Analyzing, Abstracting, and Mining Event-Driven Systems

Sam Blackshear
Facebook

Shawn Meier

Maxwell Russek

Aleksandar Chakarov

Sergio Mover

Manu Sridharan
Samsung Research

Bor-Yuh Evan Chang

University of Colorado Boulder

ETH Zürich

October 8, 2016
Unfortunately, Angry Birds has stopped.

OK
Unfortunately, Angry Birds has stopped.
Unfortunately, Angry Birds has stopped.
A crash is magnified by the crowd
A crash is magnified by the crowd
A crash is magnified by the crowd
Reviews

2.9

- 5 stars: 16,993
- 4 stars: 9,140
- 3 stars: 4,928
- 2 stars: 4,648
- 1 star: 22,594

58,303 total reviews

Philip Husom ★ ★ ★ ★ ★
pretty awful Finally has ability to cast, but the constant crashing and freezing make it unusable.

Brad Laninga ★ ★ ★ ★ ★
Horrible Serious lag issues and stops playback randomly during live stream. I've had to uninstall and reinstall multiple times.

Jessica Calme ★ ★ ★ ★ ★
Drops out every five minutes When using Chromecast this app drops the live stream about every five minutes.

Peter Bullert ★ ★ ★ ★ ★
Sluggish and spyware Why this app needs to know with whom I'm talking on the phone? None of your business.

What's New

Bug fixes and optimizations
Philip Husom  ★★★★★
pretty awful Finally has ability to cast, but the constant crashing and freezing make it

Jessica Calme  ★★★★★
Drops out every five minutes When using Chromecast this app drops the live stream about

Brad Laninga  ★★★★★
Horrible Serious lag issues and stops playback randomly during live stream. I've had to

Peter Bullert  ★★★★★
Sluggish and spyware Why this app needs to know with whom I'm talking on the phone? None

WHAT'S NEW
Bug fixes and optimizations
Philip Husom pretty awful Finally has ability to cast, but the constant crashing and freezing make it

Jessica Calme Drops out every five minutes When using Chromecast this app drops the live stream about

Brad Laninga Horrible Serious lag issues and stops playback randomly during live stream. I've had to

Peter Bullert Sluggish and spyware Why this app needs to know with whom I'm talking on the phone? None

WHAT’S NEW
Bug fixes and optimizations

I don’t know how that field became null.
Ask the expert developers ...

Crash Fast: Square's Approach to Android Crashes

Pierre-Yves Ricau
@piwai
Feb 17 2016

```java
class RuntimeInit {
    public static void main(String[] arg) {
        commonInit();
    }

    private static void commonInit() {
        Thread.setDefaultUncaughtExceptionHandler(new UncaughtExceptionHandler()
            // ... init more stuff
        }

    private static class UncaughtExceptionHandler implements Thread.UncaughtExceptionHandler {
        public void uncaughtException(Thread t, Throwable e) {
            try {
                logCrashToLogcat(t, e);
                displayErrorDialog(e);
            } finally {
                // Try everything to make sure this process goes away.
                Process.killProcess(Process.myPid());
                System.exit(1);
            }
        }
    }
}
```
Ask the expert developers ...

Inevitable. Not necessarily app’s fault.
Ask the expert developers …

Inevitable. Not necessarily app’s fault.

Log information when it happens.
Inevitable. Not necessarily app’s fault.

Log information when it happens.

Check conditions to crash early and “fast” ...
Ask the expert developers …

Inevitable. Not necessarily app’s fault.

Log information when it happens.

Check conditions to crash early and “fast” …

… to be more likely seen in testing.
Suppose we’re lucky and get a crash. Now what?

- Where in the code (which callback) does the field get set to null?
- Why did that callback happen before this one?
- Is there another callback that should’ve reset the field to be non-null?
Suppose we’re lucky and get a crash. Now what?

- Where in the code (which callback) does the field get set to null?
- Why did that callback happen before this one?
- Is there another callback that should’ve reset the field to be non-null?

I don’t know how that field became null.
I don’t know how that field became null.

Often: A misunderstanding of how the (sometimes modified) framework interacts with the app.

Suppose we’re lucky and get a crash. Now what?

- Where in the code (which callback) does the field get set to null?
- Why did that callback happen before this one?
- Is there another callback that should’ve reset the field to be non-null?
I don’t know how that field became null.

Often: A misunderstanding of how the (sometimes modified) framework interacts with the app.

Bug from violating (implicit) framework protocol rules.
REVIEW

2.9

Star Rating:
- ★★★★★: 16,993
- ★★★★☆☆: 9,140
- ★★★☆☆☆: 4,928
- ★★★☆☆☆: 4,648
- ★★☆☆☆☆: 22,594

Total Reviews: 58,303

Philip Husom ★★★★★:
pretty awful Finally has ability to cast, but the constant crashing and freezing make it

Brad Laninga ★★★★★:
Horrible Serious lag issues and stops playback randomly during live stream. I've had to

Jessica Calme ★★★★★:
Drops out every five minutes  When using Chromecast this app drops the live stream about

Peter Bullert ★★★★★:
Sluggish and spyware Why this app needs to know with whom I'm talking on the phone? None

WHAT'S NEW

Bug fixes and optimizations
REVIEWS

2.9

★ 5 16,993
★ 4 9,140
★ 3 4,928
★ 2 4,648
★ 1 22,594

58,303 total

Philip Husom ★★★★★

pretty awful Finally has ability to cast, but the constant crashing and freezing make it

Jessica Calme ★★★★★

Drops out every five minutes When using Chromecast this app drops the live stream about

Brad Laninga ★★★★★★

Horrible Serious lag issues and stops playback randomly during live stream. I've had to

Peter Bullert ★★★★★

Sluggish and spyware Why this app needs to know with whom I'm talking on the phone? None

WHAT'S NEW

Bug fixes and optimizations
### Reviews

<table>
<thead>
<tr>
<th>Rating</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>16,993</td>
</tr>
<tr>
<td>4</td>
<td>9,140</td>
</tr>
<tr>
<td>3</td>
<td>4,928</td>
</tr>
<tr>
<td>2</td>
<td>4,648</td>
</tr>
<tr>
<td>1</td>
<td>22,594</td>
</tr>
</tbody>
</table>

**Philip Husom**

*pretty awful* Finally has ability to cast, but the constant crashing and freezing make it

**Jessica Calme**

*Drops out every five minutes* When using Chromecast this app drops the live stream about

**Brad Laninga**

*Horrible* Serious lag issues and stops playback randomly during live stream. I've had to

**Peter Bullert**

*Sluggish and spyware* Why this app needs to know with whom I'm talking on the phone? None

### What's New

- Bug fixes and optimizations
Imagining social programming ...

I don’t know how that field became null.
Imagining social programming ...

I am not alone

I don’t know how that field became null.
Imagining social programming ...

I am not alone

HELP! I don’t know how that field became null.
Imagining social programming ...

I am **not** alone

I don’t know how that field became **null**.
I am not alone

I don’t know how that field became null
Imagining social programming ...

I am not alone

“Transfer” the bug fix with program analysis and synthesis

I don’t know how that field became `null`
Our task in this talk
Our task in this talk

Prove and triage safety properties in event-driven applications (assuming protocol specifications)
Our task in this talk

Prove and triage safety properties in event-driven applications (assuming protocol specifications)

Mine artifacts for protocol specifications to subsequently “transfer” bug fixes
Our task in this talk

Prove and triage safety properties in event-driven applications (assuming protocol specifications)

**Hopper**: Goal-Directed Program Analysis with Jumping

Mine artifacts for protocol specifications to subsequently “transfer” bug fixes
Our task in this talk

Prove and triage safety properties in event-driven applications (assuming protocol specifications)

**Hopper**: Goal-Directed Program Analysis with Jumping

Mine artifacts for protocol specifications to subsequently “transfer” bug fixes

**Fixr**: Mining and Understanding Bug Fixes
Our task in this talk

Prove and triage safety properties in event-driven applications (assuming protocol specifications)

**Hopper**: Goal-Directed Program Analysis with Jumping

**Mine**

subsequently “

**Fixr**
Hopper: Goal-Directed Program Analysis with Jumping

Crash

Fortunately, Angry Birds has stopped.
OK
3% of all commit messages say “NullPointerException”
3% of all commit messages say “NullPointerException”
Callback-oriented programming: A partially ordered “lifecycle”
Callback-oriented programming:
A partially ordered “lifecycle”
Callback-oriented programming: A partially ordered “lifecycle”
Callback-oriented programming: A partially ordered “lifecycle”
callbacks (e.g., Activity.onCreate)
Callback-oriented programming: A partially ordered “lifecycle”

callbacks (e.g., Activity.onCreate)

Activity

- onCreate
- onResume
- onClick
- onPause
- onDestroy
Callback-oriented programming: A partially ordered “lifecycle”

- App
- Android Framework

Callbacks (e.g., Activity.onCreate)

Android components have an ordered lifecycle

**Activity**

- onCreate
- onResume
- onClick
- onPause
- onDestroy
Callback-oriented programming:
A partially ordered “lifecycle”

Android components have an ordered lifecycle

callbacks (e.g., Activity.onCreate)

Activity

onCreate

onResume

onClick

onPause

onDestroy

this.mHostDb = null;
this.mService = null;
Callback-oriented programming: A partially ordered “lifecycle”

Android components have an ordered lifecycle

callbacks (e.g., Activity.onCreate)

Activity

onCreate

onResume

onClick

onPause

onDestroy

this.mHostDb = null;
this.mService = null;

Collect resources when done
Callback-oriented programming:
A partially ordered “lifecycle”

Android components have an ordered lifecycle

callbacks (e.g., Activity.onCreate)

But, lifecycles of different components and other callbacks can interleave ...

Activity
- onCreate
- onResume
- onClick
- onPause
- onDestroy

this.mHostDb = null;
this.mService = null;

Collect resources when done
Callback-oriented programming: A partially ordered “lifecycle”

Need to eagerly release resources but safety (e.g., of dereferences) depends on callback interleaving.

Android components have an ordered lifecycle.

callbacks (e.g., Activity.onCreate)

But, lifecycles of different components and other callbacks can interleave ...

```java
this.mHostDb = null;
this.mService = null;
```

Collect resources when done.
Callback-oriented programming:
Interacting through a shared heap
Callback-oriented programming:
Interacting through a shared heap
Callback-oriented programming:
Interacting through a shared heap

Heap

onCreate → onResume → onClick → onPause → onDestroy

onCreate → onResume → onClick → onPause → onDestroy
Callback-oriented programming: Interacting through a shared heap

Heap

... and operate over a shared, global heap

onCreate → onResume → onClick → onPause → onDestroy

onCreate → onResume → onClick → onPause → onDestroy
Callback-oriented programming:
Interacting through a shared heap

Safety (e.g., of dereferences) depends on the order of heap writes that depends on the interleaving of callbacks
Explore callback interleavings …
Explore callback interleavings ...
Explore callback interleavings …
Explore callback interleavings ...

Previous analyses do not consider inter-component interleavings in a flow-sensitive way.
Explore callback interleavings ...

Previous analyses do not consider inter-component interleavings in a flow-sensitive way.

An app with 1,320 callbacks would have created a product automaton with $10^{111}$ nodes (with unsoundly one instance per class).
... it shouldn’t be so hard
... it shouldn’t be so hard

```
this.mHostDb.s()
```
... it shouldn't be so hard

```
this.mHostDb.s()
```

Diagram:
- onCreate
  - onResume
  - onPause
  - onDestroy
  - onClick (safe?)
... it shouldn’t be so hard

Idea: Safety of a particular dereference should not require reasoning about all callback interleavings
... it shouldn’t be so hard

Idea: Safety of a particular dereference should not.

A “smart” goal-directed analysis could consider relevant callback orderings without considering all of them.
... it shouldn’t be so hard

Idea: Safety of a particular dereference should not require reasoning about all callback interleavings. A "smart" goal-directed analysis could consider relevant callback orderings without considering all of them.
Goal-directed program analysis

```
safe?
this.mHostDb.s()
```
Goal-directed program analysis

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

safe?

this.mHostDb.s()
Given a program configuration **goal**, derive a **contradiction** w.r.t. its reachability

```
mHostDb == null
```

```
this.mHostDb.s()
```

```java
mHostDb == null
```
Goal-directed program analysis

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

```
true  false  true  true  false

mHostDb == null
```

```
this.mHostDb.s()
```

```
mHostDb == null
```
Goal-directed program analysis

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

\[
\text{(this} \rightarrow \hat{t} \ast \hat{t} \cdot \text{mHostDb} \rightarrow \hat{a} \ast \text{true}) \land \hat{a} = \text{null}
\]
Given a program configuration \textbf{goal}, derive a \textbf{contradiction} w.r.t. its reachability.

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

**Thresher:** A backwards abstract interpretation with separation logic constraints to \textbf{refute} error conditions [PLDI’13]
Goal-directed program analysis

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

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

Over-approximate

Thresher: A backwards abstract interpretation with separation logic constraints to refute error conditions [PLDI’13]
Being smart ...
Two dereferences: one safe and one buggy

onCreate

onResume

onClick

onPause

onDestroy
void onClick(...) {
    this.mHostDb.s(this.mService.g());
}
Being smart ...

Two dereferences: one safe and one buggy

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

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

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

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

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

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

void onDestroy(...) {
    this.mHostDb = null;
    this.mService = null;
}
```
void onCreate() {
    bindService(..., new ServiceConn {
        void connected(@NonNull Service s) {
            this.mService = s;
        }
    });
    this.mHostDb = new Db();
}

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Need to consider **some** but not all callback ordering constraints using data relevance

### lifecycle constraints relevant

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

- **onBind**
- **onConnected**
- **onDisconnected**
- **onUnbind**
- **onRebind**
- **onDestroy**
Idea: **Jumping to relevant callbacks**

- onCreate
- onResume
- onClick
- onPause
- onDestroy

---

- onBind
- onConnected
- onDisconnected
- onUnbind
- onRebind
- onDestroy

---

**safe?**
Idea: **Jumping to relevant callbacks**

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

- onCreate
- onResume
- onClick
- onPause
- onDestroy

safe?

Find **data-relevant** callbacks

- onBind
- onConnected
- onDisconnected
- onUnbind
- onDestroy

- onCreate
- onResume
- onClick
- onPause
- onDestroy

- onBind
- onConnected
- onDisconnected
- onUnbind
- 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

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: Hopper is an analysis that jumps
Contributions: Hopper is an analysis that jumps

\[ \langle Q, \ell \rangle \leadsto T \]

Framework for sound jumping analyses
Contributions: Hopper is an analysis that jumps

1. \( \langle Q, \ell \rangle \xrightarrow{\sim} T \)

Framework for sound jumping analyses

2. Applied to Android lifecycles
Contributions: Hopper is an analysis that jumps

1. Framework for sound jumping analyses
   \[ \langle Q, \ell \rangle \leadsto T \]

2. Applied to Android lifecycles
Interleave data-relevance with control-feasibility to realize jumping
Interleave data-relevance with control-feasibility to realize jumping

\[ \langle Q, \ell \rangle \rightsquigarrow T \]
Interleave data-relevance with control-feasibility to realize jumping

\[ \langle Q, \ell \rangle \leadsto T \]
Interleave data-relevance with control-feasibility to realize jumping

Follow the normal control-flow of the program
Interleave data-relevance with control-feasibility to realize jumping

$T$

next
transitions

transition
relation

$\langle Q, \ell \rangle \rightsquigarrow T$

current
location

$\ell$

current
query

$Q$
Interleave data-relevance with control-feasibility to realize jumping

\[ \langle Q, \ell \rangle \rightsquigarrow T \]
Interleave data-relevance with control-feasibility to realize jumping
Interleave data-relevance with control-feasibility to realize jumping

\[
\langle Q, l \rangle \leadsto T
\]

next transitions

relevance relation

Follow some other transitions based on query \( Q \)
Interleave data-relevance with control-feasibility to realize jumping

\[ \langle Q, \ell \rangle \leadsto T \]
Interleave data-relevance with control-feasibility to realize jumping

Identify (with analysis) program locations that can affect query $Q$
Interleave **data-relevance with control-feasibility** to realize jumping

\[ \langle Q, \ell \rangle \rightsquigarrow T \]

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

\[ T \]

next transitions

Identify (with analysis) program locations that can affect query \( Q \)

Filter to the locations that can feasibly reach \( \ell \) (without “going through” any other transition in \( T \) )

current location

current query
Data-relevance identifies relevant writes

Identify (with analysis) program locations that can affect query $Q$
Data-relevance identifies relevant writes

Identify (with analysis) program locations that can affect query $Q$
Data-relevance identifies relevant writes

\[ \langle Q, \ell \rangle \sim T \]

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

Identify (with analysis) program locations that can affect query \( Q \)

\[ \ell_1 \quad \ell_2 \quad \ell_3 \quad \ldots \]

\( \ell \)
\( Q \)

current location

current query
Classic idea: Following data dependencies yields a **sparse** analysis (but, here, flow-insensitive)

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

Identify (with analysis) program locations that can affect query $Q$
Control-feasibility recovers flow-sensitivity

\[ \langle Q, \ell \rangle \sim T \]
Control-feasibility recovers flow-sensitivity

$\langle Q, \ell \rangle \sim T$

Diagram:

```
Control-feasibility

$l_1 \quad l_2 \quad l_3 \quad \cdots$

Data-relevance

$l \quad Q$

current location

current query
```
Control-feasibility recovers flow-sensitivity

\[ \langle Q, \ell \rangle \sim T \]
Control-feasibility recovers flow-sensitivity

\[ \langle Q, \ell \rangle \leadsto T \]
Control-feasibility recovers flow-sensitivity

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

\( \langle Q, \ell \rangle \leadsto T \)
Filter the set of data-relevant locations using control flow and the current program point.
Filter the set of data-relevant locations using control flow and the current program point.

Not backward-reachable from current location.
Control-feasibility recovers flow-sensitivity

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

Must visit another relevant location first.

Not backward-reachable from current location.
Insensitive versus sensitive

\[ \langle Q, l \rangle \sim T \]

Diagram:

- \( l_1 \) \( l_2 \) \( l_3 \) \( \ldots \)
- \( l \)
- \( Q \)

Labelled boxes:
- Control-feasibility
- Data-relevance

Labels:
- current location
- current query
Insensitive versus sensitive

Different choices yield classical variants of flow, path, and context sensitivity/insensitivity

\[ \langle Q, \ell \rangle \sim T \]
Insensitive versus sensitive

Different choices yield classical variants of flow, path, and context sensitivity/insensitivity

Make sparse
Insensitive versus sensitive

Data-relevance

Control-feasibility

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

Make sparse

current location

current query

\( \langle Q, \ell \rangle \sim T \)
Insensitive versus sensitive

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

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

Different choices yield classical variants of flow, path, and context sensitivity/insensitivity.
Soundness

\[ \langle Q, \ell \rangle \leadsto T \]

\[
\begin{align*}
\ell_1 & \quad \ell_2 & \quad \ell_3 & \quad \ldots \\
\text{Data-relevance} & \\
\ell & \quad Q
\end{align*}
\]

current location

current query

Control-feasibility
Theorem: If data-relevance and control-feasibility are sound, then no behavior relevant to refuting $Q$ can be missed (i.e., "jumping is sound")
Contributions: Hopper is an analysis that jumps

1. $\langle Q, \ell \rangle \rightsquigarrow T$

Framework for sound jumping analyses

2. Applied to Android lifecycles
Contributions: Hopper is an analysis that jumps

onClick  

onCreate  

onConnected  

\langle Q, \ell \rangle \leadsto T

Framework for sound jumping analyses

Applied to Android lifecycles
An effective jumping policy for inter-event Android analysis
An effective jumping policy for inter-event Android analysis

Within an event-callback (intra-event), follow predecessor transitions
An effective jumping policy for inter-event Android analysis

Within an event-callback (intra-event), follow predecessor transitions still feasible to be as precise as possible within callbacks
An effective jumping policy for **inter-event** Android analysis

Within an event-callback (**intra-event**), follow predecessor transitions

still feasible to be as precise as possible within callbacks

Between event-callbacks (**inter-event**), jump using lifecycle graphs for control-feasibility filtering
An effective jumping policy for **inter-event** Android analysis

Within an event-callback (**intra-event**), follow predecessor transitions

- still feasible to be as precise as possible within callbacks

Between event-callbacks (**inter-event**), jump using lifecycle graphs for control-feasibility filtering

- avoiding costly and unnecessary interleavings
Is jumping effective for inter-event analysis?
Is jumping effective for inter-event analysis?

Proving all dereferences safe
Is jumping effective for inter-event analysis?

Proving **all** dereferences safe

... for evaluation. But a use-case could be directed by the user.
Is jumping effective for inter-event analysis?

Proving all dereferences safe
Is jumping effective for inter-event analysis?

Proving all dereferences safe
10 open source Android apps
3,000 to 57,000 lines of code
10 to 100 components
120 to 1,320 callbacks
Is jumping effective for inter-event analysis?

Proving all dereferences safe

10 open source Android apps

3,000 to 57,000 lines of code

10 to 100 components

120 to 1,320 callbacks

Event product graph would have $10^{10}$ to $10^{111}$ nodes (with unsoundly one instance per class)
Is jumping effective for inter-event analysis?

Proving all dereferences safe
10 open source Android apps
3,000 to 57,000 lines of code
10 to 100 components
120 to 1,320 callbacks

Event product graph would have $10^{10}$ to $10^{111}$ nodes (with unsoundly one instance per class)

Previous analyses do not consider inter-component interleavings in a flow-sensitive way.
Is jumping effective for inter-event analysis?

Proving all dereferences safe
10 open source Android apps
3,000 to 57,000 lines of code
10 to 100 components
120 to 1,320 callbacks

Previous analyses do not consider inter-component interleavings in a flow-sensitive way.
Is jumping effective for inter-event analysis?

Proving all dereferences safe
10 open source Android apps
3,000 to 57,000 lines of code
10 to 100 components
120 to 1,320 callbacks

Compared 3 analyses

**Nit**: type-based (flow-insensitive)
**Thresher**: goal-directed path-sensitive
**Hopper**: goal-directed jumping

Previous analyses do not consider inter-component interleavings in a flow-sensitive way
Is jumping effective?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td><strong>266</strong></td>
</tr>
</tbody>
</table>
Is jumping effective?

<table>
<thead>
<tr>
<th>Project</th>
<th>KLOC</th>
<th>Deref</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
</tr>
</tbody>
</table>

**Summary**  
266  62562
Is jumping effective?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
</tr>
</tbody>
</table>

Summary: 266 62562

Huge number of dereferences
### Is jumping effective?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
</tr>
</tbody>
</table>

**Summary**

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>266</td>
<td>62562</td>
<td>41071</td>
</tr>
</tbody>
</table>
Is jumping effective?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td><strong>266</strong></td>
<td><strong>62562</strong></td>
<td><strong>41071</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>
</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>
</tbody>
</table>

Summary: 266 62562 41071 9968
Is jumping effective?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</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>Summary</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>
Is jumping effective?

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

Find callback interleavings

Are given callback interleavings feasible?
### Is jumping effective?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop</th>
<th>% Improve</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72</td>
<td>60</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51</td>
<td>72</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143</td>
<td>72</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477</td>
<td>50</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290</td>
<td>52</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625</td>
<td>37</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74</td>
<td>77</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330</td>
<td>53</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988</td>
<td>36</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362</td>
<td>44</td>
</tr>
<tr>
<td><strong>Summary</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</strong></td>
<td><strong>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</th>
<th>% Improve</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</td>
<td>60</td>
<td>92</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51</td>
<td>72</td>
<td>94</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143</td>
<td>72</td>
<td>93</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477</td>
<td>50</td>
<td>90</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290</td>
<td>52</td>
<td>92</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625</td>
<td>37</td>
<td>92</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74</td>
<td>77</td>
<td>97</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330</td>
<td>53</td>
<td>94</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988</td>
<td>36</td>
<td>90</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362</td>
<td>44</td>
<td>91</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>266</td>
<td>62562</td>
<td>41071</td>
<td>9968</td>
<td>5412</td>
<td>54</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>KLOC</td>
<td>Deref</td>
<td>Nit</td>
<td>Thr</td>
<td>Hop</td>
<td>% Improve</td>
<td>% Proven</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
<td>-------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----------</td>
<td>----------</td>
</tr>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72</td>
<td>60</td>
<td>92</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51</td>
<td>72</td>
<td>94</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143</td>
<td>72</td>
<td>93</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477</td>
<td>50</td>
<td>90</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290</td>
<td>52</td>
<td>92</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625</td>
<td>37</td>
<td>92</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74</td>
<td>77</td>
<td>97</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330</td>
<td>53</td>
<td>94</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988</td>
<td>36</td>
<td>90</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362</td>
<td>44</td>
<td>91</td>
</tr>
<tr>
<td><strong>Summary</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</strong></td>
<td><strong>54</strong></td>
<td><strong>92</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</th>
<th>% Improve</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</td>
<td>60</td>
<td>92</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51</td>
<td>72</td>
<td>94</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143</td>
<td>72</td>
<td>93</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477</td>
<td>50</td>
<td>90</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290</td>
<td>52</td>
<td>92</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625</td>
<td>37</td>
<td>92</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74</td>
<td>77</td>
<td>97</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330</td>
<td>53</td>
<td>94</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988</td>
<td>36</td>
<td>90</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362</td>
<td>44</td>
<td>91</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>266</td>
<td>62562</td>
<td>41071</td>
<td>9968</td>
<td>5412</td>
<td><strong>54</strong></td>
<td><strong>92</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</th>
<th>% Improve</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</td>
<td>60</td>
<td>92</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51</td>
<td>72</td>
<td>94</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143</td>
<td>72</td>
<td>93</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477</td>
<td>50</td>
<td>90</td>
</tr>
<tr>
<td>github</td>
<td></td>
<td>290</td>
<td></td>
<td></td>
<td></td>
<td>52</td>
<td>92</td>
</tr>
<tr>
<td>seriesguide</td>
<td></td>
<td>625</td>
<td></td>
<td></td>
<td></td>
<td>37</td>
<td>92</td>
</tr>
<tr>
<td>connectbot</td>
<td></td>
<td>74</td>
<td></td>
<td></td>
<td></td>
<td>77</td>
<td>97</td>
</tr>
<tr>
<td>textsecure</td>
<td></td>
<td>330</td>
<td></td>
<td></td>
<td></td>
<td>53</td>
<td>94</td>
</tr>
<tr>
<td>k-9</td>
<td></td>
<td>1988</td>
<td></td>
<td></td>
<td></td>
<td>36</td>
<td>90</td>
</tr>
<tr>
<td>wordpress</td>
<td></td>
<td>1362</td>
<td></td>
<td></td>
<td></td>
<td>44</td>
<td>91</td>
</tr>
<tr>
<td><strong>Summary</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</strong></td>
<td><strong>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!
## Is jumping effective?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop</th>
<th>% Improve</th>
<th>% Proven</th>
<th>Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72</td>
<td>60</td>
<td>92</td>
<td>12.0</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51</td>
<td>72</td>
<td>94</td>
<td>11.5</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143</td>
<td>72</td>
<td>93</td>
<td>98.7</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477</td>
<td>50</td>
<td>90</td>
<td>29.2</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290</td>
<td>52</td>
<td>92</td>
<td>62.5</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625</td>
<td>37</td>
<td>92</td>
<td>232.2</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74</td>
<td>77</td>
<td>97</td>
<td>14.7</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330</td>
<td>53</td>
<td>94</td>
<td>38.4</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988</td>
<td>36</td>
<td>90</td>
<td>453.3</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362</td>
<td>44</td>
<td>91</td>
<td>309.9</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>266</td>
<td>62562</td>
<td>41071</td>
<td>9968</td>
<td>5412</td>
<td>54</td>
<td>92</td>
<td>1262.3</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</th>
<th>% Improve</th>
<th>% Proven</th>
<th>Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72</td>
<td>60</td>
<td>92</td>
<td>12.0</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51</td>
<td>72</td>
<td>94</td>
<td>11.5</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143</td>
<td>72</td>
<td>93</td>
<td>98.7</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477</td>
<td>50</td>
<td>90</td>
<td>29.2</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290</td>
<td>52</td>
<td>92</td>
<td>62.5</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625</td>
<td>37</td>
<td>92</td>
<td>232.2</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74</td>
<td>77</td>
<td>97</td>
<td>14.7</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330</td>
<td>53</td>
<td>94</td>
<td>38.4</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988</td>
<td>36</td>
<td>90</td>
<td>453.3</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362</td>
<td>44</td>
<td>91</td>
<td>309.9</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>266</td>
<td>62562</td>
<td>41071</td>
<td>9968</td>
<td>5412</td>
<td>54</td>
<td>92</td>
<td>1262.3</td>
</tr>
</tbody>
</table>

- **unproven derefs**
- **jumping effectiveness**
- **type-based**
- **no jumping**
- **jumping**

*~1 sec per deref*
### Is jumping effective?

<table>
<thead>
<tr>
<th></th>
<th>KLOC</th>
<th>Deref</th>
<th>Nit</th>
<th>Thr</th>
<th>Hop</th>
<th>% Improve</th>
<th>% Proven</th>
<th>Time (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>drupaleditor</td>
<td>3</td>
<td>928</td>
<td>679</td>
<td>179</td>
<td>72</td>
<td>60</td>
<td>92</td>
<td>12.0</td>
</tr>
<tr>
<td>npr</td>
<td>5</td>
<td>829</td>
<td>617</td>
<td>181</td>
<td>51</td>
<td>72</td>
<td>94</td>
<td>11.5</td>
</tr>
<tr>
<td>duckduckgo</td>
<td>11</td>
<td>1969</td>
<td>1341</td>
<td>518</td>
<td>143</td>
<td>72</td>
<td>93</td>
<td>98.7</td>
</tr>
<tr>
<td>lastfm</td>
<td>13</td>
<td>4840</td>
<td>3528</td>
<td>954</td>
<td>477</td>
<td>50</td>
<td>90</td>
<td>29.2</td>
</tr>
<tr>
<td>github</td>
<td>19</td>
<td>3603</td>
<td>2520</td>
<td>601</td>
<td>290</td>
<td>52</td>
<td>92</td>
<td>62.5</td>
</tr>
<tr>
<td>seriesguide</td>
<td>32</td>
<td>8184</td>
<td>5438</td>
<td>986</td>
<td>625</td>
<td>37</td>
<td>92</td>
<td>232.2</td>
</tr>
<tr>
<td>connectbot</td>
<td>33</td>
<td>2190</td>
<td>1562</td>
<td>316</td>
<td>74</td>
<td>77</td>
<td>97</td>
<td>14.7</td>
</tr>
<tr>
<td>textsecure</td>
<td>38</td>
<td>5921</td>
<td>3643</td>
<td>698</td>
<td>330</td>
<td>53</td>
<td>94</td>
<td>38.4</td>
</tr>
<tr>
<td>k-9</td>
<td>55</td>
<td>19032</td>
<td>11968</td>
<td>3104</td>
<td>1988</td>
<td>36</td>
<td>90</td>
<td>453.3</td>
</tr>
<tr>
<td>wordpress</td>
<td>57</td>
<td>15066</td>
<td>9775</td>
<td>2431</td>
<td>1362</td>
<td>44</td>
<td>91</td>
<td>309.9</td>
</tr>
<tr>
<td><strong>Summary</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</strong></td>
<td><strong>54</strong></td>
<td><strong>92</strong></td>
<td><strong>1262.3</strong></td>
</tr>
</tbody>
</table>

Hopper **proves** 92% of dereferences safe with interleaving of callbacks from an arbitrary number of components.
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
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
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

one not accepted in a seemingly inactive project
Summary: Hopper is an analysis that jumps
Summary: Hopper is an analysis that jumps next locations via a sound relevance relation.

Selective control-flow abstraction via a sound relevance relation.
Summary: Hopper is an analysis that jumps

Selective control-flow abstraction via a sound relevance relation

Inter-event ordering-sensitive reasoning via data-relevance and lifecycle control-feasibility
Our task in this talk

Prove and triage safety properties in event-driven applications (assuming protocol specifications)

**Hopper**: Goal-Directed Program Analysis with Jumping

Mine artifacts for protocol specifications to subsequently “transfer” bug fixes

**Fixr**: Mining and Understanding Bug Fixes for Event-Driven Protocols
Our task in this talk

Prove and triage applications (assuming protocol specifications)

Hopper

Mine artifacts for protocol specifications to subsequently “transfer” bug fixes

Fixr: Mining and Understanding Bug Fixes for Event-Driven Protocols


Fixr: Mining and Understanding Bug Fixes for Event-Driven Protocols

Bor-Yuh Evan Chang  Kenneth M. Anderson  Pavol Černý  Sriram Sankaranarayanan  Tom Yeh
Edmund S.L. Lam  Sergio Mover  Shawn Meier  Rhys Braginton Pettee Olsen  Maxwell Russek

University of Colorado Boulder
Fixr: Mining and Understanding Bug Fixes for Event-Driven Protocols

I am not alone

"Transfer" the bug fix with program analysis and synthesis

I don't know how that field became null.

Bor-Yuh Evan Chang

Edmund S.L. Lam
Fixr: Mining and Understanding Bug Fixes for Event-Driven Protocols

University of Colorado Boulder
The Fixr Project

Deltar: Inferring Semantic Deltas and Repair Specifications

Prepair: Deriving Probabilistic Repair Specifications

Harvestr: Social Validation and Mining of Fixes

Patchr: Detecting Potential Bugs and Synthesizing Patches

FixrDB

semantic
statistical-semantic
syntactic
social

interaction
commit

Github
The **Fixr** Project

- **Deltar**: Inferring Semantic Deltas and Repair Specifications
- **Prepair**: Deriving Probabilistic Repair Specifications
- **Harvestr**: Social Validation and Mining of Fixes
- **Patchr**: Detecting Potential Bugs and Synthesizing Patches

**FixrDB**

- Symbolic program analysis
  - Semantic
  - Statistical-semantic
  - Syntactic
  - Social

**Github**

- Interaction
- Commit
The **Fixr** Project

**Deltar**: Inferring Semantic Deltas and Repair Specifications

**Prepair**: Deriving Probabilistic Repair Specifications

**Harvestr**: Social Validation and Mining of Fixes

**Patchr**: Detecting Potential Bugs and Synthesizing Patches

**Github**

- symbolic program analysis
- numerical-probabilistic program analysis
- semantic
- statistical-semantic
- syntactic
- social
- interaction
- commit

**FixrDB**

- repair specification
- semantic facts
- probabilistic repair specification
- program synthesis
The **Fixr** Project

**Deltar**: Inferring Semantic Deltas and Repair Specifications

**Prepair**: Deriving Probabilistic Repair Specifications

**Harvestr**: Social Validation and Mining of Fixes

**Patchr**: Detecting Potential Bugs and Synthesizing Patches

FixrDB

**symbolic program analysis**

**semantic**

**statistical-semantic**

**syntactic**

**social**

**interaction**

**commit**

**user-centered big data analytics**

**numerical-probabilistic program analysis**

**program synthesis**
The **Fixr Project**

**Deltar**: Inferring Semantic Deltas and Repair Specifications

**Prepair**: Deriving Probabilistic Repair Specifications

**Harvestr**: Social Validation and Mining of Fixes

**Patchr**: Detecting Potential Bugs and Synthesizing Patches

**FixrDB**

**FixrDB**

**Github**

**symbolic program analysis**

**numerical-probabilistic program analysis**

**software engineering for big data**

**interaction**

**commit**

**user-centered big data analytics**

**program synthesis**

**semantic**

**statistical-semantic**

**syntactic**

**social**

**fix**

**repair**

**specification**

**repair specification**

**semantic facts**

**probalistic repair specification**

**program synthesis**

**user-centered big data analytics**

**program analysis**

**big data analytics**

**big data**

**software engineering**

**syntactic**

**statistical-semantic**

**symbolic**

**numerical-probabilistic**

**program analysis**
The **Fixr** Project

Multi-faculty project collaboratively investigating “transferring bug fixes” by mining code commits

**Deltar**: Inferring Semantic Deltas and Repair Specifications

**Harvestr**: Social Validation and Mining of Fixes

**Prepair**: Deriving Probabilistic Repair Specifications

**Patchr**: Detecting Potential Bugs and Synthesizing Patches

**symbolic program analysis**

**semantic**

**statistical-semantic**

**syntactic**

**social**

**interaction**

**commit**

**GitHub**

**user-centered big data analytics**

**user-centered big data analytics**

**social**

**semantic**

**statistical-semantic**

**syntactic**

**social**

**interaction**

**commit**

**GitHub**

**symbolic program analysis**

**numerical-probabilistic program analysis**

Tom Yeh

Bor-Yuh Evan Chang

Sriram Sankaranarayanan

Ken Anderson

Pavol Cerny

Tom Yeh

Bor-Yuh Evan Chang

Sriram Sankaranarayanan

Ken Anderson

Pavol Cerny
Abstracting Event-Driven Systems with Lifestate Specifications
Android is an event-driven system
Android is an event-driven system
Android is an event-driven system

Callback ordering constraints are not static
Android is an event-driven system
Android is an event-driven system
Android is an event-driven system
Android is an event-driven system

Event: User clicks button

Framework

Application
Android is an event-driven system

Event: User clicks button

Framework

activity.onClick()

Application
Android is an event-driven system

Event: User clicks button

A callback is where the framework invokes an application method
Android is an event-driven system

A callback is where the framework invokes an application method

Event: User clicks button

Framework

Application

activity.onClick()

asynctask.execute()
Android is an event-driven system

Event: User clicks button

A callback is where the framework invokes an application method

A callin is where the application invokes a framework method
Android is an event-driven system.

Event: User clicks button

Framework

Application

activity.onClick()

A callback is where the framework invokes an application method.

A callin is where the application invokes a framework method.

asyncTask.execute()
Android is an event-driven system

Event: User clicks button

A callback is where the framework invokes an application method

A callin is where the application invokes a framework method

onClick returns to the framework

activity.onClick()
Android is an event-driven system

Event: User clicks button
- **activity.onClick()**
- **asynctask.execute()**

Event: Background task finishes

A **callback** is where the framework invokes an application method.

A **callin** is where the application invokes a framework method.

**onClick** returns to the framework.
Android is an event-driven system

Event: User clicks button

Event: Background task finishes

A callback is where the framework invokes an application method

A callin is where the application invokes a framework method

onClick returns to the framework
Android is an event-driven system

Event: User clicks button

activity.onClick()

A callback is where the framework invokes an application method

asynctask.execute()

A callin is where the application invokes a framework method

asynctask.onPostExecute()

onClick returns to the framework

onPostExecute cannot happen unless execute has been called

Event: Background task finishes
Android is an event-driven system

The event-driven framework uses **callbacks** to notify the application of events and the application uses **callins** to affect how the framework invokes future callbacks.
Android is an event-driven system

Event: User clicks button
- activity.onClick()
- asynctask.execute()

Event: Background task finishes
- asynctask.onPostExecute()
Android is an event-driven system
Android is an event-driven system

Event: User clicks button

Event: User clicks button

Event: Background task finishes
Android is an event-driven system

- **Event: User clicks button**
  - `activity.onClick()`
  - `asynctask.execute()`
  - Activity's `onClick()` method is called again.

- **Event: Background task finishes**
  - `asynctask.onPostExecute()`
Android is an event-driven system
Event: User clicks button

Exception: Cannot call `execute` on same `AsyncTask` instance.
Android is an event-driven system

Event: User clicks button

Exception: Cannot call `execute` on same `AsyncTask` instance.

```java
button.setOnClickListener(new OnClickListener() {
    void onClick() {
        + button.disable();
        asynctask.execute();
    }
});
```
Android is an event-driven system

Event: User clicks button

Exception: Cannot call `execute` on same `AsyncTask` instance.

```
button.setOnClickListener(new OnClickListener() {
    void onClick() {
        button.disable();
        asynctask.execute();
    }
});
```
Android is an event-driven system

**Need**: Modeling and reasoning about how *callins* affect *callbacks* (and vice versa)

```
button.setOnClickListener(new OnClickListener() {
    void onClick() {
        button.disable();
        asynctask.execute();
    }
});
```

**Exception**: Cannot call `execute` on same `AsyncTask` instance.
Contributions: Lifestate
Contributions: Lifestate

\( \lambda \text{life}: \) A (concrete) model of event-driven systems capturing how callins and callbacks affect each other
Contributions: Lifestate

\( \lambda \text{life}: \) A (concrete) model of event-driven systems capturing how callins and callbacks affect each other

Lifestate Rules: A specification language to model the effects of Android callins and callbacks
Contributions: Lifestate

\( \lambda \text{life}: \) A (concrete) **model** of event-driven systems capturing how callins and callbacks affect each other

**Lifestate Rules:** A **specification language** to model the effects of Android callins and callbacks

**DroidLife:** **Mining** lifestate specifications and verifying the absence of lifestate "races"
Enabled callbacks and allowed callins

Framework

Application
Enabled callbacks and allowed callins

**Enabled**

- Framework
- Application
Enabled callbacks and allowed callins

- **Enabled**
- **Framework**: action.onClick()
- **Application**
Enabled callbacks and allowed callins

**Enabled**

```
activity.onClick()
```

```
activity.onClick()
```
Enabled callbacks and allowed callings

**Enabled**

activity.onClick()

**Framework**

**Activity**

activity.onClick()

**Application**

asynctask.execute()
button.disable()
Enabled callbacks and allowed callins

- **Enabled**: 
  - `activity.onClick()`

- **Allowed**: 
  - `asynctask.execute()`
  - `button.disable()`
Enabled callbacks and allowed call-ins

**Enabled**

- `activity.onClick()`
- `activity.onClick()`
- `activity.onClick()`
- `asynctask.onPostExecute()`

**Framework**

**Application**

**Allowed**

- `asynctask.execute()`
- `button.disable()`
Enabled callbacks and allowed callins

**Enabled**

- `activity.onClick()`
- `activity.onClick()`
- `asynctask.execute()`
- `asynctask.onPostExecute()`
- `asynctask.execute()`

**Allowed**

- `asynctask.execute()`
- `button.disable()`
- `asynctask.execute()`
- `button.disable()`
Enabled callbacks and allowed callins

**Enabled**

- `activity.onClick()`
- `activity.onClick()`
- `activity.onClick()`
- `asynctask.execute()`
- `asynctask.onPostExecute()`

**Allowed**

- `asynctask.execute()`
- `button.disable()`
- `asynctask.execute()`
- `asynctask.execute()`
- `button.disable()`
Enabled callbacks and allowed callins

**Enabled**

- `activity.onClick()`
- `asynctask.execute()`
- `button.disable()`

**Framework**

**Application**

**Allowed**

- `asynctask.execute()`
- `button.disable()`
- `asynctask.onPostExecute()`
Enabled callbacks and allowed callins

- activity.onClick()
- asynctask.execute()
- button.disable()
- asynctask.onPostExecute()
Enabled callbacks and allowed callins

Model state: Enabled callbacks track what the framework may invoke next and allowed callins track what the application can invoke (without error)
Lifestate rules specify enabledness and allowedness effects.
Lifestate rules specify enabledness and allowedness effects

\[ \text{enable} \]

\[ message_1 \rightarrow_{cb} message_2 \]
Lifestate rules specify enabledness and allowedness effects.

A suspended callback or callin invocation (thunk)

```
enable
message_1 →cb message_2
```
Lifestate rules specify enabledness and allowedness effects

A suspended callback or callin invocation (thunk)

\[
\text{enable} \quad message_1 \rightarrow \text{cb} \quad message_2
\]

Message causes a callback to be added to the enabled set
Lifestate rules specify enabledness and allowedness effects

A suspended callback or call in invocation (thunk)

**enable**

\[message_1 \rightarrow_{cb} message_2\]

Message causes a callback to be added to the enabled set

**disable**

\[message_1 \rightarrow_{cb} message_2\]
Lifestate rules specify enabledness and allowedness effects.

A suspended callback or callin invocation (thunk)

**enable**

\[message_1 \rightarrow cb \ message_2\]

Message causes a callback to be added to the enabled set

**disable**

\[message_1 \leftrightarrow cb \ message_2\]

Message causes a callback to be removed to the enabled set
Lifestate rules specify enabledness and allowedness effects

A suspended callback or callin invocation (thunk)

**enable**

\[ message_1 \rightarrow \text{cb} \quad message_2 \]

Message causes a callback to be added to the enabled set

**disable**

\[ message_1 \rightarrow \text{<-cb} \quad message_2 \]

Message causes a callback to be removed to the enabled set

**allow**

\[ message_1 \rightarrow \text{ci} \quad message_2 \]
Lifestate rules specify enabledness and allowedness effects

- **enable**
  \[message_1 \rightarrow_{cb} message_2\]
  Message causes a callback to be added to the enabled set

- **disable**
  \[message_1 \leftarrow_{cb} message_2\]
  Message causes a callback to be removed to the enabled set

- **allow**
  \[message_1 \rightarrow_{ci} message_2\]
  Message causes a callin to be added to the allowed set

A suspended callback or callin invocation (thunk)
Lifestate rules specify enabledness and allowedness effects.

A suspended callback or callin invocation (thunk)

**Enable**

\[ \text{message}_1 \rightarrow \text{cb} \rightarrow \text{message}_2 \]

Message causes a callback to be added to the enabled set.

**Allow**

\[ \text{message}_1 \rightarrow \text{ci} \rightarrow \text{message}_2 \]

Message causes a callin to be added to the allowed set.

**Disable**

\[ \text{message}_1 \notightarrow \text{cb} \rightarrow \text{message}_2 \]

Message causes a callback to be removed from the enabled set.

**Disallow**

\[ \text{message}_1 \notightarrow \text{ci} \rightarrow \text{message}_2 \]

Message causes a callin to be removed from the allowed set.
Lifestate rules specify enabledness and allowedness effects

A suspended callback or callin invocation (thunk)

**enable**

\[ message_1 \rightarrow_{cb} message_2 \]

Message causes a callback to be added to the enabled set

**disable**

\[ message_1 \nrightarrow_{cb} message_2 \]

Message causes a callback to be removed to the enabled set

**allow**

\[ message_1 \rightarrow_{ci} message_2 \]

Message causes a callin to be added to the allowed set

**disallow**

\[ message_1 \nrightarrow_{ci} message_2 \]

Message causes a callin to be removed from the allowed set
Lifestate rules specify enabledness and allowedness effects.

A suspended callback or callin invocation (thunk):

Specify: When $message_1$ is invoked, the effect on the enabled-allowed state is to enable/disable/allow/disallow $message_2$.

**Enable**

$message_1 \rightarrow_{cb} message_2$

Message causes a callback to be added to the enabled set.

**Allow**

$message_1 \rightarrow_{ci} message_2$

Message causes a callin to be added to the allowed set.

**Disable**

$message_1 \rightarrow_{cb} message_2$

Message causes a callback to be removed from the enabled set.

**Disallow**

$message_1 \rightarrow_{ci} message_2$

Message causes a callin to be removed from the allowed set.
Need: Mining lifestate specifications
Need: Mining lifestate specifications

These specifications often don’t exist in the documentation.

An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps, called onPreExecute, doInBackground, onProgressUpdate and onPostExecute.
Need: Mining lifestate specifications

These specifications often don’t exist in the documentation.

An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called \texttt{Params}, \texttt{Progress} and \texttt{Result}, and 4 steps, called \texttt{onPreExecute}, \texttt{doInBackground}, \texttt{onProgressUpdate} and \texttt{onPostExecute}.

The \texttt{AsyncTask} documentation does not fully document the behavior.
Need: Mining lifestate specifications

These specifications often don’t exist in the documentation.

An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called `Params`, `Progress` and `Result`, and 4 steps, called `onPreExecute`, `doInBackground`, `onProgressUpdate` and `onPostExecute`.

The Android Framework is Huge

100s of API packages, 1,000s of API classes, 10,000s+ of API methods (as of API 23)

The `AsyncTask` documentation does not fully document the behavior
Need: Mining lifestate specifications

These specifications often don’t exist in the documentation.

An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called `Params`, `Progress` and `Result`, and 4 steps, called `onPreExecute`, `doInBackground`, `onProgressUpdate` and `onPostExecute`.

The Android Framework is Huge

- 100s of API packages,
- 1,000s of API classes,
- 10,000s+ of API methods (as of API 23)

Writing specifications by hand is error prone

The `AsyncTask` documentation does not fully document the behavior
Need: Mining lifestate specifications

These specifications often don’t exist in the documentation.

The Android Framework is Huge

100s of API packages, 1,000s of API classes, 10,000s+ of API methods (as of API 23)

An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread. An asynchronous task is defined by 3 generic types, called `Params`, `Progress` and `Result`, and 4 steps, called `onPreExecute`, `doInBackground`, `onProgressUpdate` and `onPostExecute`.

The `AsyncTask` documentation does not fully document the behavior.

Task: **Mine** lifestate specifications of the Android framework from large corpus of actual apps interacting with the framework based on the λlife model
Learn lifestate rules that explain observed traces
Generate Traces

Learn lifestate rules that explain observed traces
Learn lifestate rules that explain observed traces
Learn lifestate rules that explain observed traces

Generate Traces

Slice Traces

Learn Rules (unsupervised)

AsyncTask Specification

Button Specification
Learn lifestate rules that explain observed traces

Generate Traces

Slice Traces

Traces

AsyncTask

Learn Rules (unsupervised)

AsyncTask Specification

Traces

Learn automata models (hidden Markov models [HMMs], probabilistic finite state automata [PFSAs]).

Abstract automata into lifestate rules.
Learn lifestate rules that explain observed traces.

Generate Traces → Slice Traces → Traces

Direct rule learning: maximum likelihood rules via model counting (MC) using \#SAT

Learn automata models (hidden Markov models [HMMs], probabilistic finite state automata [PFSAs]).
Abstract automata into lifestate rules.

AsyncTask Specification
Learn lifestate rules that explain observed traces

Generate Traces

Slice Traces

Direct rule learning: maximum likelihood rules via model counting (MC) using #SAT

Learn Rules (unsupervised)

AsyncTask Specification

Learn automata models (hidden Markov models [HMMs], probabilistic finite state automata [PFSAs]). Abstract automata into lifestate rules.

A portfolio of approaches to learn candidate specifications
Evaluating lifestate specification mining
Evaluating lifestate specification mining

Do we learn rules that correspond to actual Android behavior?
Evaluating lifestate specification mining

Do we learn rules that correspond to actual Android behavior?

4 Android framework classes
Evaluating lifestate specification mining

Do we learn rules that correspond to actual Android behavior?

4 Android framework classes
7326 possible specification rules
Evaluating lifestate specification mining

Do we learn rules that correspond to actual Android behavior?

4 Android framework classes
7326 possible specification rules

Found 82 rules corresponding to actual Android behavior by examining 163 rules
Evaluating lifestate specification mining

Do we learn rules that correspond to actual Android behavior?

4 Android framework classes
7326 possible specification rules

Found 82 rules corresponding to actual Android behavior by examining 163 rules

Found actual rules in under-constrained search space. Discovered undocumented rules.
Evaluating lifestate specification mining

Do we learn rules that correspond to actual Android behavior?

4 Android framework classes

7326 possible specification rules

Found 82 rules corresponding to actual Android behavior by examining 163 rules

Found actual rules in under-constrained search space. Discovered undocumented rules.
Evaluating lifestate specification mining

Do we learn rules that correspond to actual Android behavior?

4 Android framework classes
7326 possible specification rules

Found 82 rules corresponding to actual Android behavior by examining 163 rules

A developer gets a crash by misusing `Fragment.getResources` and then looks for the following documentation:

```
public final Resources getResources ()
```

Returns

```
Resources
```

Found actual rules in under-constrained search space. Discovered undocumented rules.
Evaluating lifestate specification mining

Do we learn rules that correspond to actual Android behavior?

4 Android framework classes
7326 possible specification rules
Found 82 rules corresponding to actual Android behavior by examining 163 rules

A developer gets a crash by misusing Fragment.getResources and then looks for the following documentation

Found actual rules in under-constrained search space. Discovered undocumented rules.
Evaluating lifestate specification mining

Do we learn rules that correspond to actual Android behavior?

4 Android framework classes
7326 possible specification rules

Found 82 rules corresponding to actual Android behavior by examining 163 rules

Found actual rules in under-constrained search space. Discovered undocumented rules.

A developer gets a crash by misusing `Fragment.getResources` and then looks for the following documentation
Conclusion
Hopper: Prove safety properties in event-driven applications by soundly jumping between callbacks
Conclusion

**Hopper**: Prove safety properties in event-driven applications by soundly jumping between callbacks

**DroidLife**: Mine lifestate models of how callins and callbacks affect each other in Android
**Hopper**: Prove safety properties in event-driven applications by soundly jumping between callbacks

**DroidLife**: Mine lifestate models of how callins and callbacks affect each other in Android

www.cs.colorado.edu/~bec
plv.colorado.edu