Goal-Directed Program Analysis with Jumping

Sam Blackshear
University of Colorado Boulder

Bor-Yuh Evan Chang
University of Colorado Boulder

Manu Sridharan
Samsung Research America

Google
July 24, 2015
Lab: Program analysis in the whole bug mitigation process
Lab: **Program analysis in the whole bug mitigation process**

![Diagram showing program analysis in the bug mitigation process](image)
Lab: Program analysis in the whole bug mitigation process
Lab: **Program analysis in the whole bug mitigation process**
Lab: **Program analysis in the whole bug mitigation process**

![Diagram showing the verification process with components labeled](image)
Lab: **Program analysis in the whole bug mitigation process**

- **Verifier**
  - **Thresher: Goal-Directed Refutation Analysis**
    - [Blackshears SAS’11, Blackshear+ PLDI’13, under review]

- **Manual Triaging**

- **Programming**
  - **Program**

- **Verifier**
  - **proof of no bug**
  - **Alarm Report**
Lab: Program analysis in the whole bug mitigation process

Verifier

Thresher: Goal-Directed Refutation Analysis
[Blackshear+ SAS’11, Blackshear+ PLDI’13, under review]

Program analysis
in the whole bug mitigation process
Fissile Types: Checking Almost Everywhere Invariants

[Coughlin+ POPL’14, in prep]

Thresher: Goal-Directed Refutation Analysis

[Blackshear+ SAS’11, Blackshear+ PLDI’13, under review]
Fissile Types: Checking Almost Everywhere Invariants
[Coughlin+ POPL’14, in prep]

Intertwine type-based and separation-based analysis to verify almost-everywhere invariants
Lab: Program analysis in the whole bug mitigation process

Fissile Types: Checking Almost Everywhere Invariants
[Coughlin+ POPL’14, in prep]

Verifier

proof of no bug

Alarm Report

Thresher: Goal-Directed Refutation Analysis
[Blackshear+ SAS’11, Blackshear+ PLDI’13, under review]
Program analysis in the whole bug mitigation process

**Fissile Types:**
Checking Almost Everywhere Invariants
[Coughlin+ POPL’14, in prep]

**Thresher:** Goal-Directed Refutation Analysis
[Blackshear+ SAS’11, Blackshear+ PLDI’13, under review]
Lab: Program analysis in the whole bug mitigation process

- Fissile Types: Checking Almost Everywhere Invariants
  [Coughlin+ POPL’14, in prep]

- Thresher: Goal-Directed Refutation Analysis
  [Blackshear+ SAS’11, Blackshear+ PLDI’13, under review]

- Divva: Synthesizing Short-Circuiting Data Structure Checks
  [under review]

- Manual Triaging

- Test Input → Program → Specification → Runner → Test Output

- Verifier:
  - proof of no bug
  - Alarm Report

- Program-Mining
Program analysis in the whole bug mitigation process

Verifier

- Proof of no bug

Alarm

- Report

Program-Verifying

- Manual
- Triaging

Program

- Triaging
- Test Input
- Specification

Use static shape analysis to synthesize short-circuiting dynamic validation of data structure invariants

uninterpreted inductive separation logic predicates

Thresher: Goal-Directed Refutation Analysis

[Blackshear+ SAS’11, Blackshear+ PLDI’13, under review]

Fissile Types: Checking Almost Everywhere Invariants

[Coughlin+ POPL’14, in prep]

Divva: Synthesizing Short-Circuiting Data Structure Checks

[under review]
Program analysis in the whole bug mitigation process.

**Fissile Types:** Checking Almost Everywhere Invariants

\[\text{Coughlin}^+ \text{ POPL'14, in prep}\]

**Divva:** Synthesizing Short-Circuiting Data Structure Checks

\[\text{under review}\]

**Thresher:** Goal-Directed Refutation Analysis

\[\text{Blackshear}^+ \text{ SAS'11, Blackshear}^+ \text{ PLDI'13, under review}\]
Program analysis in the whole bug mitigation process

- **Verifier**: Synthesizing Short-Circuiting Data Structure Checks [under review]
- **Thresher**: Goal-Directed Refutation Analysis [Blackshear+ SAS’11, Blackshear+ PLDI’13, under review]

**Programing**

- **Test Input**
- **Speciﬁcation**
- **Program**
- **Github**

**Alarm Report**

- **proof of no bug**

**Manual Triaging**
Lab: Program analysis in the whole bug mitigation process

**Fissile Types:** Checking Almost Everywhere Invariants
[Coughlin+ POPL’14, in prep]

**Divva:** Synthesizing Short-Circuiting Data Structure Checks
[under review]

**Thresher:** Goal-Directed Refutation Analysis
[Blackshear+ SAS’11, Blackshear+ PLDI’13, under review]

**Github**

**Programmimg**

**Manual Triaging**

**Test Input**

**Program**

**Verifier**

**Alarm Report**

**Test Output**

**proof of no bug**

**Fixr:** Mining Bug Fixes from Commits
[in progress]
Lab: Program analysis in the whole bug mitigation process

Fissile Types: Checking Almost Everywhere Invariants
[Coughlin + POPL’14, in prep]

Verifier

Test Input

Specification

Program

Test Output

Alarm Report

proof of no bug

Program-Ming

Github

Fixr: Mining Bug Fixes from Commits
[in progress]

Runner

Jsana: Abstract Domain Combinators for Dynamic Languages
[Cox+ ECOOP’13, Cox+ SAS’14, Cox+ ESOP’15]

Divva: Synthesizing Short-Circuiting Data Structure Checks
[under review]

Thresher: Goal-Directed Refutation Analysis
[Blackshear+ SAS’11, Blackshear+ PLDI’13, under review]
Lab: Program analysis in the whole bug mitigation process

Verifier ✔  proof of no bug

Alarm  Report  ❌

Program

Manual

Triaging

Runner

Spec-
ification

Test Input

Test Output

Program-
ing

Jsana: Abstract Domain Combinators for Dynamic Languages
[Cox+ ECOOP’13, Cox+ SAS’14, Cox+ ESOP’15]

Divva: Synthesizing Short-Circuiting Data Structure Checks [under review]

Fissile Types: Checking Almost Everywhere Invariants
[Coughlin+ POPL’14, in prep]

Fissile Types

Github

Fixr: Mining Bug Fixes from Commits [in progress]

Heap with set symbols partitioning “open objects”

<table>
<thead>
<tr>
<th></th>
<th>A_1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>F_1</td>
<td>V_1</td>
<td></td>
</tr>
<tr>
<td>F_2</td>
<td>V_2</td>
<td></td>
</tr>
<tr>
<td>F_3</td>
<td>A_2</td>
<td></td>
</tr>
</tbody>
</table>

Fixr

A_1

V_1

V_2

A_2

X

Heap with set symbols partitioning “open objects”

separation logic with open object predicates and desynchronized separation
Program analysis in the whole bug mitigation process:

- **Verifier**
  - **Proof of no bug**
  - **Alarm Report**

- **Runner**
  - **Test Input** → **Test Output**

- **Program**: Test Input → Specification → Program → Verification

- **Programming**
  - **Github**

- **Manual Triaging**

- **Manual Fixing**
  - **Fixr**: Mining Bug Fixes from Commits [in progress]

- **Thresher**: Goal-Directed Refutation Analysis
  - [Blackshear+ SAS’11, Blackshear+ PLDI’13, under review]

- **Divva**: Synthesizing Short-Circuiting Data Structure Checks
  - [under review]

- **Jsana**: Abstract Domain Combinators for Dynamic Languages
  - [Cox+ ECOOP’13, Cox+ SAS’14, Cox+ ESOP’15]

- **Fissile Types**: Checking Almost Everywhere Invariants
  - [Coughlin+ POPL’14, in prep]
Lab: **Program analysis in the whole bug mitigation process**

**Fissile Types:**
- Checking Almost Everywhere Invariants
  
  \[\text{Coughlin+ POPL'14, in prep}\]

**Thresher:** Goal-Directed Refutation Analysis

\[\text{Blackshear+ SAS’11, Blackshear+ PLDI’13, under review}\]

**Divva:** Synthesizing Short-Circuiting Data Structure Checks

\[\text{under review}\]

**Gsana:** Abstract Domain Combinators for Dynamic Languages

\[\text{Cox+ ECOOP’13, Cox+ SAS’14, Cox+ ESOP’15}\]

**Fixr:** Mining Bug Fixes from Commits

\[\text{in progress}\]

**This Talk**
Goal-Directed Program Analysis with Jumping

Sam Blackshear
University of Colorado Boulder

Bor-Yuh Evan Chang
University of Colorado Boulder

Manu Sridharan
Samsung Research America

Google
July 24, 2015
Roughly, 3% of all commits fix NullPointerExceptions.
Callback-oriented programming
Callback-oriented programming

App

Android Framework
Callback-oriented programming

callbacks (e.g., Activity.onCreate)
Callback-oriented programming

callbacks (e.g., Activity.onCreate)
Callback-oriented programming

Android components have an ordered, event-driven lifecycle

Activity

onCreate

onResume

onClick

onPause

onDestroy

callbacks (e.g., Activity.onCreate)
Callback-oriented programming

Android components have an ordered, event-driven lifecycle.

callbacks (e.g., Activity.onCreate)

mHostDb = null;
mService = null;
Callback-oriented programming

Android components have an ordered, event-driven lifecycle

callsbacks (e.g., Activity.onCreate)

Activity

onCreate

onResume

onClick

onPause

onDestroy

mHostDb = null;
mService = null;

But, lifecycles of different components and other callbacks can interleave ...
Challenge: Verifying safety (of dereferences) depends on callback interleaving

Android components have an ordered, event-driven lifecycle

callbacks (e.g., Activity.onCreate)

onResume

onClick

onPause

onDestroy

mHostDb = null;
mService = null;

But, lifecycles of different components and other callbacks can interleave ...
Callback-oriented programming

onCreate

onResume

onClick

onPause

onDestroy
Callback-oriented programming

- onCreate
- onResume
- onClick
- onPause
- onDestroy
Callback-oriented programming

Heap

onCreate → onResume

onResume → onClick

onClick → onPause

onPause → onDestroy

onDestroy
Callback-oriented programming

... and operates over a shared, global heap
Challenge: Verifying safety (of dereferences) depends **ordering of heap writes**
But it shouldn’t be so hard …

onCreate

onResume

onPause

onDestroy
But it shouldn’t be so hard ...

```java
mHostDb.s();
```
But it shouldn’t be so hard ...

```java
mHostDb.s();
```

```java
safe?
```

```java
onCreate
```

```java
onResume
```

```java
onPause
```

```java
onDestroy
```
But it shouldn’t be so hard ...

Idea: Safety of a particular dereference should not require reasoning about all callback interleavings.

```
safe?
mHostDb.s();
```

```
onCreate

onChange

onResume

onPause

onDestroy
```
But it shouldn’t be so hard ...

```
import android.view.View;

mHostDb.s();
```

Idea: Safety of a particular dereference should not

A “smart” goal-directed analysis could consider relevant callback orderings without considering all of them
Goal-directed program analysis

safe?

mHostDb.s();
Goal-directed program analysis

Given a program configuration **goal**, derive a **contradiction** w.r.t. its reachability

```c
mHostDb.s();
```
Given a program configuration goal, derive a contradiction w.r.t. its reachability.
Goal-directed program analysis

Given a program configuration \textbf{goal}, derive a \textbf{contradiction} w.r.t. its reachability

\begin{align*}
\text{safe?} \\
\text{mHostDb.s();} \\
\text{mHostDb == null}
\end{align*}
Given a program configuration **goal**, derive a **contradiction** w.r.t. its reachability.

\[
\text{(this} \leftrightarrow \hat{t} \ast \hat{t} \cdot \text{mHostDb} \leftrightarrow \hat{a} \ast \text{true}) \land \hat{a} = \text{null}
\]
Goal-directed program analysis

Given a program configuration goal, derive a contradiction w.r.t. its reachability

\[(\texttt{this} \mapsto \hat{t} \ast \hat{t} \cdot \texttt{mHostDb} \mapsto \hat{\texttt{a}} \ast \texttt{true}) \land \hat{\texttt{a}} = \texttt{null}\]

A precise backwards abstract interpretation (with separation logic constraints) to refute error conditions [PLDI’13]
Two dereferences: one safe and one buggy
Two dereferences: one safe and one buggy

```java
void onClick(...) {
    mHostDb.s(mService.g());
}
```
Two dereferences: one safe and one buggy

```
void onClick(...) {
    mHostDb.s(mService.g());
}
```
Two dereferences: one safe and one buggy

```java
void onClick(...) {
    mHostDb.s(mService.g());
}

void onDestroy(...) {
    mHostDb = null;
    mService = null;
}
```
Two dereferences: one safe and one buggy

```java
void onClick(...) {
    mHostDb.s(mService.g());
}
```

```java
void onDestroy(...) {
    mHostDb = null;
    mService = null;
}
```
Two dereferences: one safe and one buggy

void **onCreate**(()) {
    bindService(…, new ServiceConn {
        void **onConnected**(nonnull Service s) {
            mService = s;
        }
    });
    mHostDb = new Db();
}

void **onClick**(…) {
    mHostDb.s(mService.g());
}

void **onDestroy**(…) {
    mHostDb = null;
    mService = null;
}
Two dereferences: one safe and one buggy

```java
void onCreate() {
    bindService(..., new ServiceConn {
        void onConnected(@Nonnull Service s) {
            mService = s;
        }
    });
    mHostDb = new Db();
}

void onClick(...) {
    mHostDb.s(mService.g());
}

void onDestroy(...) {
    mHostDb = null;
    mService = null;
}
```

- **Safe Dereference:**
  - **onCreate**
  - **onClick**

- **Buggy Dereference:**
  - **onDestroy**
void onCreate() {
    bindService(…, new ServiceConn {
        void onConnected(@Nonnull Service s) {
            mService = s;
        }
    });
    mHostDb = new Db();
}

void onClick(…) {
    mHostDb.s(mService.g());
}

void onDestroy(…) {
    mHostDb = null;
    mService = null;
}
Two dereferences: one safe and one buggy

```java
void onCreate() {
    bindService(...) new ServiceConn {
        void onConnected(@Nonnull Service s) {
            mService = s;
        }
    });
    mHostDb = new Db();
}

void onClick(...) {
    mHostDb.s(mService.g());
}

void onDestroy(...) {
    mHostDb = null;
    mService = null;
}
```

Need to consider **some but not all** callback ordering constraints.
Idea: Jumping to relevant callbacks

- onCreate
- onResume
- onClick
- onPause
- onDestroy
Idea: Jumping to relevant callbacks

- onCreate
- onResume
- onClick
- onPause
- onDestroy

- onCreate
- onResume
- onClick
- onPause
- onDestroy

- onCreate
- onResume
- onClick
- onPause
- onDestroy
Idea: Jumping to relevant callbacks

onCreate → onResume → onClick → onPause → onDestroy

safe?

onClick → onCreate → onResume → onDestroy

onClick → onResume → onPause → onDestroy

onClick → onDestroy
Idea: **Jumping to relevant callbacks**

Find **data relevant callbacks**
Idea: Jumping to relevant callbacks

Find data relevant callbacks
Idea: Jumping to relevant callbacks

Find data relevant callbacks
Idea: Jumping to relevant callbacks

Find data relevant callbacks
Idea: **Jumping to relevant callbacks**

- **onCreate**
- **onResume**
- **onResume**
- **onClick**
- **onPause**
- **onPause**
- **onDestroy**
- **onDestroy**

**safe?**

**Find data relevant callbacks**

**Filter using control-feasibility**
Idea: **Jumping to relevant callbacks**

Find **data relevant** callbacks

Filter using **control-feasibility**
Idea: **Jumping to relevant callbacks**

Find **data relevant** callbacks

Filter using **control-feasibility**
Idea: Jumping to relevant callbacks

Find data relevant callbacks

Filter using control-feasibility
Contributions

backwards analysis "jumps"
Contributions

Framework for jumping analyses

1. \( l_1 \xrightarrow{[c]} l_2 \)  Framework for jumping analyses
Contributions

1. Framework for jumping analyses

2. Applied to Android events

backwards analysis "jumps"
A program model for a jumping analysis

\[ l_1 \[c\] \rightarrow l_2 \]
A program model for a jumping analysis

Program

\[ l_1 \xrightarrow{c} l_2 \]
A program model for a jumping analysis

Unstructured control-flow: atomic commands connected by pre/post labels
A program model for a jumping analysis

Unstructured control-flow: atomic commands connected by pre/post labels

Represent conditionals with `assume`, loops with `assume` and a back edge.
A program model for a jumping analysis

Unstructured control-flow: atomic commands connected by pre/post labels

Represent conditionals with assume, loops with assume and a back edge.

Analysis parameters
A program model for a jumping analysis

Unstructured control-flow: atomic commands connected by pre/post labels

Represent conditionals with \texttt{assume}, loops with \texttt{assume} and a back edge.

Analysis parameters

\[
\{Q'\} \ c \ \{Q\}
\]

Backward transfer functions
A program model for a jumping analysis

Unstructured control-flow: atomic commands connected by pre/post labels

Represent conditionals with `assume`, loops with `assume` and a back edge.

Analysis parameters

\[ \{Q'\} \ c \ \{Q\} \]

- Backward transfer functions
- Current location

\[ \ell_{\text{cur}} \]

\[ \ell_1 \rightarrow [c] \rightarrow \ell_2 \]
A program model for a jumping analysis

Unstructured control-flow: atomic commands connected by pre/post labels

Represent conditionals with assume, loops with assume and a back edge.

Analysis parameters

\{ Q' \} \ c \ \{ Q \}

Backward transfer functions

\l_{\text{cur}}

Current location

Q_{\text{cur}}

Current query
Data-relevance identifies writes

\[(Q, \ell_{\text{cur}})\]
Data-relevance identifies writes to the query $Q$.

Identify locations that can write to the query $Q$. 

$(Q, \ell_{\text{cur}})$
Data-relevance identifies writes

\[(Q, \ell_{\text{cur}})\]

Identify locations that can write to the query \(Q\).

Computed using pre-pass points-to analysis, types, field-based, ...
Data-relevance identifies writes

\[
(Q, \ell_{\text{cur}})
\]

Identify locations that can write to the query \( Q \).

Computed using pre-pass points-to analysis, types, field-based, ...

Classic idea: Following data dependencies yields a **sparse** analysis (but, here, flow-insensitive)
Control-feasibility selectively recovers flow-sensitivity

\[ (Q, \ell_{\text{cur}}) \]

Data-relevance

\[ \ell_i \quad \ell_j \quad \ell_k \quad \ldots \]
Control-feasibility selectively recovers flow-sensitivity

\((Q, \ell_{\text{cur}})\)

Data-relevance

\(\ell_i \quad \ell_j \quad \ell_k \quad \ldots\)

Control-feasibility

\(\ell_i \quad \ell_j \quad \ldots\)
Control-feasibility selectively recovers flow-sensitivity.

\[ (Q, \ell_{\text{cur}}) \]

Data-relevance

\[ \ell_i \quad \ell_j \quad \ell_k \quad \ldots \]

Control-feasibility

\[ \ell_i \quad \ell_j \quad \ldots \]

Filter the set of data-relevant locations using control flow and the current program point.

1 \[ l_1 \quad [c] \rightarrow l_2 \]
Control-feasibility selectively recovers flow-sensitivity

Filter the set of data-relevant locations using control flow and the current program point.
Control-feasibility selectively recovers flow-sensitivity

\[(Q, \ell_{\text{cur}})\]

Data-relevance

\[\ell_i \quad \ell_j \quad \ell_k \quad \ldots\]

Control-feasibility

\[\ell_i \quad \ell_j \quad \ldots\]

Filter the set of data-relevant locations using control flow and the current program point

Not backward-reachable from current location

\[\ell_1 \rightarrow [c] \rightarrow \ell_2 \]

Diagram showing backward reachability:

- \(\ell_1\)
- \(\ell_2\)
- \(\ell_3\)
- \(\ell_{\text{cur}}\)
- \(\ell_4\) (not backward-reachable)
Control-feasibility \textit{selectively} recovers flow-sensitivity.

\[(Q, \ell_{\text{cur}})\]

\[\ell_i, \ell_j, \ell_k, \ldots\]

\textbf{Data-relevance}

\textbf{Control-feasibility}

\[\ell_i, \ell_j, \ldots\]

\textbf{Must visit another relevant location first.}

\textbf{Filter the set of data-relevant locations using control flow and the current program point}

\textbf{Not backward-reachable from current location}

\[l_1 \quad [c] \rightarrow l_2 \]

\[l_1, l_2, l_3, l_{\text{cur}}, l_4\]
Jumping enables sparse, selective, on-the-fly control-flow abstraction
Jumping enables sparse, selective, on-the-fly control-flow abstraction

\[ \ell_1 \xrightarrow{[c]} \ell_2 \]

\( \ell_{\text{cur}} \) Current location

\( Q_{\text{cur}} \) Current query
Jumping enables sparse, selective, on-the-fly control-flow abstraction.

\[ \ell_1 \xrightarrow{[c]} \ell_2 \]

- \( \ell \)
  - Next locations
  - Transition relation

- \( \ell_{\text{cur}} \)
  - Current location

- \( Q_{\text{cur}} \)
  - Current query
Jumping enables sparse, selective, on-the-fly control-flow abstraction.
Jumping enables sparse, selective, on-the-fly control-flow abstraction.
Jumping enables sparse, selective, on-the-fly control-flow abstraction

<table>
<thead>
<tr>
<th>Precision</th>
<th>-insensitive</th>
<th>-sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>context</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- \( l_1 \) \[ c \] \( l_2 \)
- \( \ell_{\text{cur}} \)
- \( Q_{\text{cur}} \)
- \( \ell \)
- \( \text{Data-relevance} \)
- \( \text{Control-feasibility} \)

Next locations
Jumping enables sparse, selective, on-the-fly control-flow abstraction.

<table>
<thead>
<tr>
<th>Precision</th>
<th>-insensitive</th>
<th>-sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>context</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Current location \( l_1 \rightarrow [c] \rightarrow l_2 \)

Next locations

Control-feasibility

Data-relevance

Current location \( l_{cur} \)

Current query \( Q_{cur} \)

no filter
Jumping enables sparse, selective, on-the-fly control-flow abstraction.

<table>
<thead>
<tr>
<th>Precision</th>
<th>-Insensitive</th>
<th>-Sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Context</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Next locations

Control-feasibility

Data-relevance

\[ l_1 \xrightarrow{[c]} l_2 \]

<table>
<thead>
<tr>
<th>( l_\text{cur} )</th>
<th>( Q_\text{cur} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current location</td>
<td>Current query</td>
</tr>
</tbody>
</table>
Jumping enables sparse, selective, on-the-fly control-flow abstraction.

<table>
<thead>
<tr>
<th>Precision</th>
<th>-insensitive</th>
<th>-sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow</td>
<td></td>
<td></td>
</tr>
<tr>
<td>path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>context</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Jumping enables sparse, selective, on-the-fly control-flow abstraction.

<table>
<thead>
<tr>
<th>Precision</th>
<th>-Insensitive</th>
<th>-Sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>context</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Current location: $\ell_{\text{cur}}$

Current query: $Q_{\text{cur}}$

Control-feasibility

Data-relevance

$\ell_1 \xrightarrow{[c]} \ell_2$

Next locations

No filter

All locations
Jumping enables sparse, selective, on-the-fly control-flow abstraction.
Jumping enables sparse, selective, on-the-fly control-flow abstraction.

<table>
<thead>
<tr>
<th>Precision</th>
<th>-insensitive</th>
<th>-sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>context</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

only \(\text{preds}(\ell_{\text{cur}})\)

all locations

\(\ell_{\text{cur}}\)

\(Q_{\text{cur}}\)

Current location

Current query

Data-relevance

Control-feasibility

Next locations

\(\ell_1 \rightarrow [c] \rightarrow \ell_2\)
Jumping enables sparse, selective, on-the-fly control-flow abstraction.

- **Precision**
  - -insensitive
    - flow
    - path
    - context
  - -sensitive

- **Current location**
- **Current query**
- **Next locations**

- only_preds($\ell_{cur}$)
- all locations

- **Precision**
  - -insensitive
    - flow
    - path
    - context
  - -sensitive

- **Control-feasibility**
- **Data-relevance**

- $\ell_1 \xrightarrow{[c]} \ell_2$
Jumping enables sparse, selective, on-the-fly control-flow abstraction.

<table>
<thead>
<tr>
<th>Precision</th>
<th>-insensitive</th>
<th>-sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>path</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>context</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>

only feasible-preds($\ell_{\text{cur}}, Q_{\text{cur}}$)

all locations
Jumping enables sparse, selective, on-the-fly control-flow abstraction

<table>
<thead>
<tr>
<th>Precision</th>
<th>-insensitive</th>
<th>-sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>path</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>context</td>
<td>✔</td>
<td></td>
</tr>
</tbody>
</table>

only feasible-preds$(\ell_{\text{cur}}, Q_{\text{cur}})$

all locations
Jumping enables sparse, selective, on-the-fly control-flow abstraction

<table>
<thead>
<tr>
<th>Precision</th>
<th>-insensitive</th>
<th>-sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>path</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>context</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

only feasible-preds($\ell_{\text{cur}}$, $Q_{\text{cur}}$)

all locations
Jumping enables sparse, selective, on-the-fly control-flow abstraction.

<table>
<thead>
<tr>
<th>Precision</th>
<th>-insensitive</th>
<th>-sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>path</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>context</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

-only feasible-preds(ℓ_{cur}, Q_{cur})-

all locations, except assumes
Jumping enables sparse, selective, on-the-fly control-flow abstraction

<table>
<thead>
<tr>
<th>Precision</th>
<th>-insensitive</th>
<th>-sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>context</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

only feasible-preds$(\ell_{\text{cur}}, Q_{\text{cur}})$

all locations, except assumes
Jumping enables sparse, selective, on-the-fly control-flow abstraction

<table>
<thead>
<tr>
<th>Precision</th>
<th>-insensitive</th>
<th>-sensitive</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>path</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>context</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

only feasible-preds($\ell_{\text{cur}}, Q_{\text{cur}}$)

all locations, except assumes
Jumping enables sparse, selective, on-the-fly control-flow abstraction.

Only feasible-preds($\ell_{\text{cur}}, Q_{\text{cur}}$)

All locations

Data-relevance

Current location

Current query

Control-feasibility

Next locations
Jumping enables sparse, selective, on-the-fly control-flow abstraction.

\[ \text{only feasible-preds}(\ell_{\text{cur}}, Q_{\text{cur}}) \]

\[ \text{all mods}(Q_{\text{cur}}) \]
Jumping enables sparse, selective, on-the-fly control-flow abstraction.

Be **sparse** by using data relevance with desired control-flow abstraction.

Only feasible-preds($\ell_{\text{cur}}, Q_{\text{cur}}$)

All mods($Q_{\text{cur}}$)

**Data-relevance**

**Control-feasibility**

- $\ell_{\text{cur}}$
- $Q_{\text{cur}}$

Current location

Current query
Jumping enables sparse, selective, on-the-fly control-flow abstraction.

Be **sparse** by using data relevance with desired control-flow abstraction.

- only feasible-preds($\ell_{\text{cur}}, Q_{\text{cur}}$)
- all mods($Q_{\text{cur}}$)

Be **selective** by varying the relevance relation at each an analysis step.
Contributions

1. Framework for jumping analyses

2. Applied to Android events

backwards analysis “jumps”
Contributions

backwards analysis
“jumps”

1  \[ l_1 \xrightarrow{c} l_2 \]  Framework for jumping analyses

2  Applied to Android events
Formalizing lifecycle graphs

- `onCreate`
- `onResume`
- `onPause`
- `onDestroy`
- `onClick`
Formalizing lifecycle graphs

“Lifecycle automata” accepts the concrete trace of callbacks projected onto its transitions
Formalizing lifecycle graphs

“Lifecycle automata” accepts the concrete trace of callbacks projected onto its transitions

Further scalability challenge: Lifecycle spec per class but analysis applies per object
A jumping policy for Android analysis
Within an event-callback (intra-event), follow predecessor transitions.

No jumping, path/context-sensitive.
Within an event-callback (intra-event), follow predecessor transitions

no jumping, path/context-sensitive

Between event-callbacks (inter-event), jump using lifecycle graphs for control-feasibility filtering
Hypothesis: Jumping is an effective approach to path-sensitive, inter-event analysis
Setup: Proving dereferences safe
Setup: Proving dereferences safe

10 open source Android apps
3,000 to 55,000 lines of code
10 to 100 components
120 to 1,320 callbacks
Setup: Proving dereferences safe

10 open source Android apps
3,000 to 55,000 lines of code
10 to 100 components
120 to 1,320 callbacks

Event product graph would have $10^{10}$ to $10^{111}$ nodes (with one instance per class)
Setup: Proving dereferences safe

10 open source Android apps
3,000 to 55,000 lines of code
10 to 100 components
120 to 1,320 callbacks
Setup: Proving dereferences safe

- 10 open source Android apps
- 3,000 to 55,000 lines of code
- 10 to 100 components
- 120 to 1,320 callbacks

Compared 3 analyses

- **Nit**: flow-insensitive
- **Thresher**: no jumping
- **Hopper**: jumping
Is jumping effective?

<table>
<thead>
<tr>
<th>Project</th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
</tr>
</tbody>
</table>
Is jumping effective?

<table>
<thead>
<tr>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
</tr>
<tr>
<td>Total</td>
<td>266</td>
<td>62562</td>
<td>41071</td>
<td>9968</td>
</tr>
</tbody>
</table>

Many derefs
Is jumping effective?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
</tr>
</tbody>
</table>
Is jumping effective?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>266</td>
<td>62562</td>
<td>41071</td>
<td>9968</td>
<td>5412 (54)</td>
</tr>
</tbody>
</table>
Is jumping effective?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
</tr>
</tbody>
</table>
Is jumping effective?

### Unproven Derefs

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179 (72)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181 (51)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518 (143)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954 (477)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601 (290)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986 (625)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316 (74)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698 (330)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104 (1988)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431 (1362)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968 (5412)</strong></td>
</tr>
</tbody>
</table>

Thr and Hop: 10 second budget per deref
Is jumping effective?

## unproven derefs

<table>
<thead>
<tr>
<th>Project</th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr Hop</th>
<th>(Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
</tr>
</tbody>
</table>
Is jumping effective?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr Hop</th>
<th>(Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290 (52)</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625 (37)</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74 (77)</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330 (53)</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
</tr>
</tbody>
</table>

Thresher has 74% fewer than Nit
Is jumping effective?

<table>
<thead>
<tr>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
</tr>
</tbody>
</table>

Hopper has 54% fewer than Thresher
Is jumping effective?

<table>
<thead>
<tr>
<th>Repository</th>
<th>KLOC</th>
<th>Derefs</th>
<th>Nits</th>
<th>Thr Hop (Impr %)</th>
<th>% Proven</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179 (60)</td>
<td>92</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181 (72)</td>
<td>94</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518 (72)</td>
<td>93</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954 (50)</td>
<td>90</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601 (52)</td>
<td>92</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986 (37)</td>
<td>92</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316 (77)</td>
<td>97</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698 (53)</td>
<td>94</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104 (36)</td>
<td>90</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431 (44)</td>
<td>91</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
</tr>
</tbody>
</table>
Is jumping effective?

<table>
<thead>
<tr>
<th>Project</th>
<th>KLOC</th>
<th>Derf</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop (Impr %)</th>
<th>% Proven</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72 (60)</td>
<td>92</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51 (72)</td>
<td>94</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143 (72)</td>
<td>93</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477 (50)</td>
<td>90</td>
</tr>
<tr>
<td>github</td>
<td></td>
<td>92</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>seriesguide</td>
<td></td>
<td>92</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>connectbot</td>
<td></td>
<td>97</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>textsecure</td>
<td></td>
<td>94</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988 (36)</td>
<td>90</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362 (44)</td>
<td>91</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</strong></td>
<td><strong>9968</strong></td>
<td><strong>5412 (54)</strong></td>
<td><strong>92</strong></td>
</tr>
</tbody>
</table>

Compare with state-of-the-art NPE checking work that reports 84-91% proven on normal Java programs!
Triaging alarms to find bugs
Triaging alarms to find bugs

Triaged 200 alarms (from Hopper), 189 false
Triaging alarms to find bugs

Triaged 200 alarms (from Hopper), 189 false

Reasons: insufficient Android modeling, imprecise container and string domains
Triaging alarms to find bugs

Triaged 200 alarms (from Hopper), 189 false

Reasons: insufficient Android modeling, imprecise container and string domains

Only 17 false alarms due to timeouts
Triaging alarms to find bugs

Triaged 200 alarms (from Hopper), 189 false

Reasons: insufficient Android modeling, imprecise container and string domains

Only 17 false alarms due to timeouts

Found 11 bugs in 4 apps (lastfm, seriesguide, connectbot, wordpress)

5 bugs due to **bad ordering assumptions**

10/11 patches accepted
Triaging alarms to find bugs

- Found 11 bugs in 4 apps (lastfm, seriesguide, connectbot, wordpress)
- 5 bugs due to bad ordering assumptions
- 10/11 patches accepted
Selective control-flow abstraction via a sound relevance relation
Selective control-flow abstraction via a sound relevance relation

Effective inter-event ordering-sensitive reasoning via data-relevance and control-feasibility
www.cs.colorado.edu/~bec
pl.cs.colorado.edu

Cerny
Chang
Hammer
Sankaranaryanan
Somenzi
| Staged Abstraction Refinement (e.g., CEGAR) | On-the-Fly Abstraction Coarsening |
Goal-directed: abstraction-refinement versus abstraction-coarsening

Staged abstraction refinement (e.g., CEGAR)

- Run analysis multiple times, abstraction changes only between runs.

On-the-fly abstraction coarsening

- Run analysis once, abstraction changes during analysis.
Goal-directed: abstraction-refinement versus abstraction-coarsening

staged abstraction refinement (e.g., CEGAR)  on-the-fly abstraction coarsening

**Staged**
- Run analysis multiple times, abstraction changes only between runs.

**Refinement**
- Start with imprecise abstraction, become more precise.

**On-the-fly**
- Run analysis once, abstraction changes during analysis.

**Coarsening**
- Start with precise abstraction, become less precise.