{\text{XML}} \\ 0 & \text{otherwise} \end{cases}$$

oracle & input

$$\texttt{<a>}[\texttt{a}-\texttt{z}]^{*}\texttt{</a>}$$

lexer

$$A_{\text{XML}} \rightarrow (\texttt{a} + \cdots + \texttt{z})$$
$$A_{\text{XML}} \rightarrow \texttt{<a>} A_{\text{XML}} \texttt{</a>}$$
$$A_{\text{XML}} \rightarrow A_{\text{XML}}^{*}$$

grammar

References

- I. Dillig, T. Dillig, A. Aiken. Automated error diagnosis using abductive inference. In PLDI, 2012.
- H. Zhu, T. Dillig, I. Dillig. Automated inference of library specifications for source-sink property verification. In APLAS, 2013.
- S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. L. Traon, D. Octeau, P. McDaniel. FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. In PLDI, 2014.
- T. Reps. Program analysis via graph reachability. In ILPS, 1997.
- M. Sridharan, D. Gopan, L. Shan, R. Bodik. Demand-driven points-to analysis for Java. In OOPSLA, 2005.
- B. Livshits, A. Nori, S. Rajamani, A. Banerjee, Merlin: Specification Inference for Explicit Information Flow Problems , in PLDI, 2009.
- M. Höschele, A. Zeller, Mining input grammars from dynamic taints. In ASE, 2016.
- B. P. Miller, L. Fredriksen, and B. So. An empirical study of the reliability of unix utilities. Communications of the ACM, 33(12):32–44, 1990.
- V. Ganesh, T. Leek, and M. Rinard. Taint-based directed whitebox fuzzing. In Proceedings of the 31st International Conference on Software Engineering, pages 474–484. IEEE Computer Society, 2009.
- P. Godefroid, A. Kiezun, and M. Y. Levin. Grammar-based whitebox fuzzing. In ACM Sigplan Notices, volume 43, pages 206–215. ACM, 2008.
- M. Zalewski. American fuzzy lop, 2015. http://lcamtuf.coredump.cx/afl.
- D. Angluin. Learning regular sets from queries and counterexamples. Information and computation, 75(2):87–106, 1987.
- X. Yang, Y. Chen, E. Eide, and J. Regehr. Finding and understanding bugs in c compilers. In ACM SIGPLAN Notices, volume 46, pages 283–294. ACM, 2011.
- Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. 2014.