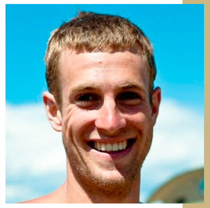
Analyzing, Abstracting, and Mining Event-Driven Systems


Sam Blackshear
Facebook


Shawn Meier


Maxwell Russek


Aleksandar Chakarov


Sergio Mover

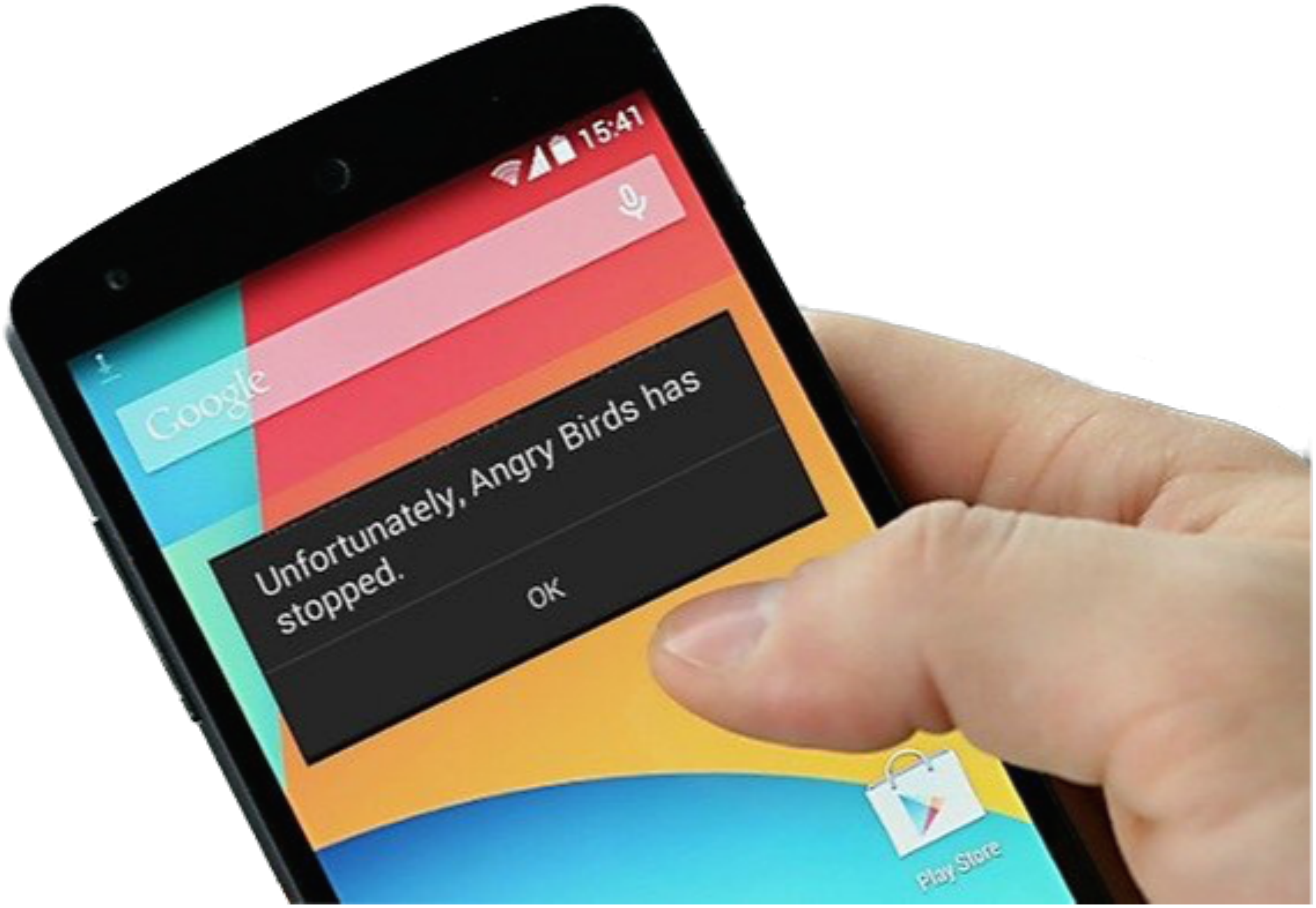

Manu Sridharan
Samsung Research


Bor-Yuh Evan Chang

CUPLV

University of Colorado Boulder

ETH Zürich
October 8, 2016

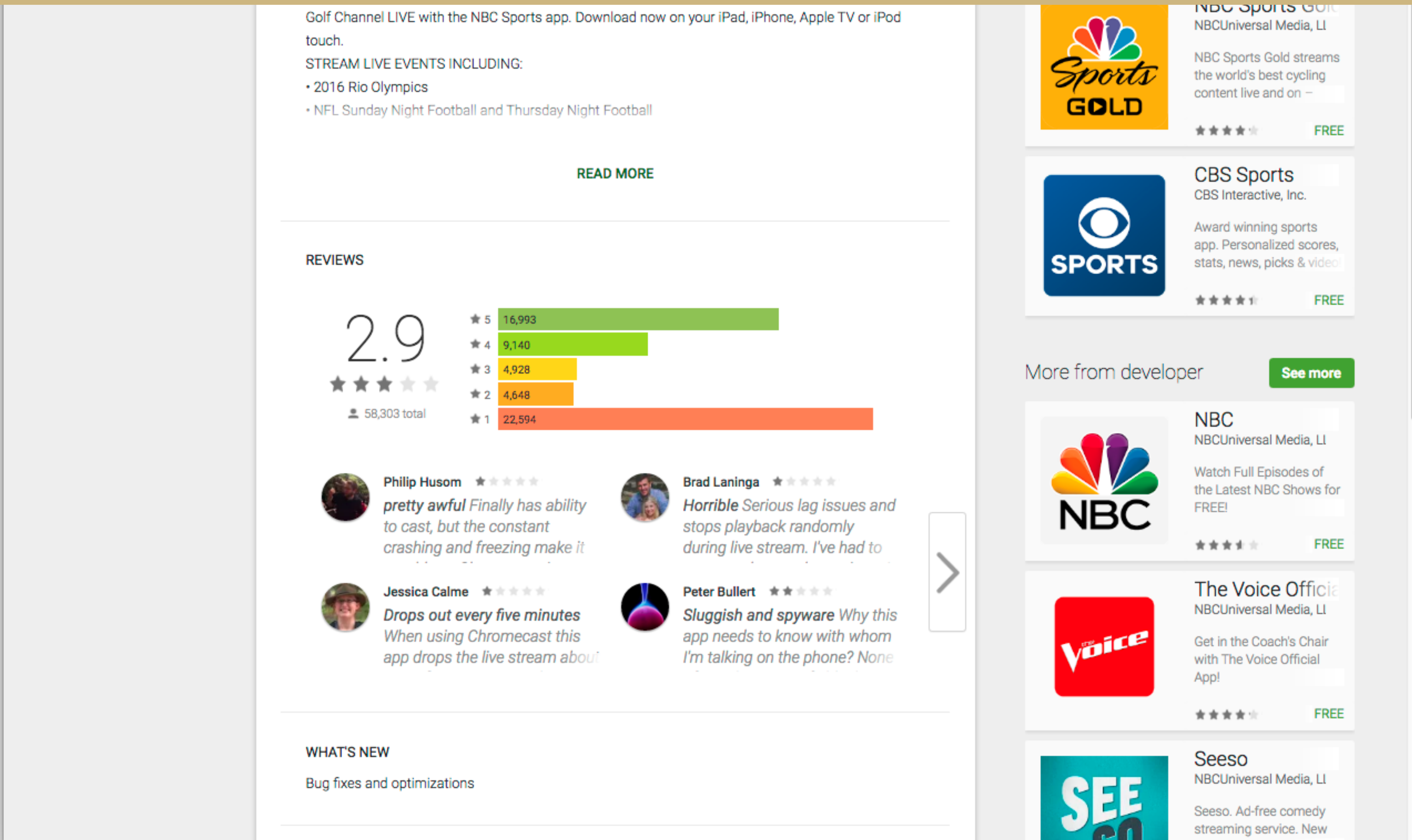

15:41
Google
Unfortunately, Angry Birds has stopped.
OK
Play Store


15:41
Google
Unfortunately, Angry Birds has stopped.
OK
Play Store
Crash

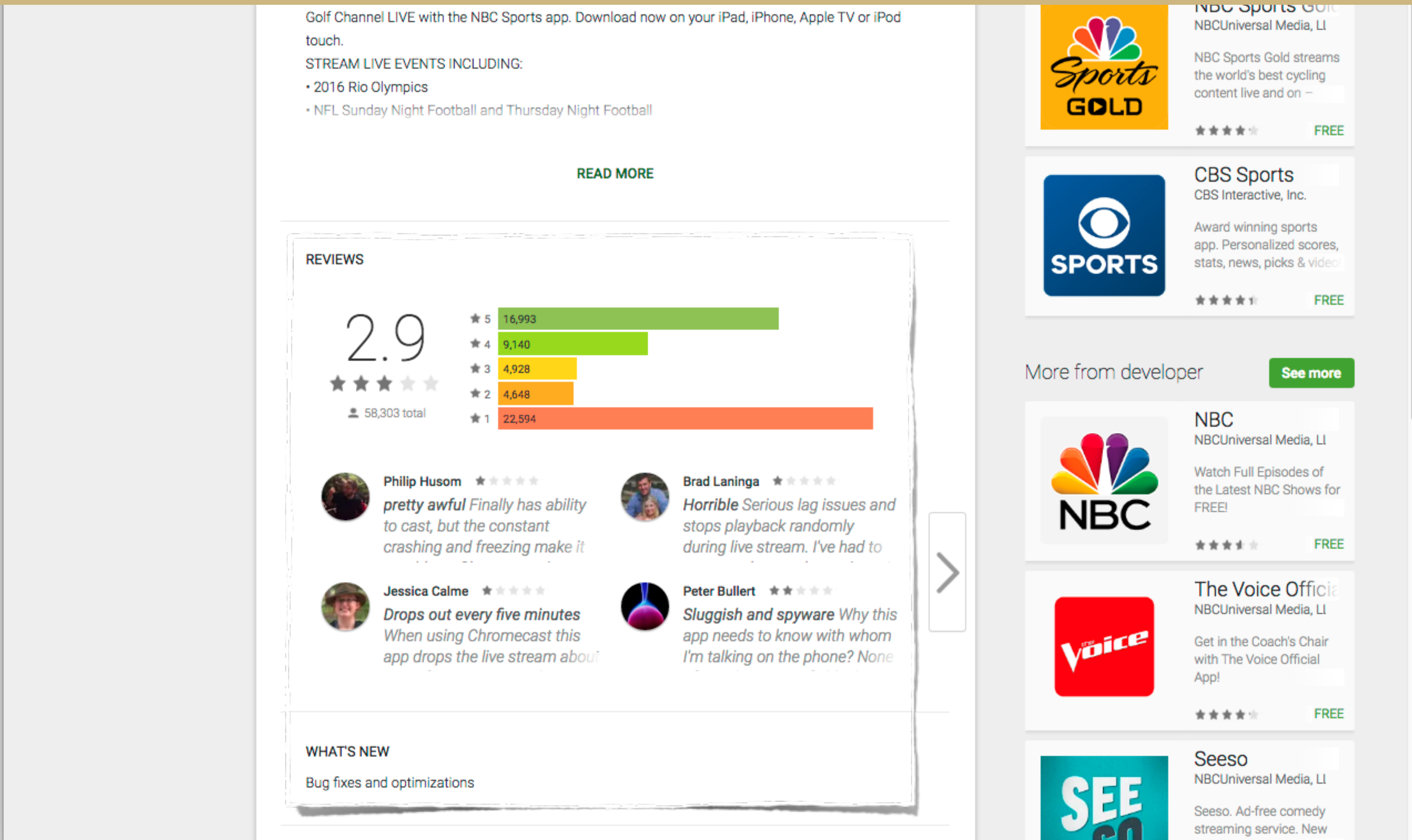

15:41
Google
Unfortunately, Angry Birds has stopped.
OK
Play Store
Crash

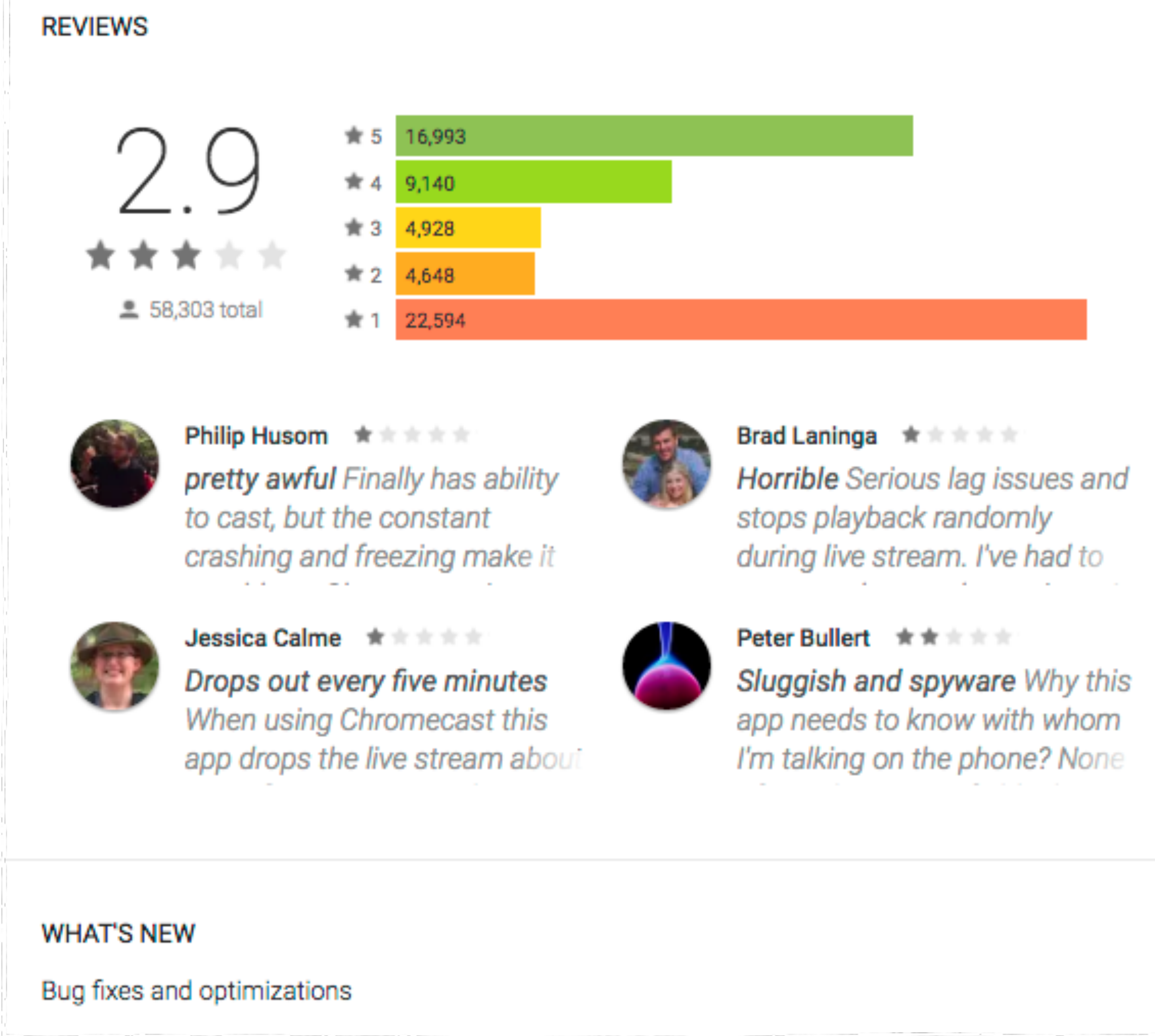
A crash is magnified by the crowd

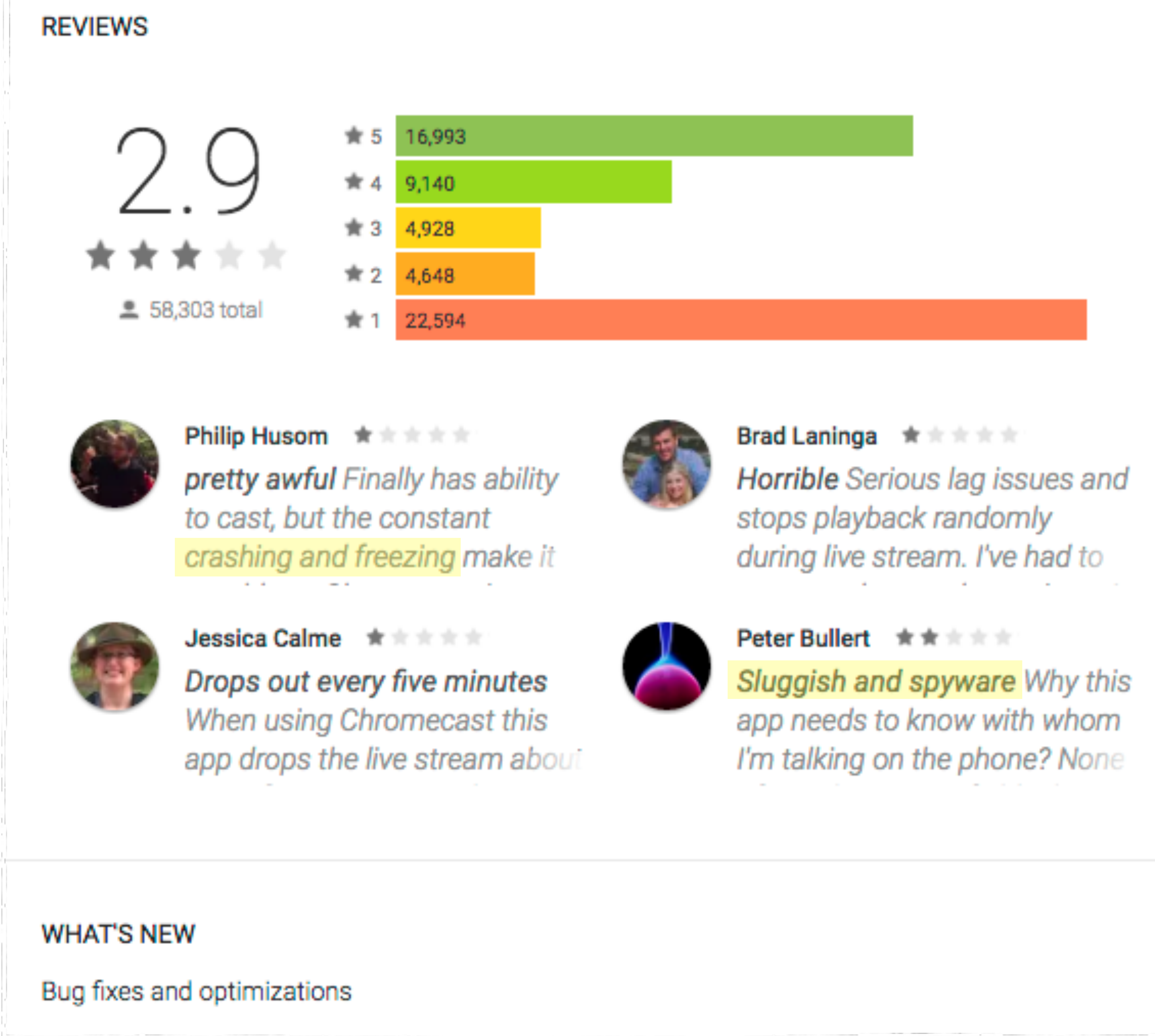
A crash is magnified by the crowd



A crash is magnified by the crowd




REVIEWS
2.9
58,303 total
5
16,993
4
9,140
3
4,928
2
4,648
1
22,594
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
58,303 total
5 16,993
4 9,140
3 4,928
2 4,648
1 22,594
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

58,303 total

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

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

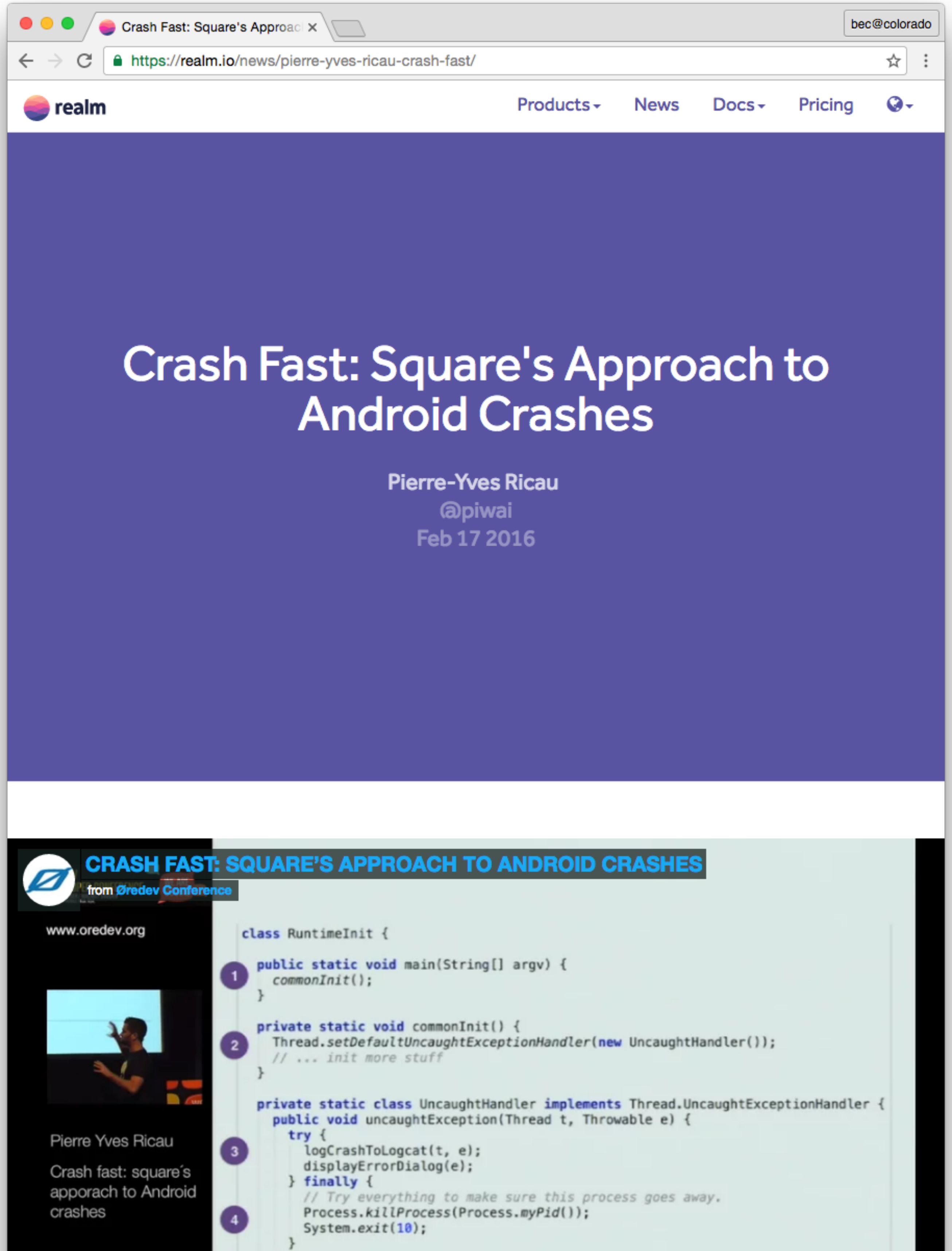
Jessica Calme
Drops out every five minutes When using Chromecast this app drops the live stream about

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

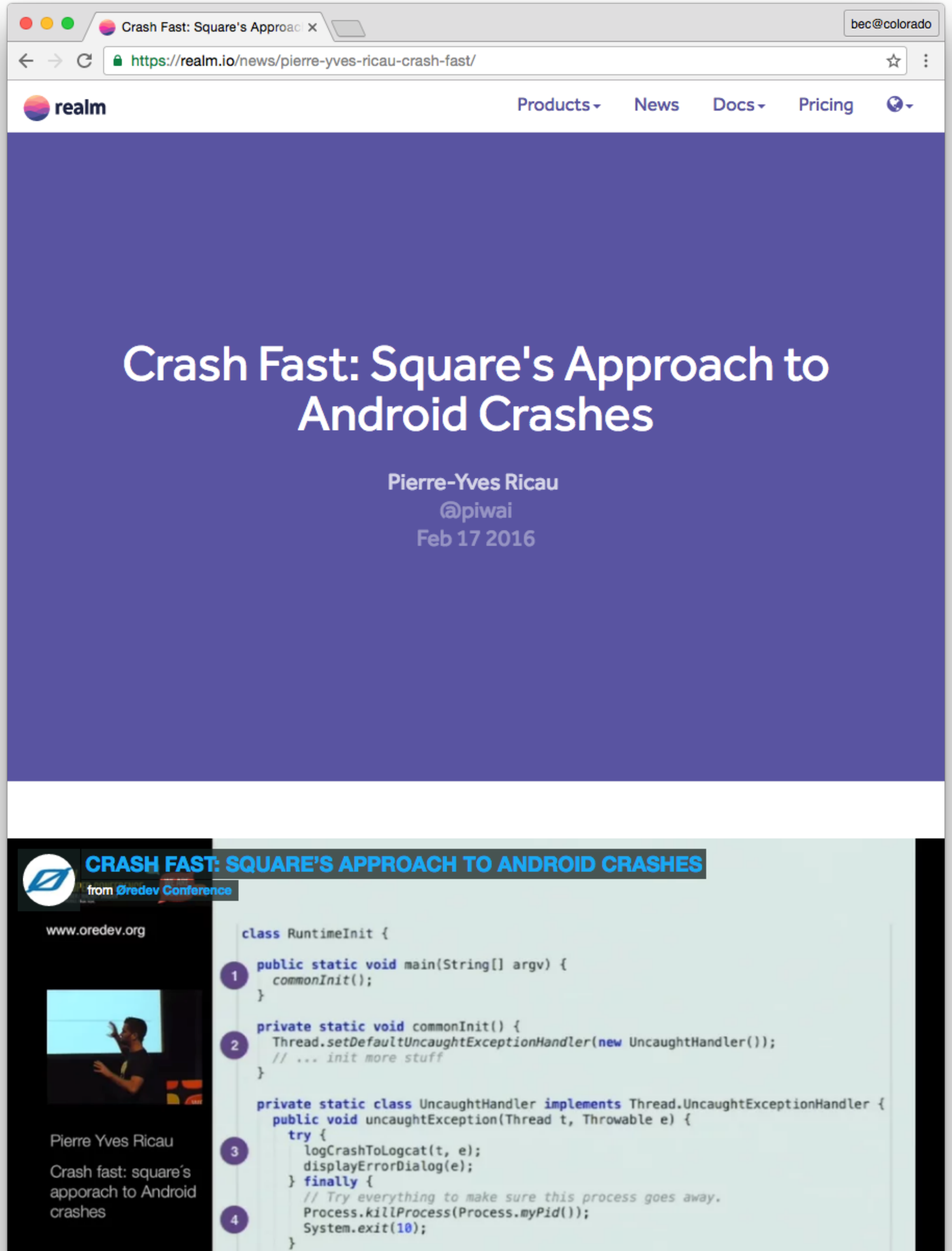
WHAT'S NEW

Bug fixes and optimizations

Ask the expert developers …

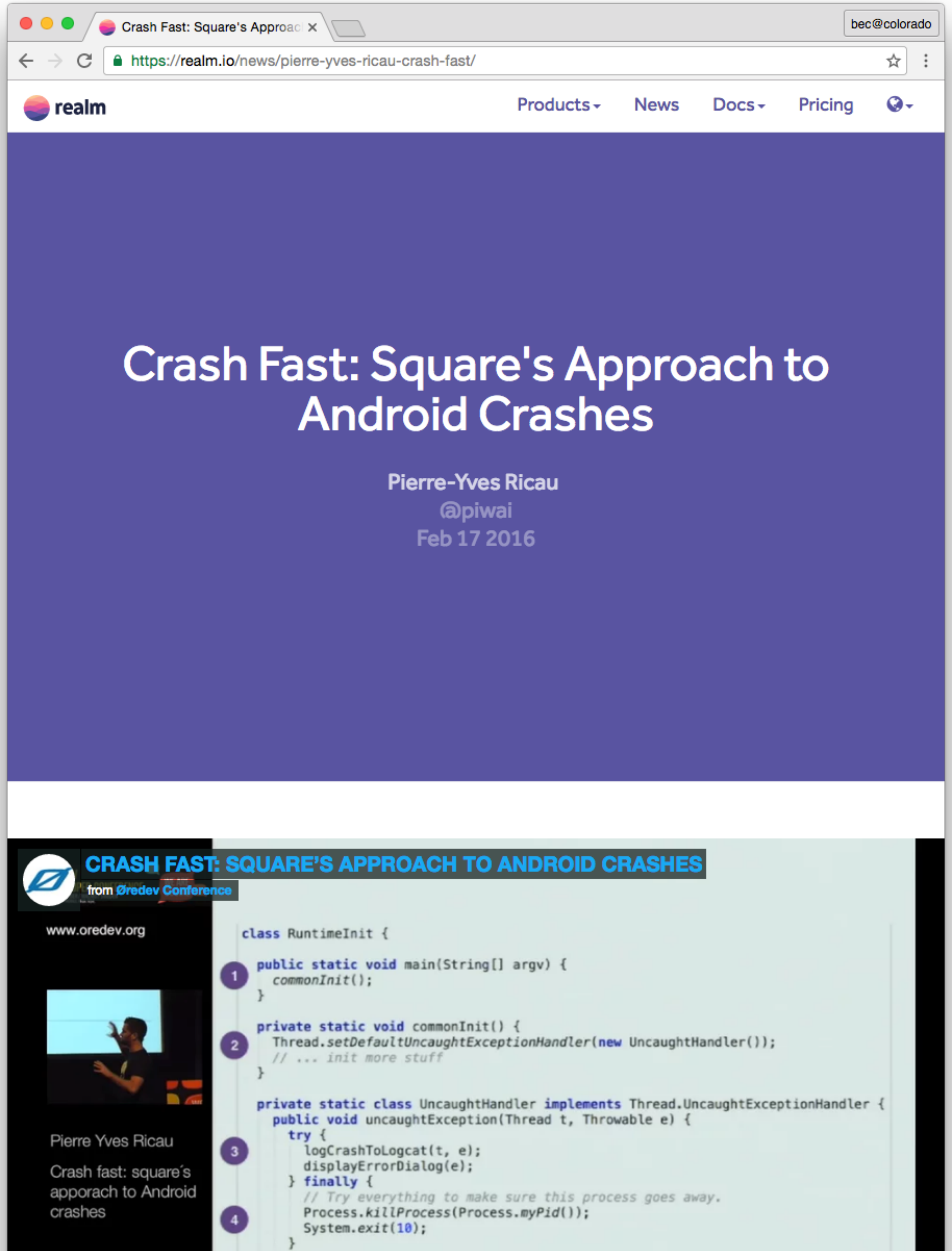


Ask the expert developers …



Inevitable. Not necessarily app's fault.

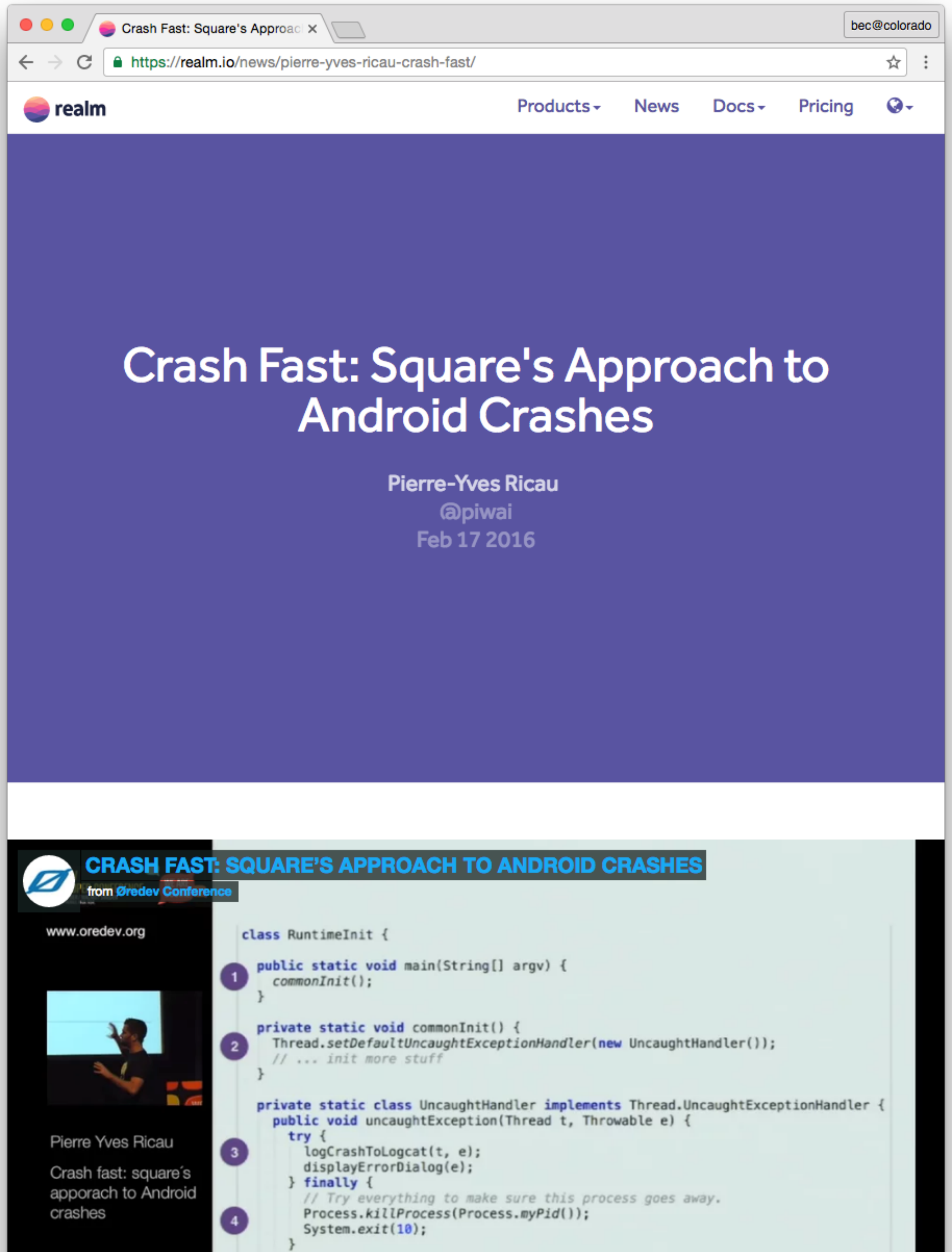
Ask the expert developers …



Inevitable. Not necessarily app's fault.

Log information when it happens.

Ask the expert developers …

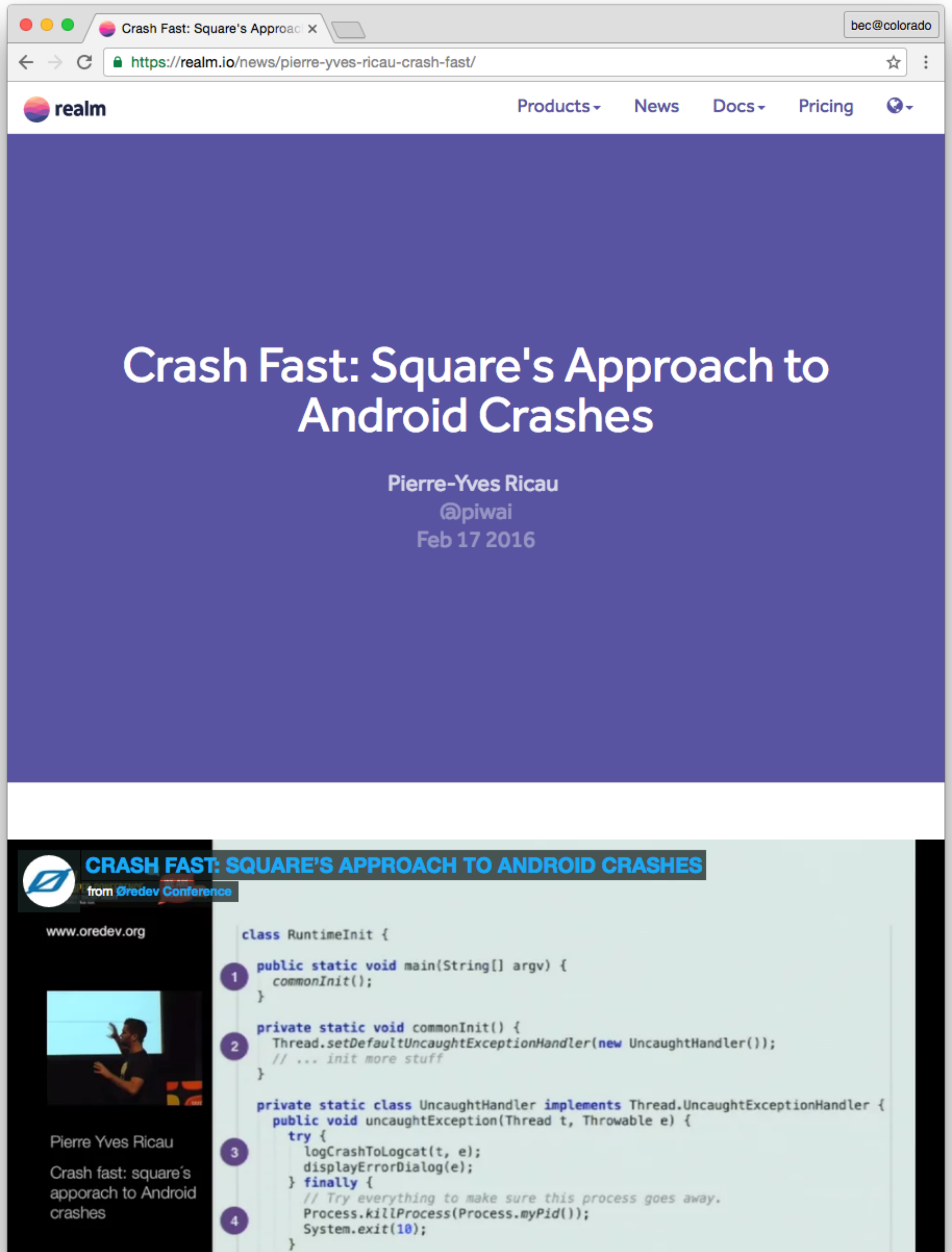


Inevitable. Not necessarily app's fault.

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.

Crash Fast: Square's Approac
https://realm.io/news/pierre-yves-ricau-crash-fast/
realm
Products
News
Docs
Pricing
Crash Fast: Square's Approach to Android Crashes
Pierre-Yves Ricau
@piwai
Feb 17 2016
CRASH FAST: SQUARE'S APPROACH TO ANDROID CRASHES
from Øredev Conference
www.oredev.org
Pierre Yves Ricau
Crash fast: square's apporach to Android crashes
class RuntimeInit {
public static void main(String[] argv) {
commonInit();
}
private static void commonInit() {
Thread.setDefaultUncaughtExceptionHandler(new UncaughtHandler());
// ... init more stuff
}
private static class UncaughtHandler 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(10);
}
}
HD vimeo

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**.

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**.

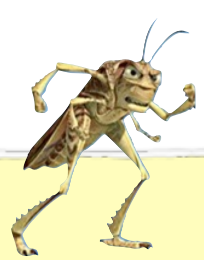
Often: A misunderstanding of how the (sometimes modified) framework interacts with the app.

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

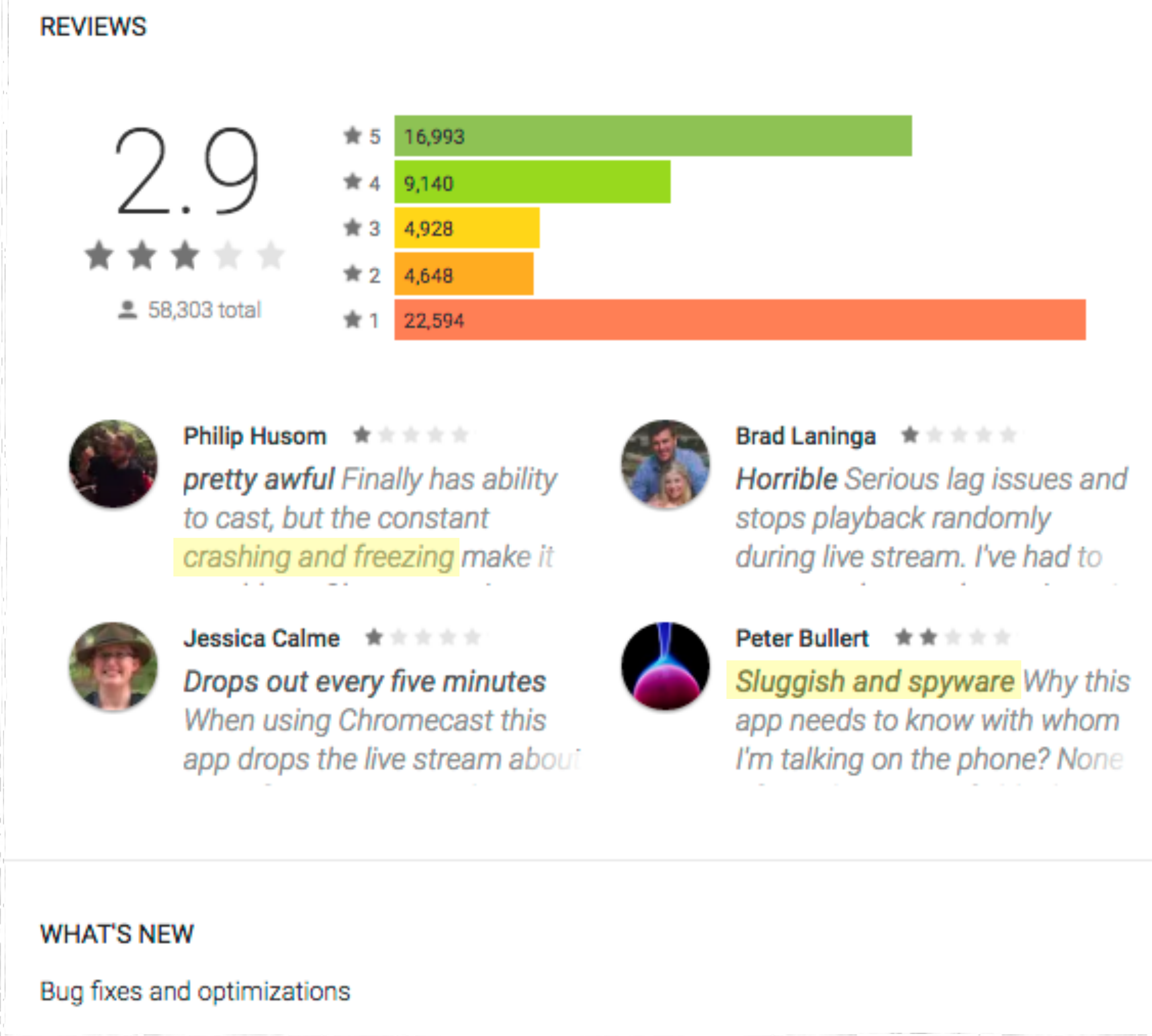
- Where in the code (which callback) does the field get set to **null**?
- Why did that callback happen before this one?
- Is there another callback that should've reset the field to be non-null?

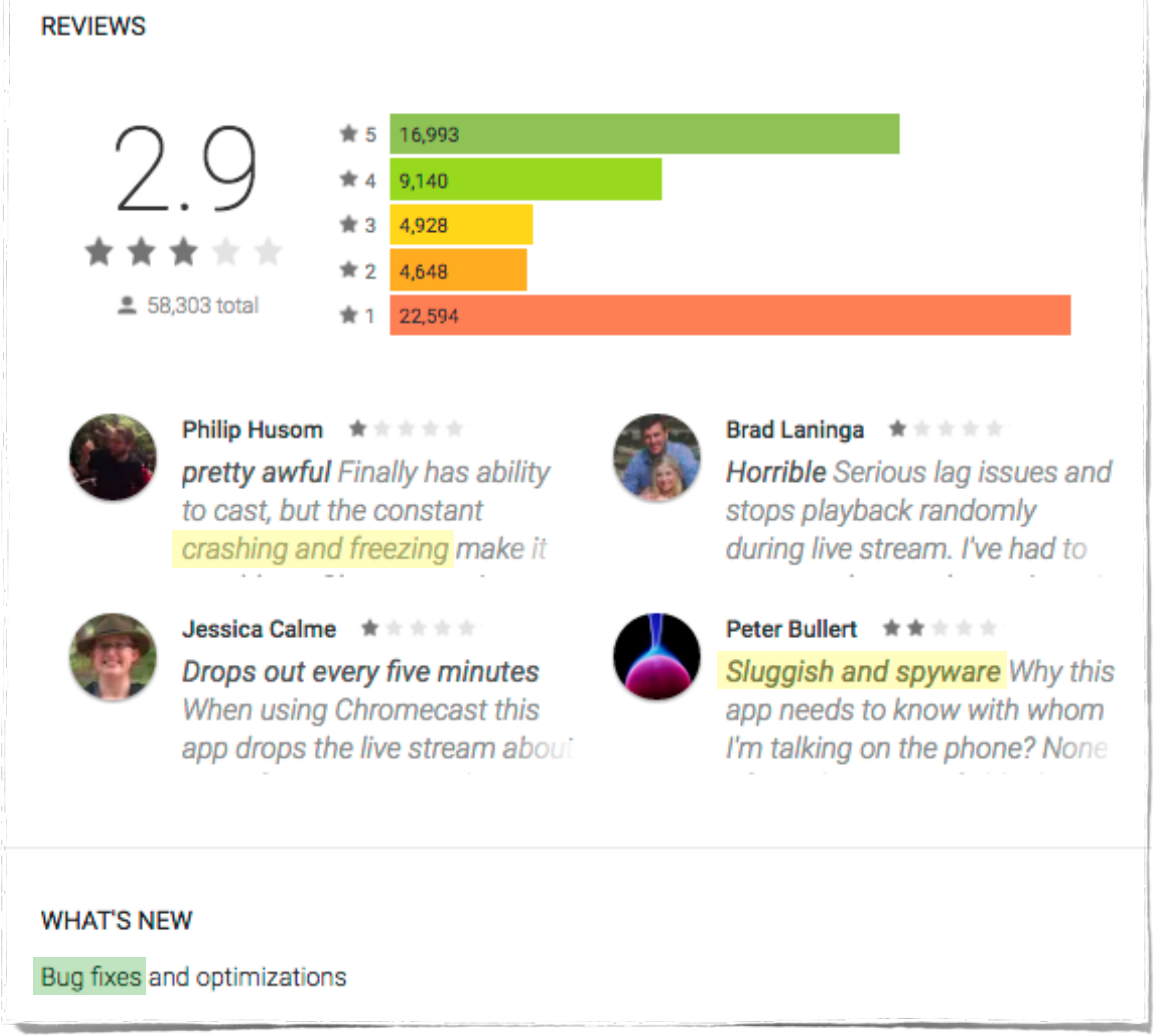
I don't know how that field became **null**.

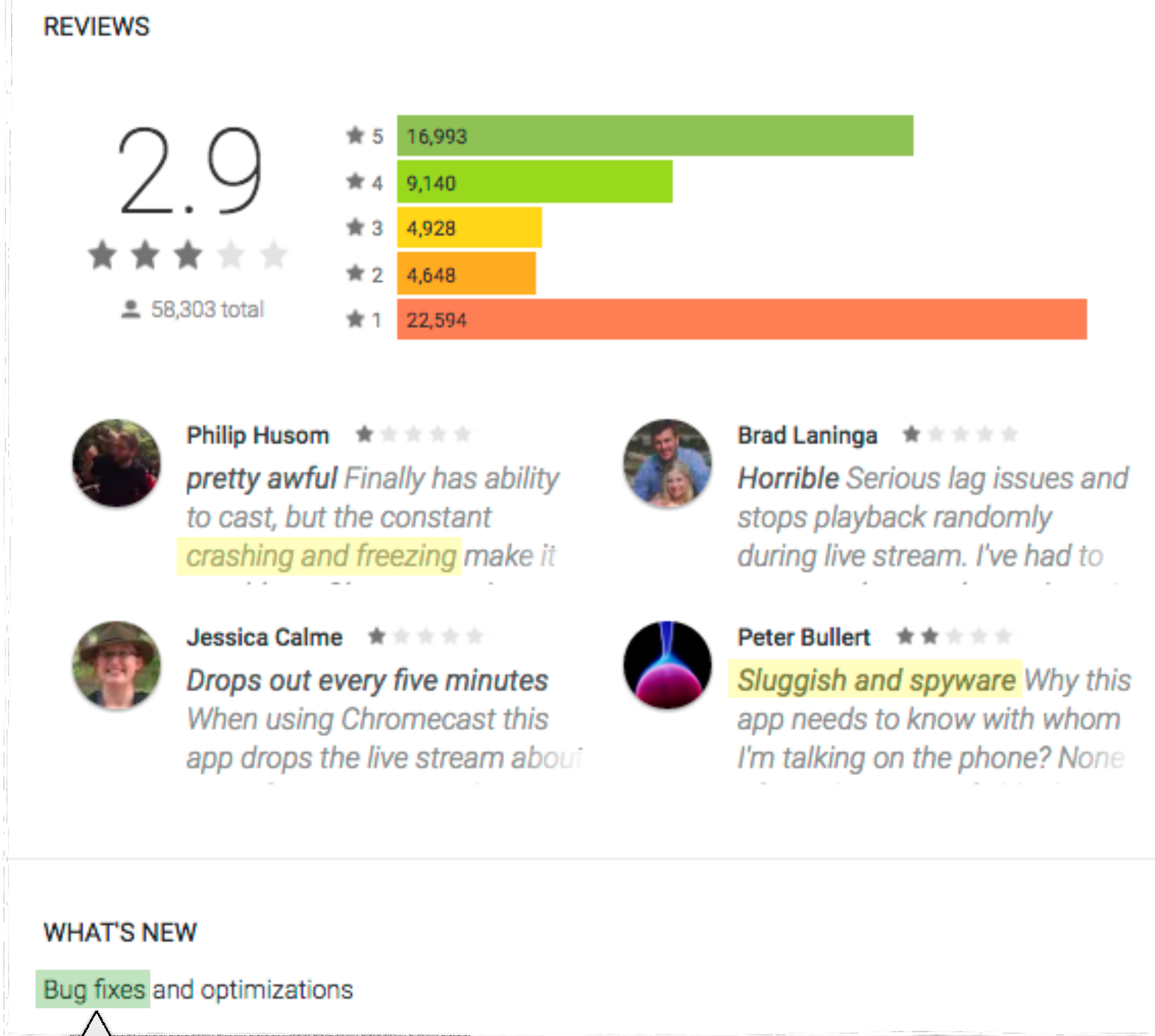
Often: A misunderstanding of how the (sometimes modified) framework interacts with the app.



Bug from violating (implicit) framework protocol rules


REVIEWS
2.9
58,303 total
5
16,993
4
9,140
3
4,928
2
4,648
1
22,594
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
58,303 total
5 16,993
4 9,140
3 4,928
2 4,648
1 22,594
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



opportunity?

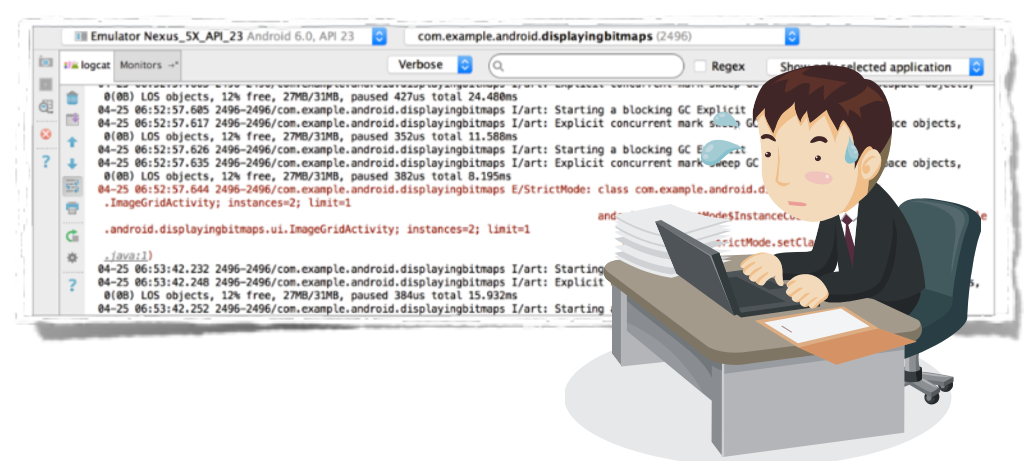
Imagining social programming …

Imagining social programming …

I am not alone

Imagining social programming …

I am not alone



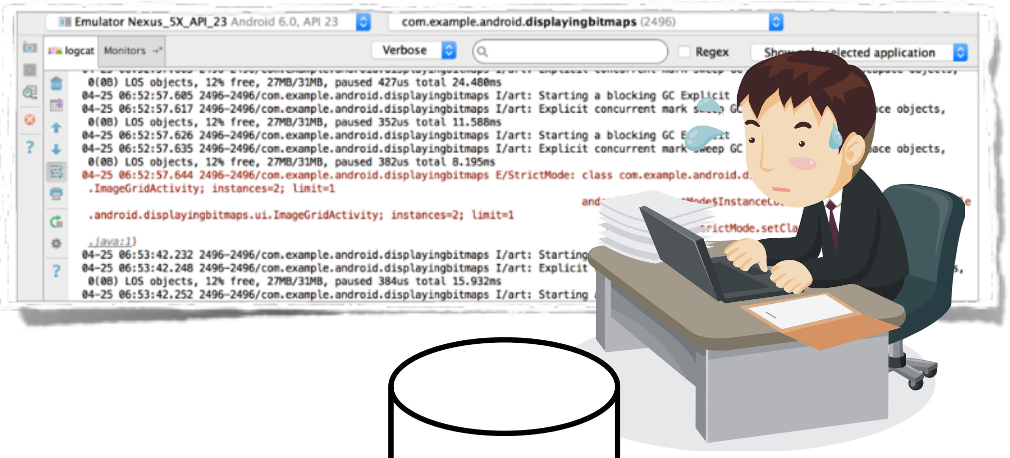
Imagining social programming …

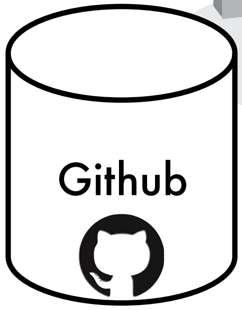
I am not alone

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

Imagining social programming …

I am not alone

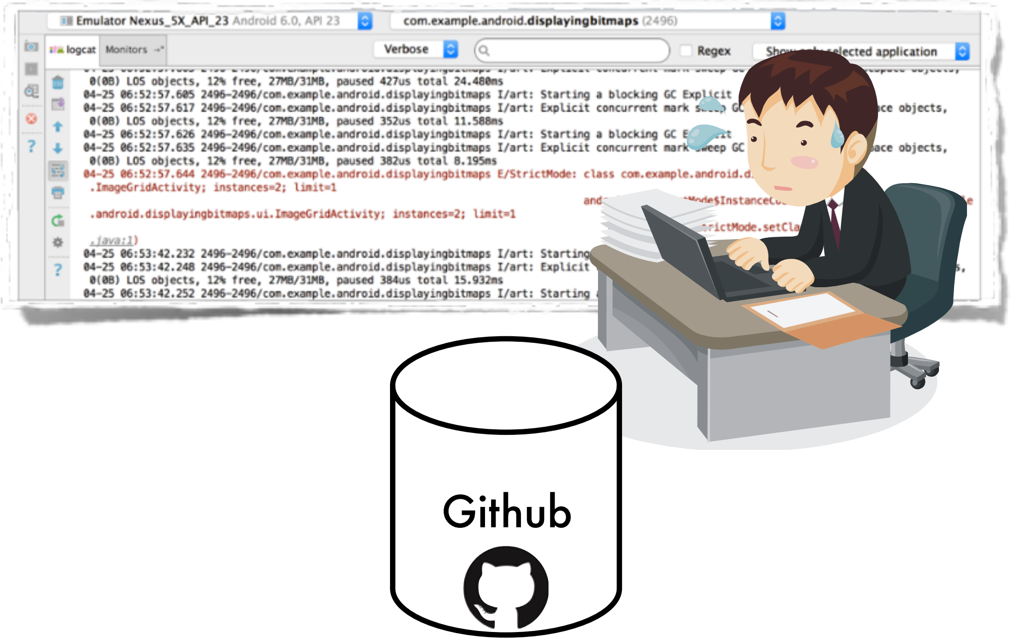




Imagining social programming …

I am not alone

“Transfer” the bug fix with program analysis and synthesis



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

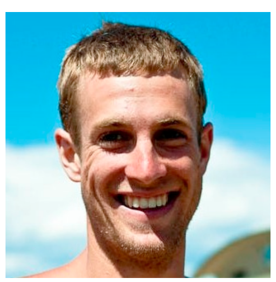
Prove and triage safety properties in event-driven applications (assuming protocol specifications)

Hopper: Goal-Directed Program Analysis with Jumping

Mine
subsequently “

Fixr

Hopper: Goal-Directed Program Analysis with Jumping



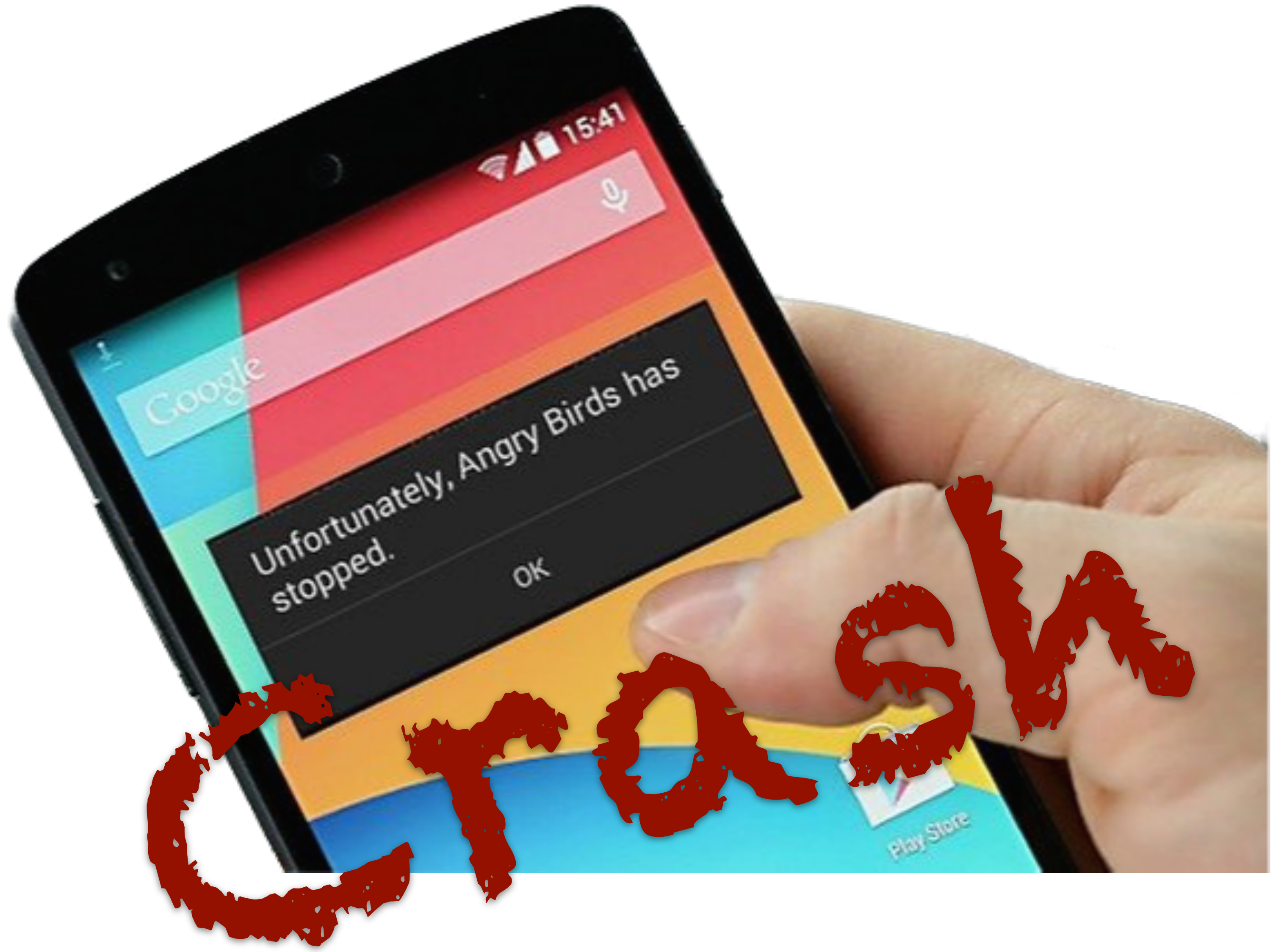
Sam Blackshear
Facebook

Bor-Yuh Evan Chang
University of Colorado Boulder

Manu Sridharan
Samsung Research America

Blackshear, Chang, and Sridharan. “Selective Control-Flow Abstraction via Jumping.” OOPSLA 2015.


15:41
Google
Unfortunately, Angry Birds has stopped.
OK
Play Store
Crash



3% of all commit messages say “NullPointerException”

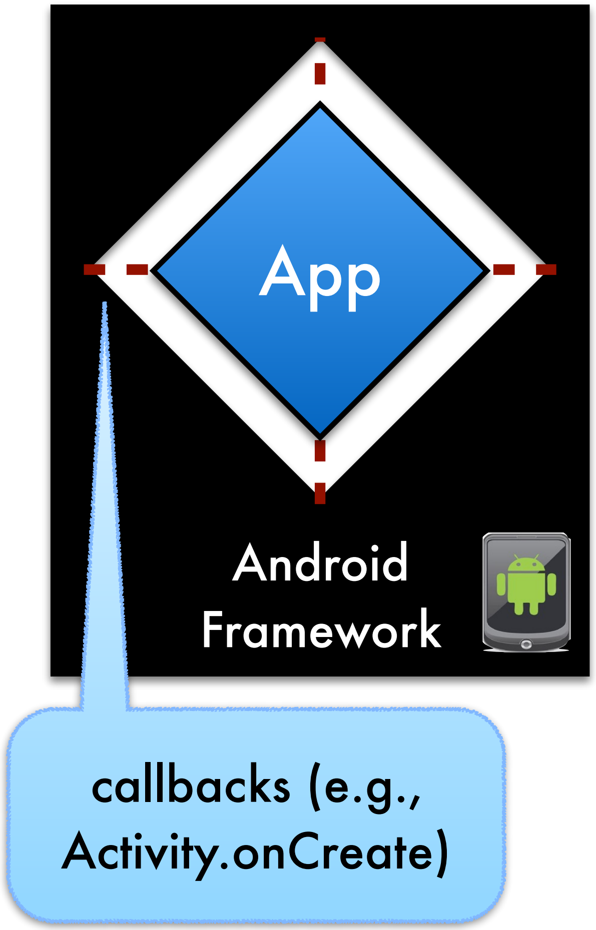
3% of all commit messages say "NullPointerException"

Callback-oriented programming:
A partially ordered "lifecycle"

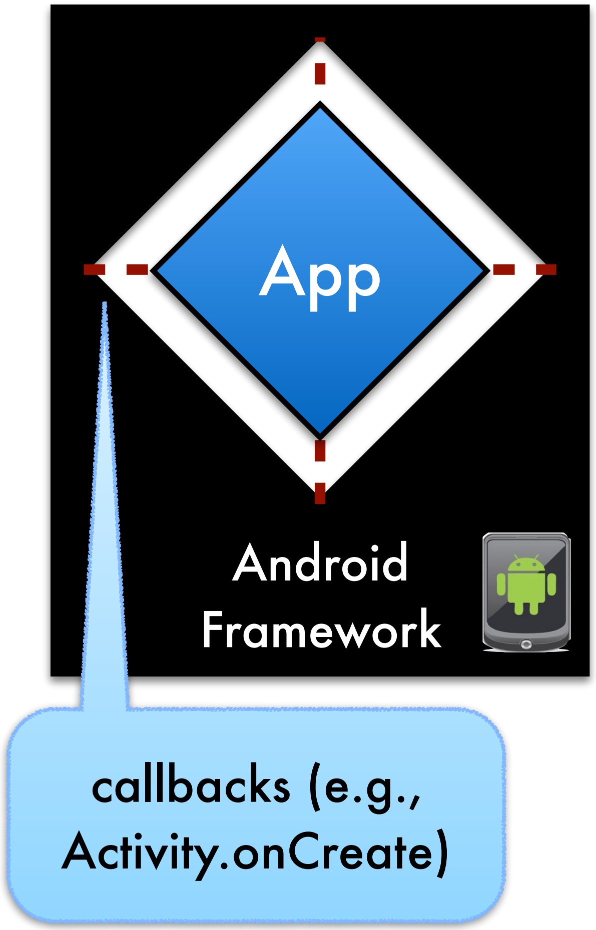
Callback-oriented programming: A partially ordered "lifecycle"

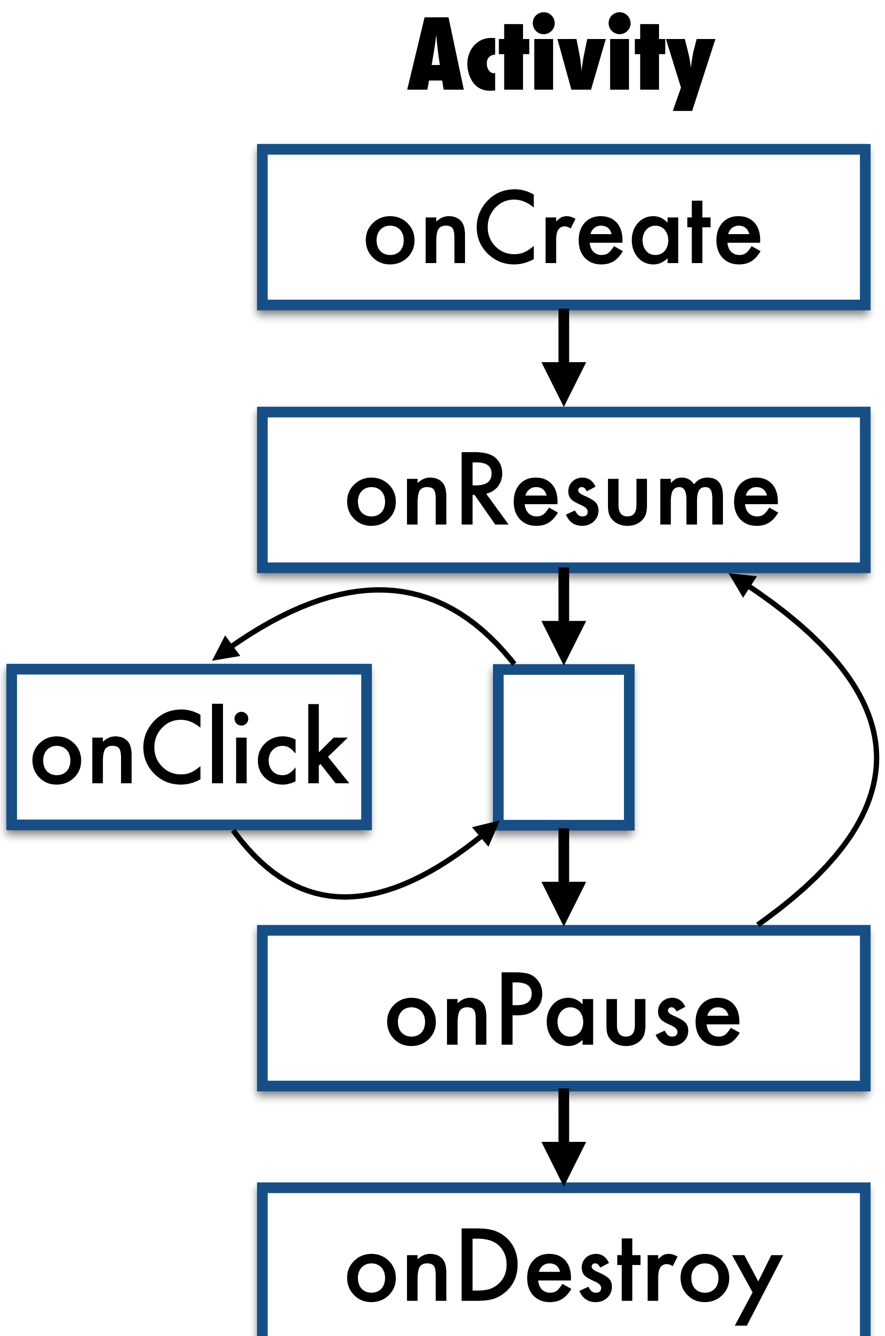
Callback-oriented programming: A partially ordered "lifecycle"

Callback-oriented programming:
A partially ordered "lifecycle"

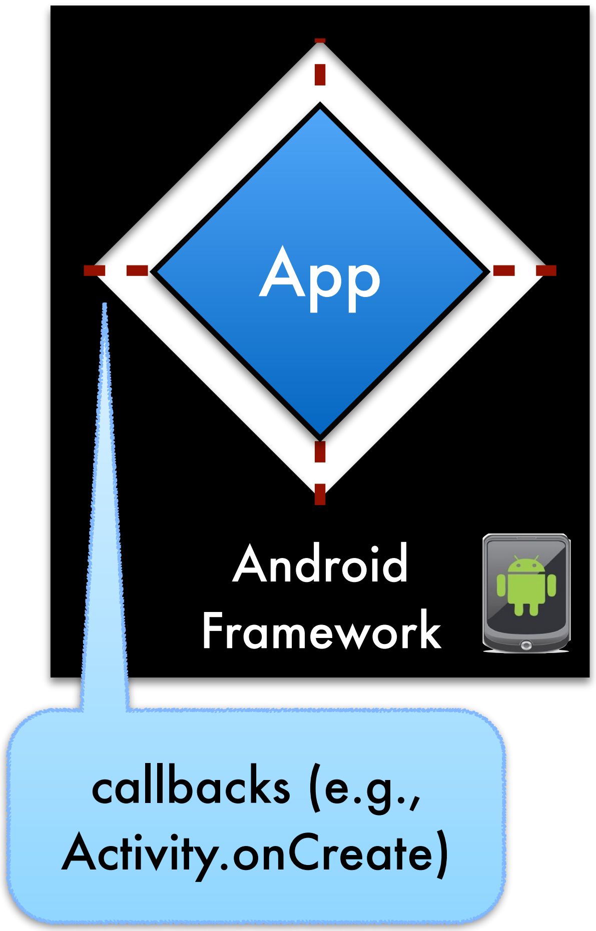


Callback-oriented programming: A partially ordered "lifecycle"

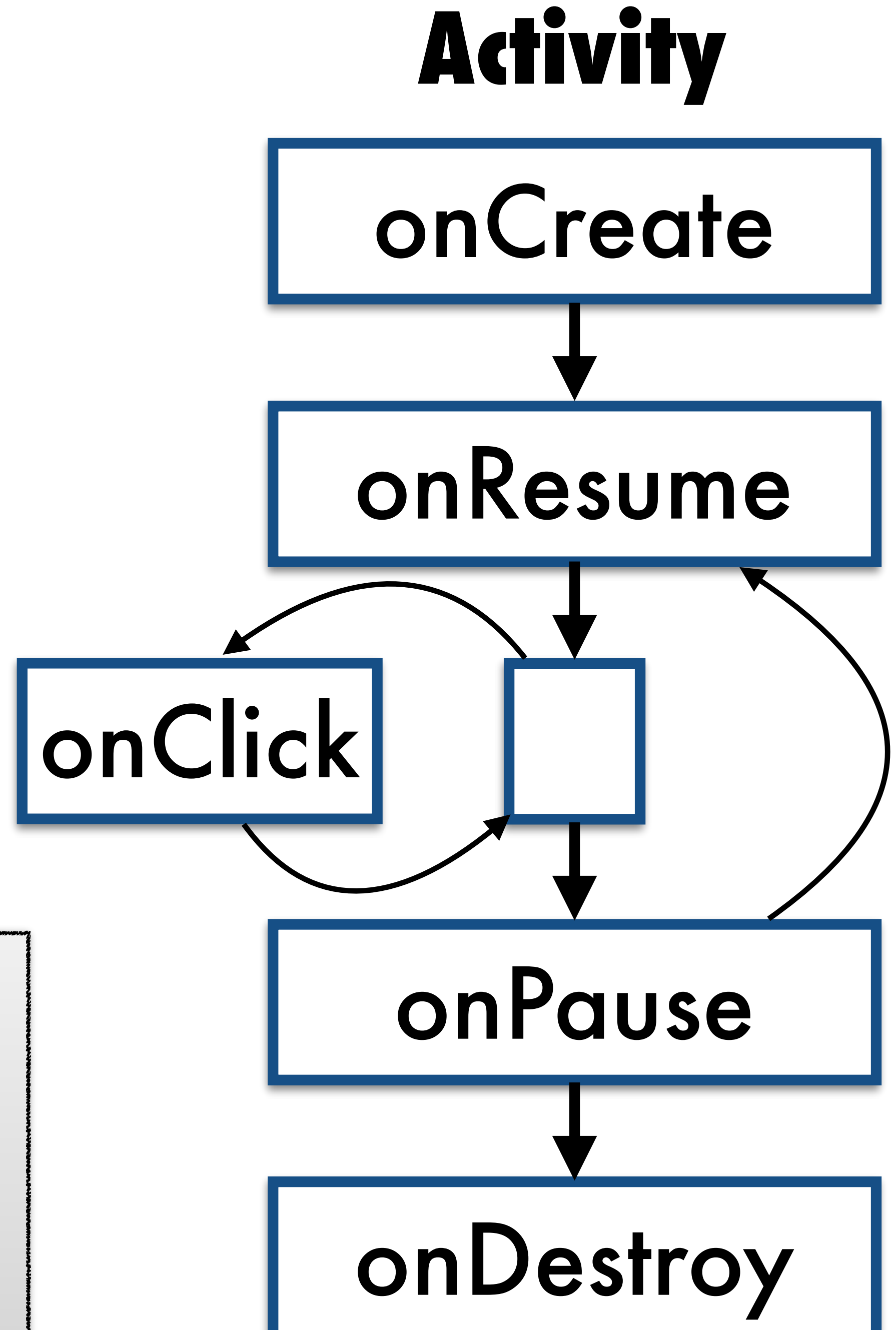




Callback-oriented programming: A partially ordered “lifecycle”

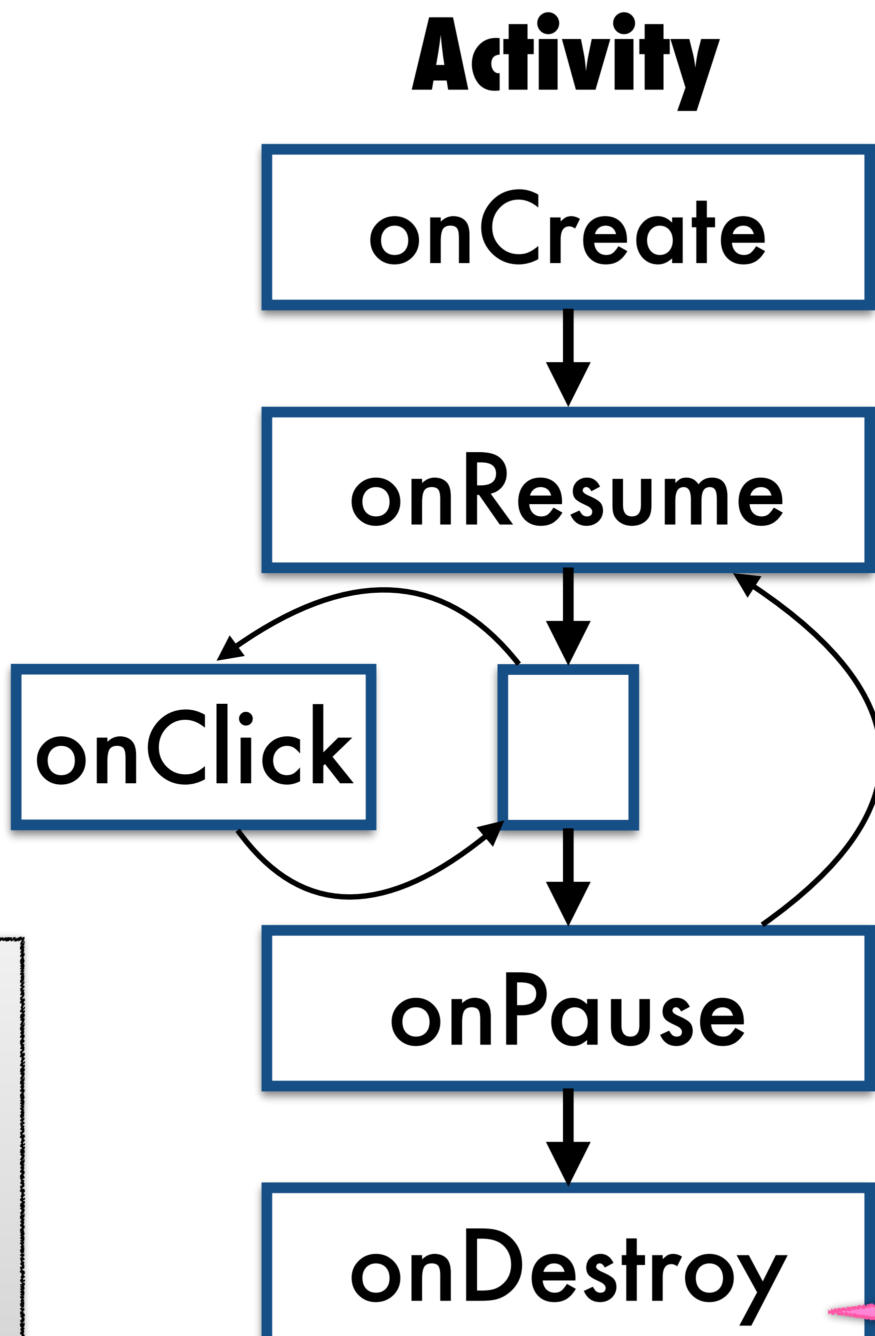


Android components have an ordered lifecycle



Callback-oriented programming:
A partially ordered "lifecycle"

callbacks (e.g., Activity.onCreate)

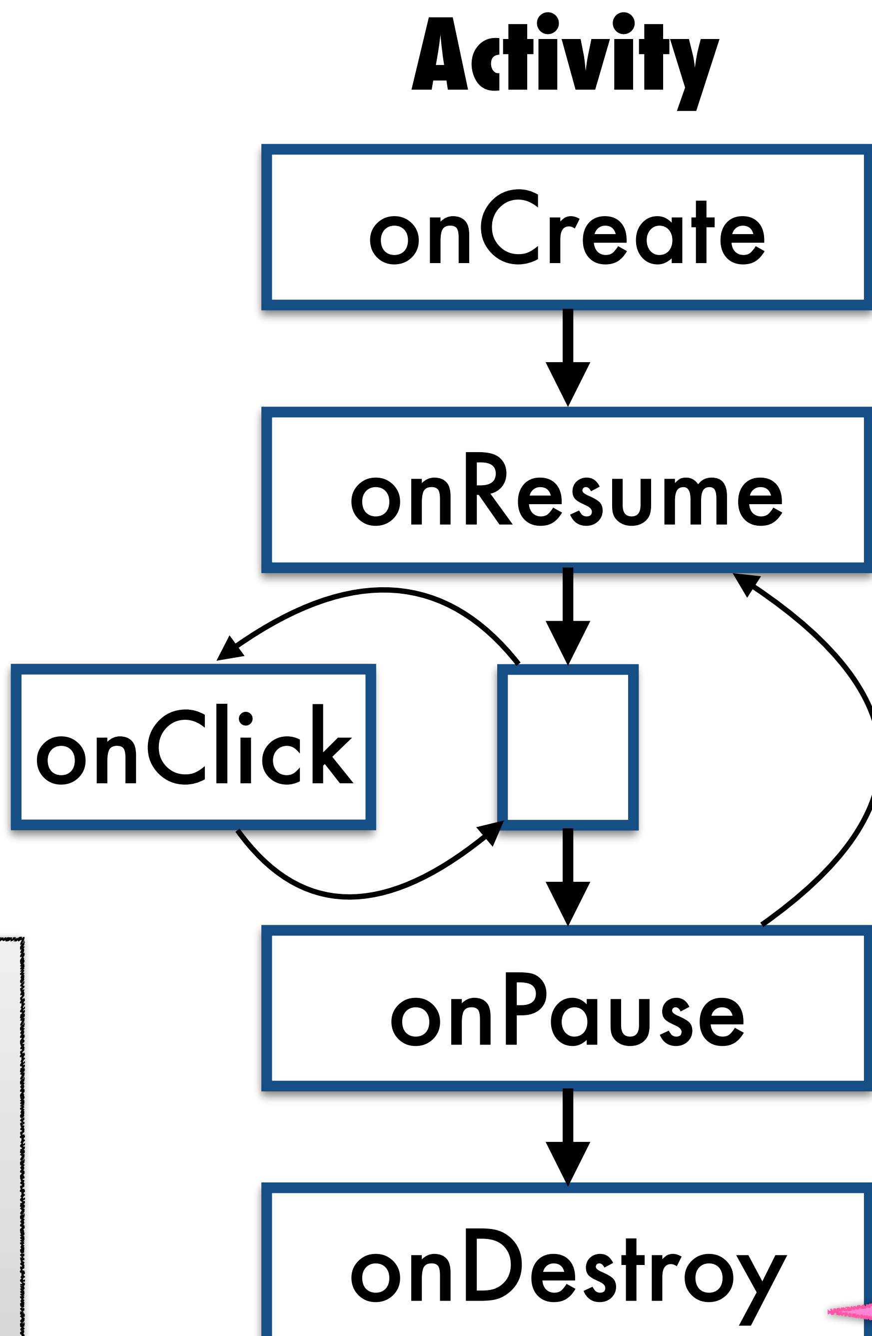



```
this.mHostDb = null;
this.mService = null;
```

Android components have an ordered lifecycle

Callback-oriented programming: A partially ordered "lifecycle"

callbacks (e.g., Activity.onCreate)



Android components have an ordered lifecycle

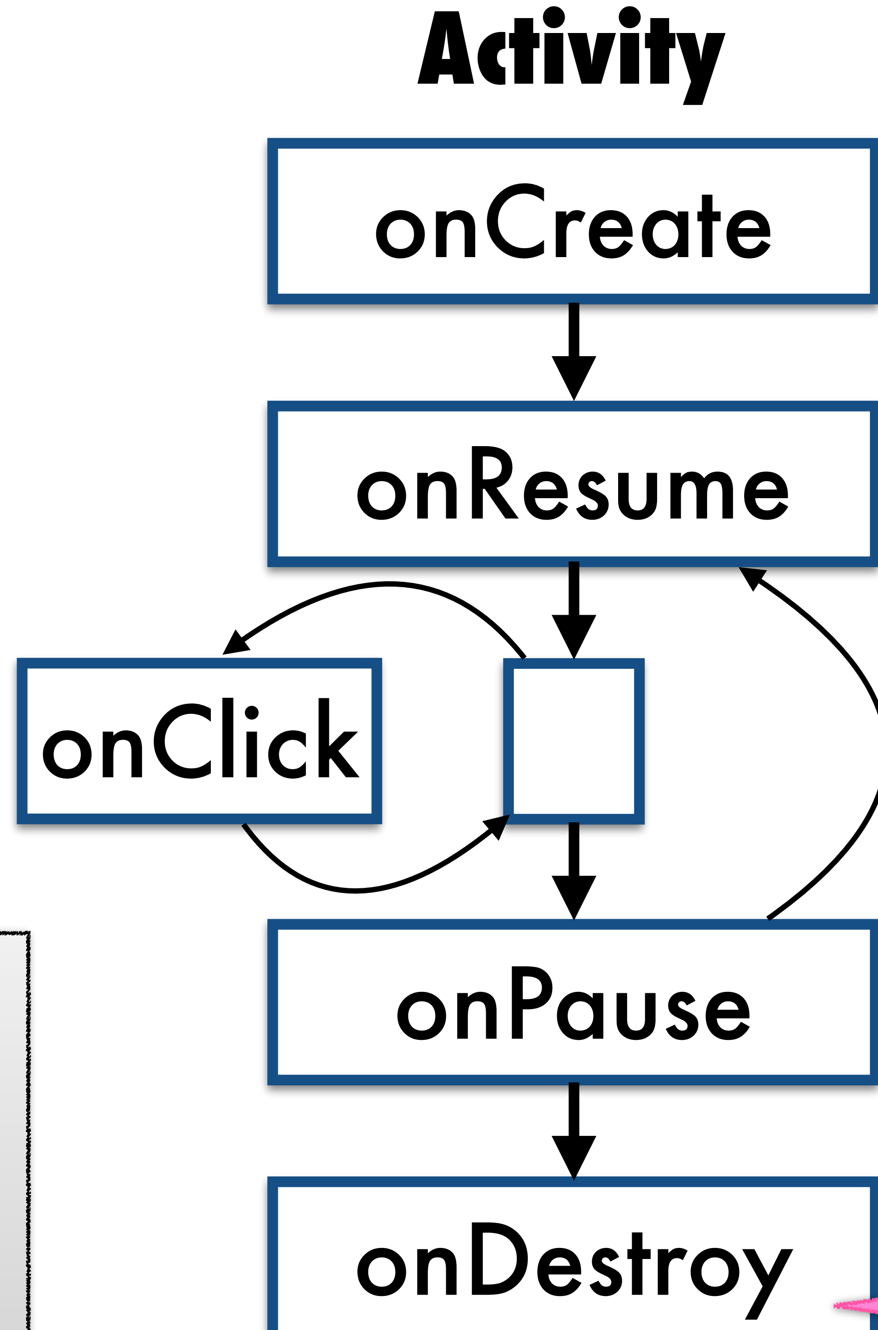
```
this.mHostDb = null;
this.mService = null;
```

Collect resources when done

Callback-oriented programming: A partially ordered "lifecycle"

callbacks (e.g., Activity.onCreate)

Android components have an ordered lifecycle



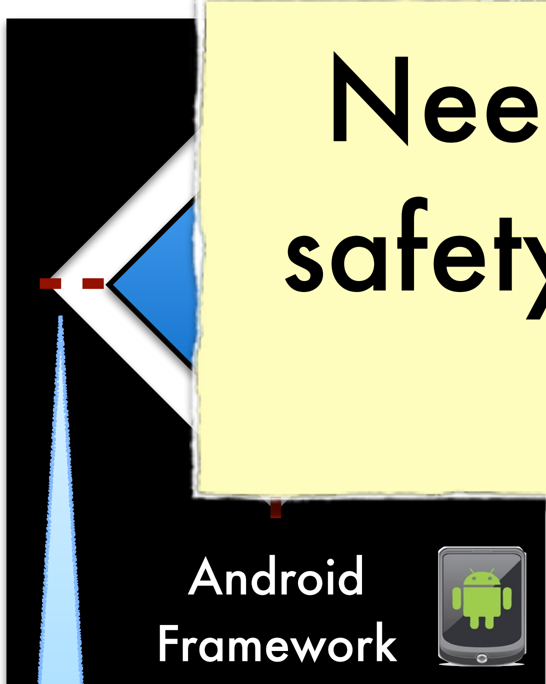
But, lifecycles of different components and other callbacks can interleave …

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

Collect resources when done

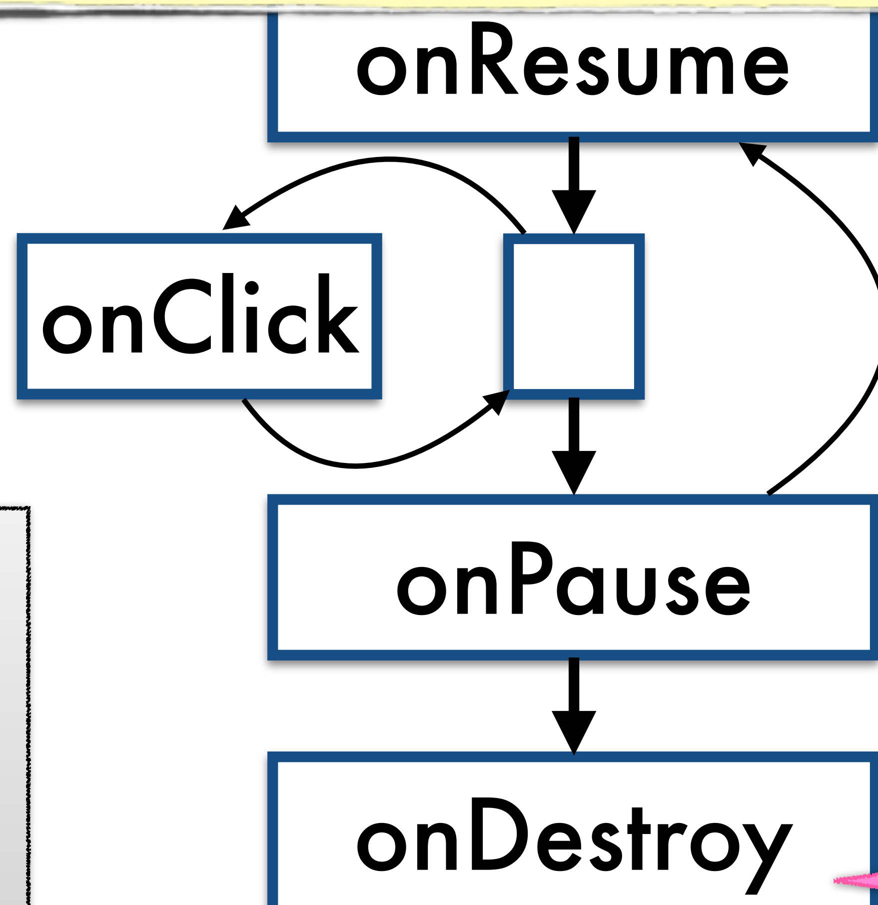
Callback-oriented programming: A partially ordered "lifecycle"

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



callbacks (e.g., Activity.onCreate)

Android components have an ordered lifecycle

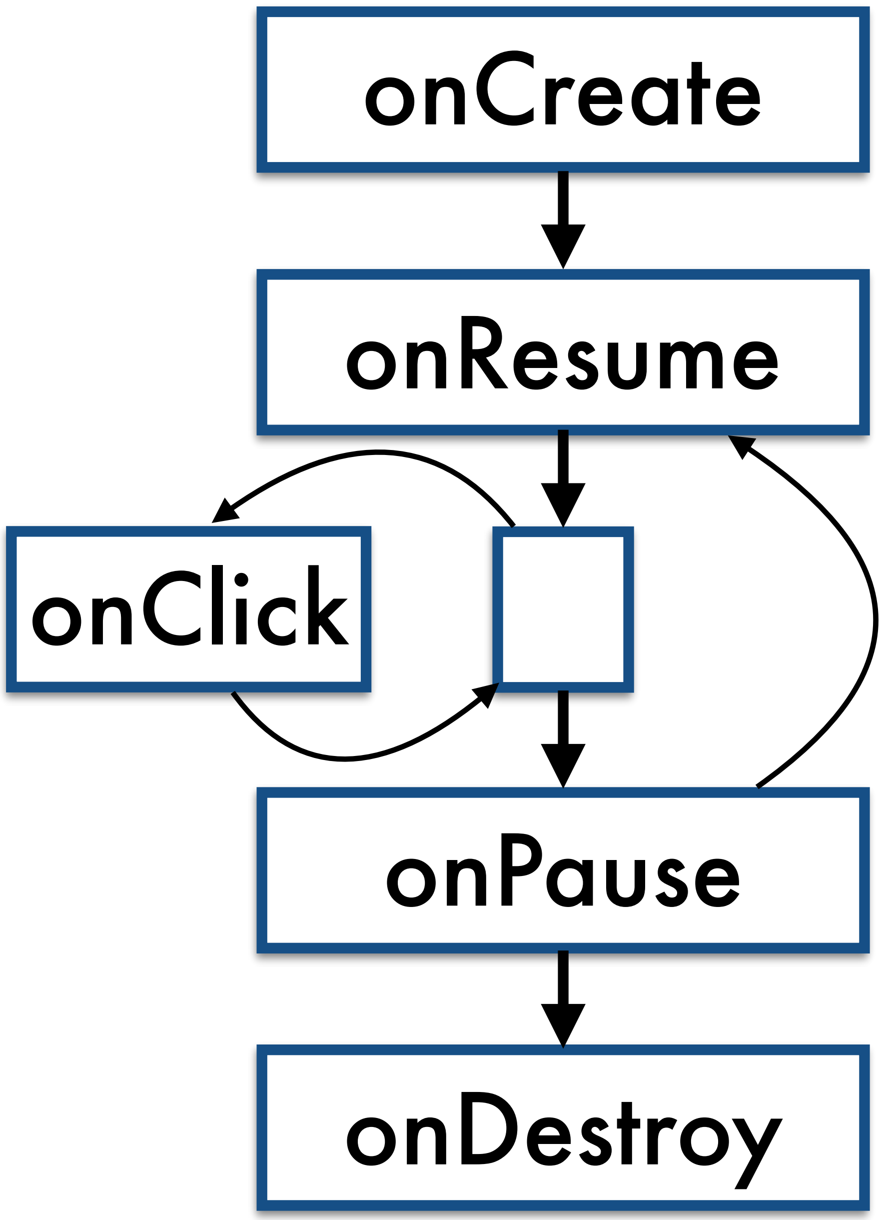


But, lifecycles of different components and other callbacks can interleave ...

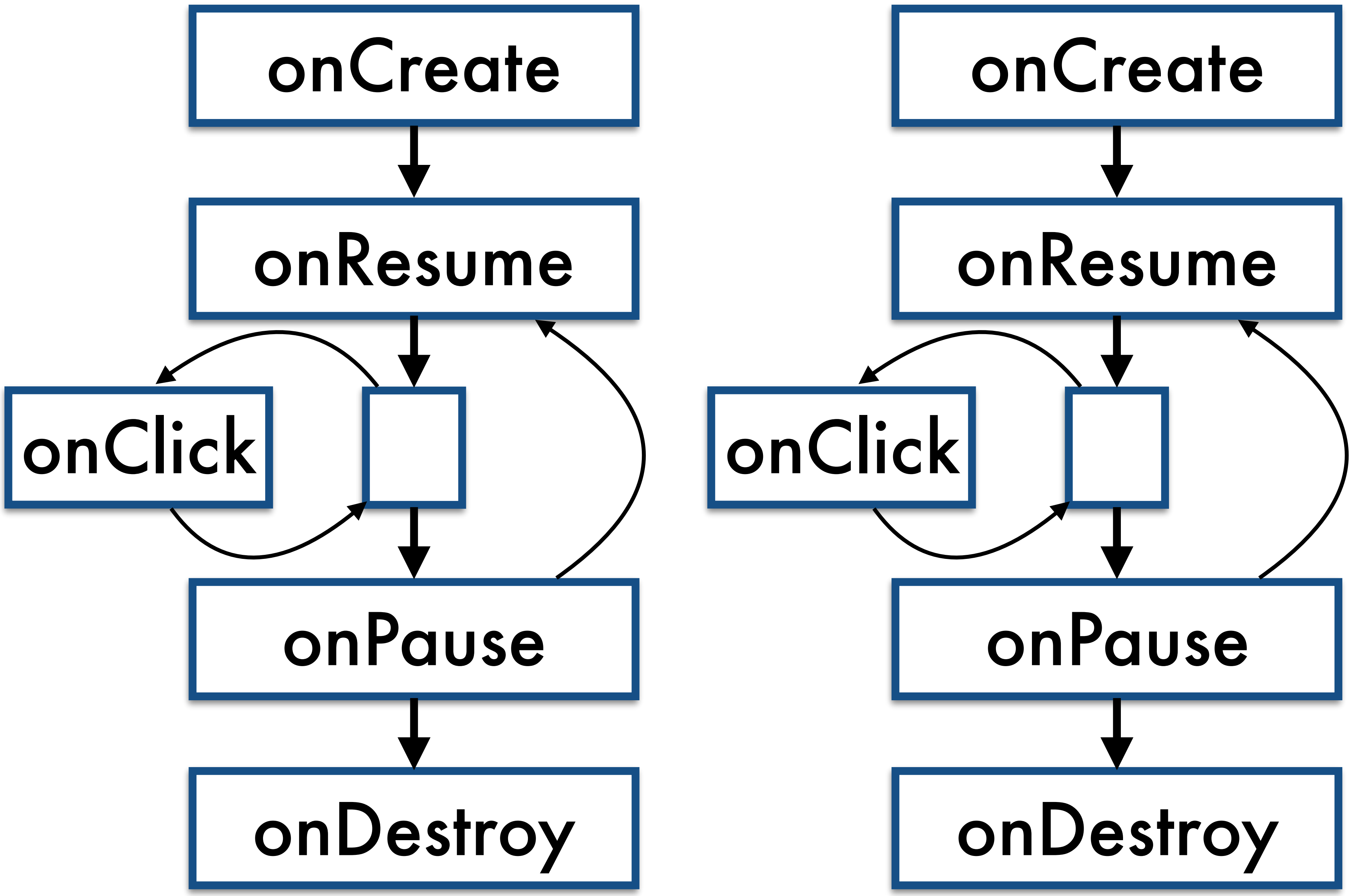
```
this.mHostDb = null;
this.mService = null;
```

Collect resources when done

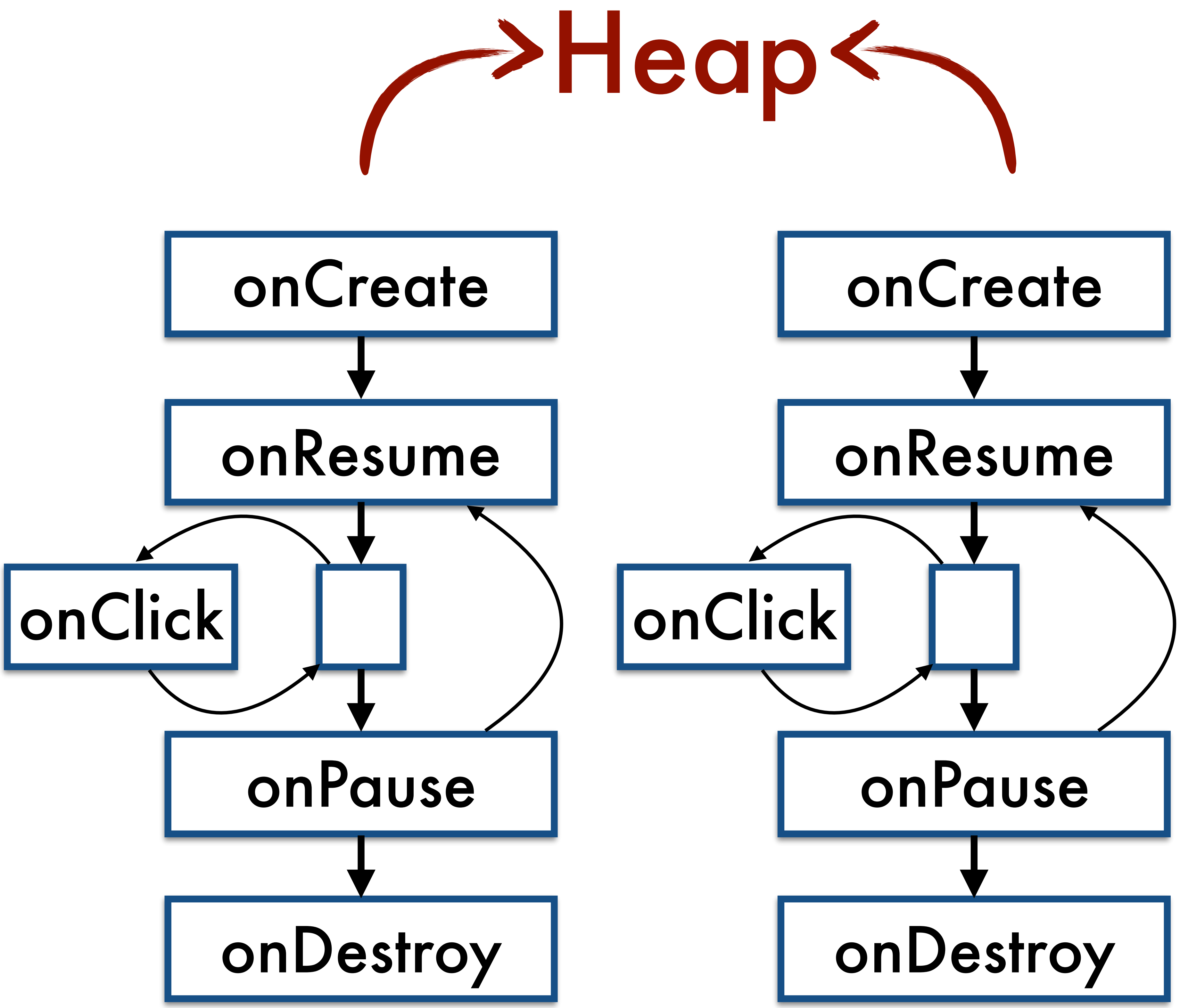
Callback-oriented programming: Interacting through a shared heap



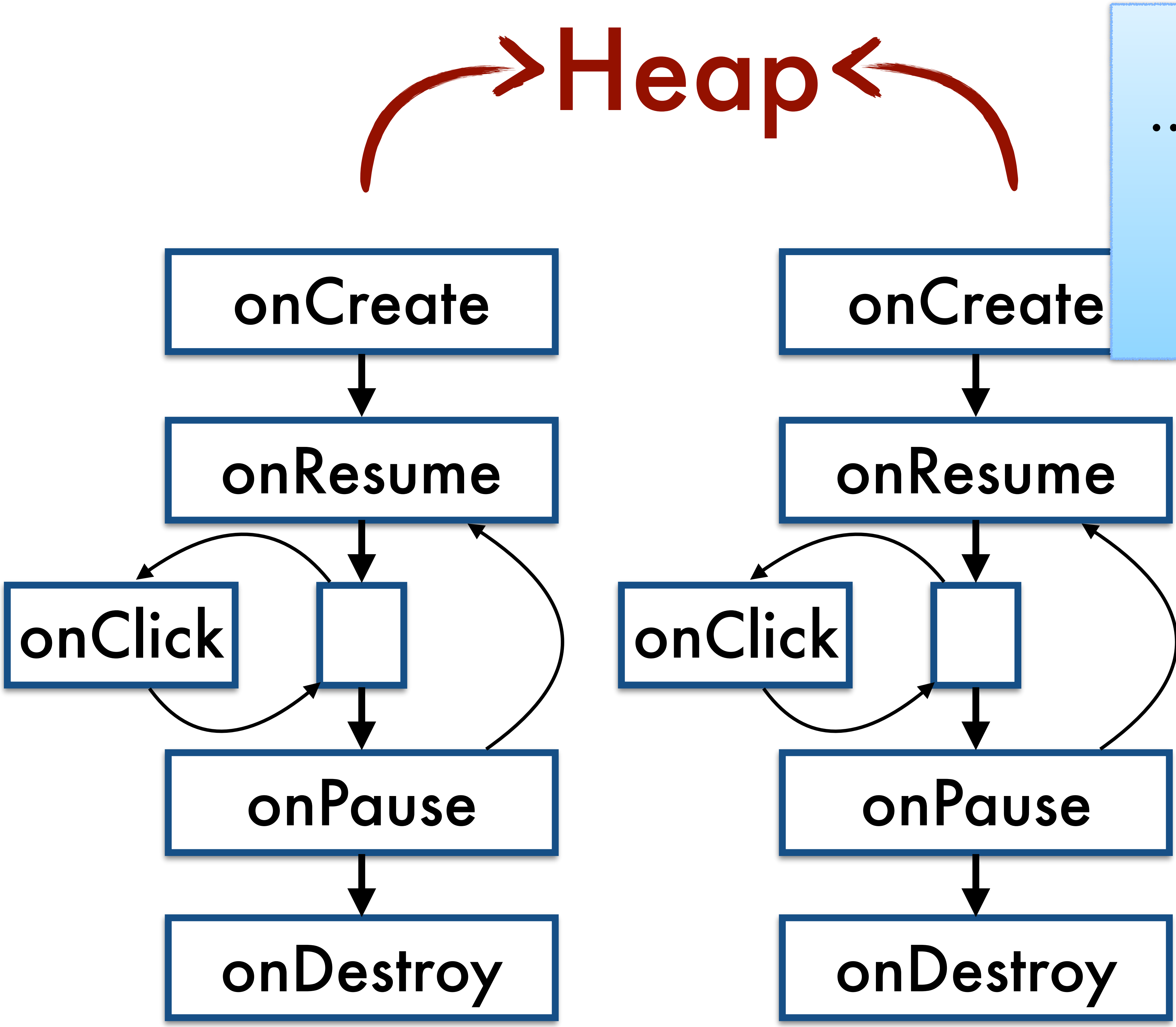
Callback-oriented programming:
Interacting through a shared heap



Callback-oriented programming: Interacting through a shared heap

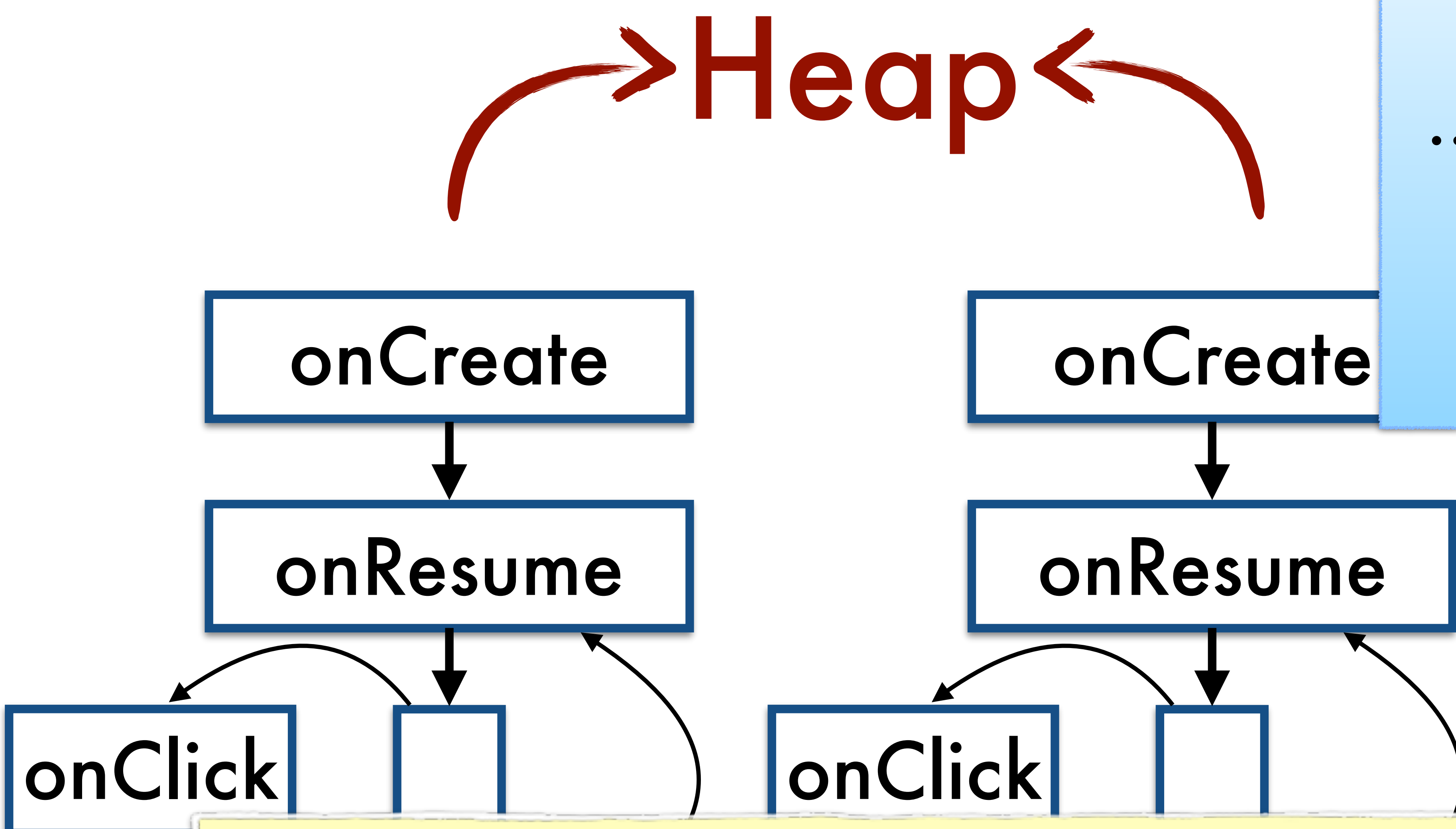


Callback-oriented programming: Interacting through a shared heap



... and operate over a shared, global heap

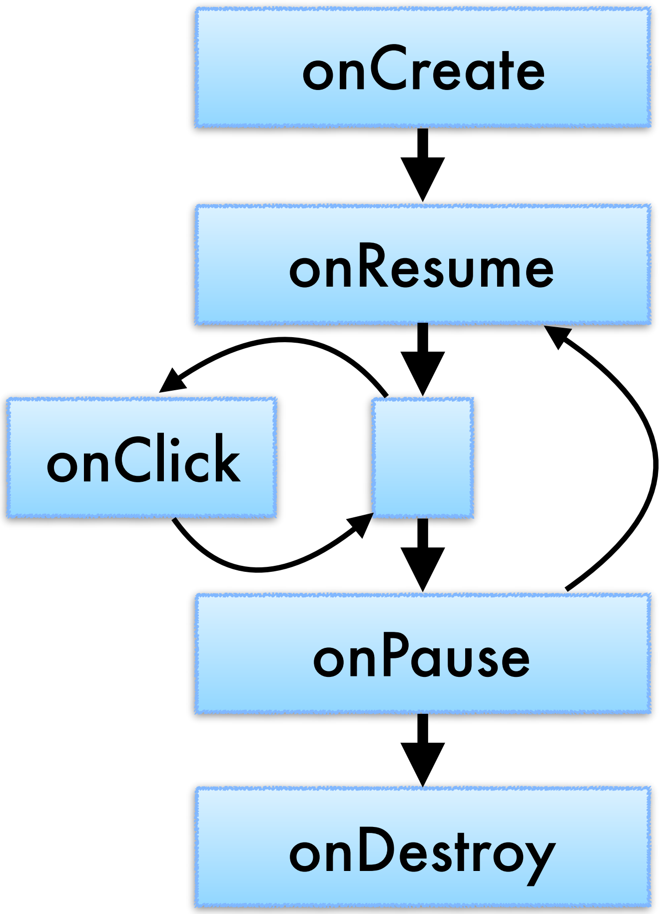
Callback-oriented programming: Interacting through a shared heap



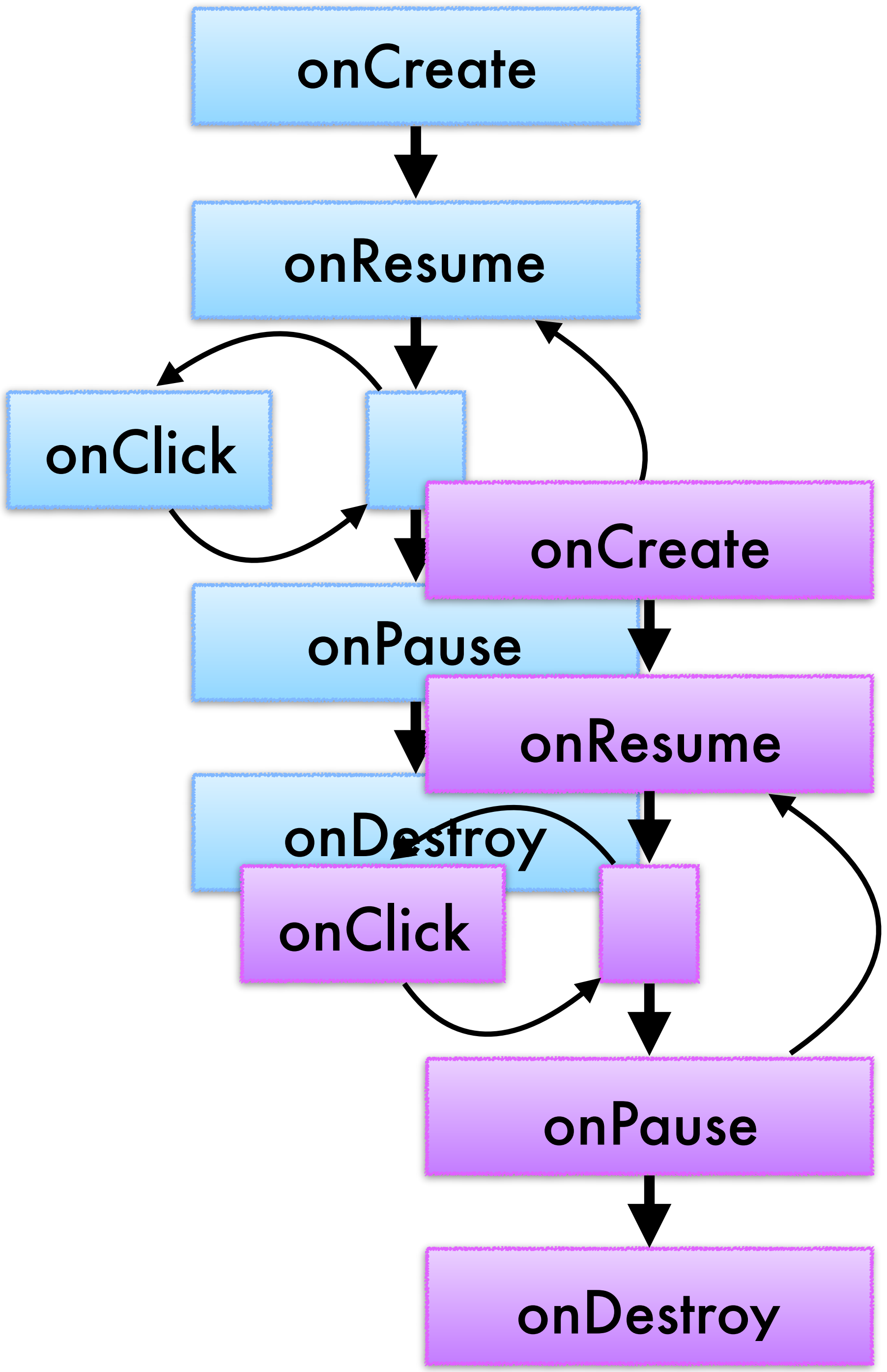
... and operate over a shared, global heap

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

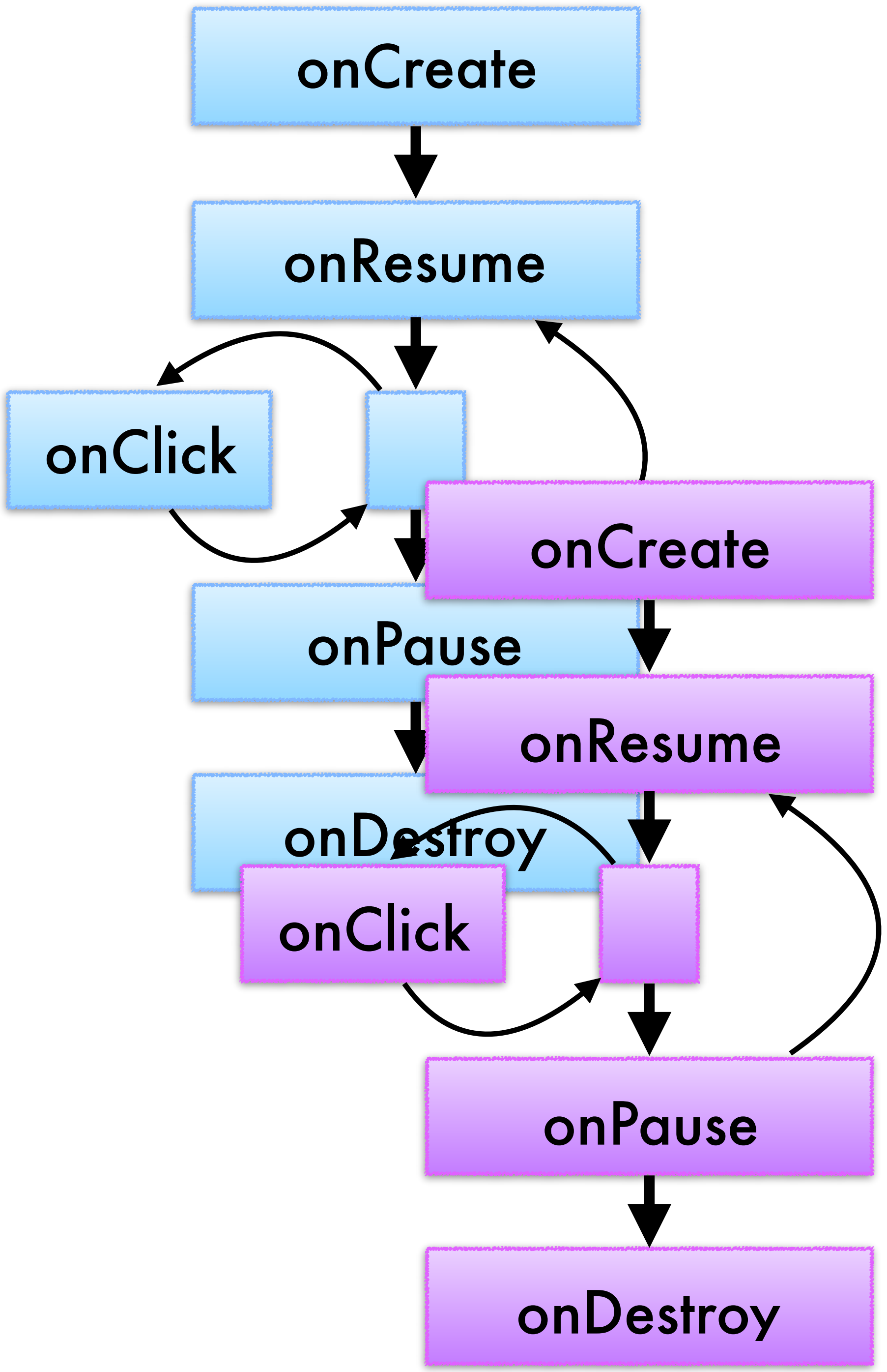
Explore callback interleavings ...

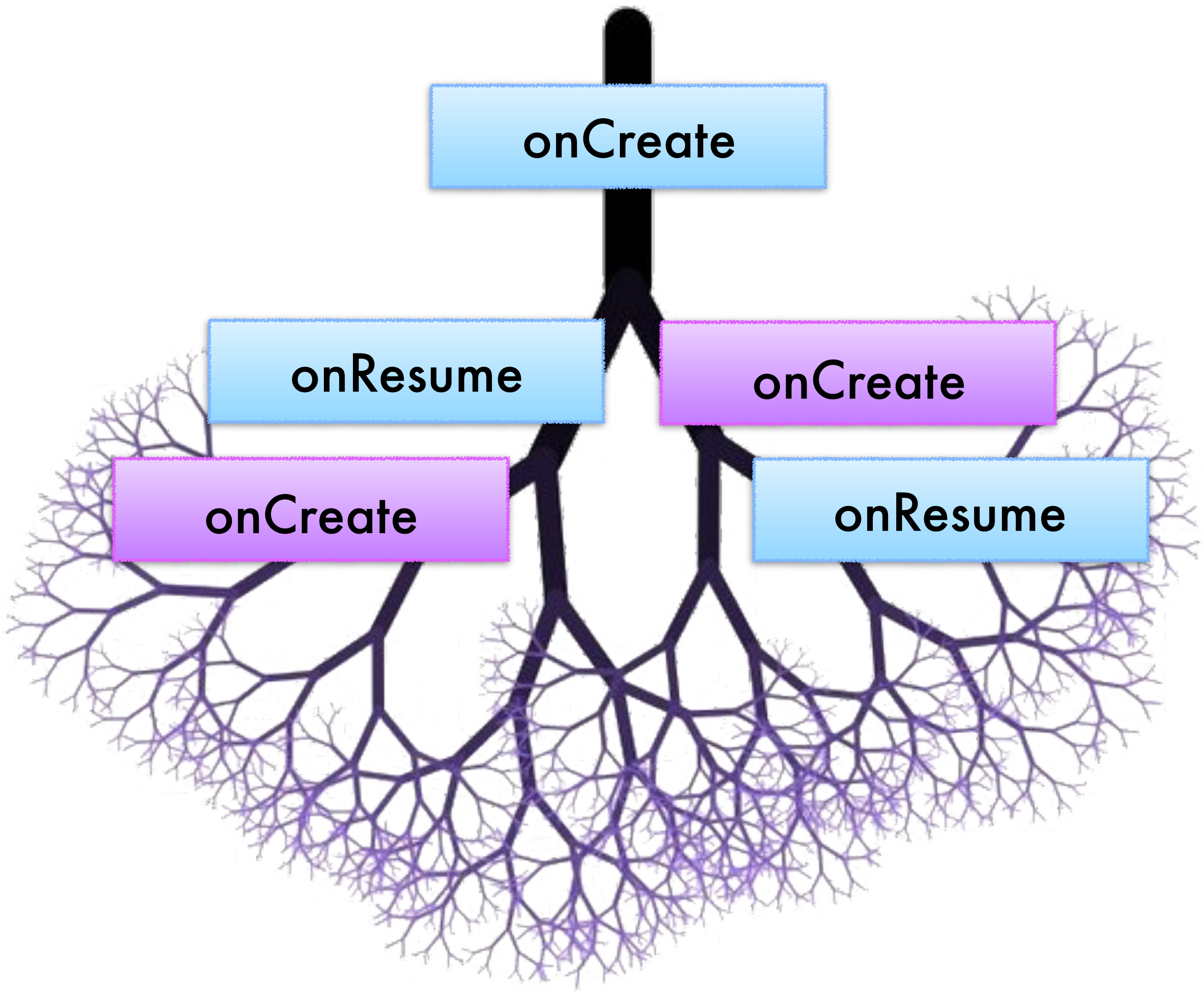


Explore callback interleavings …

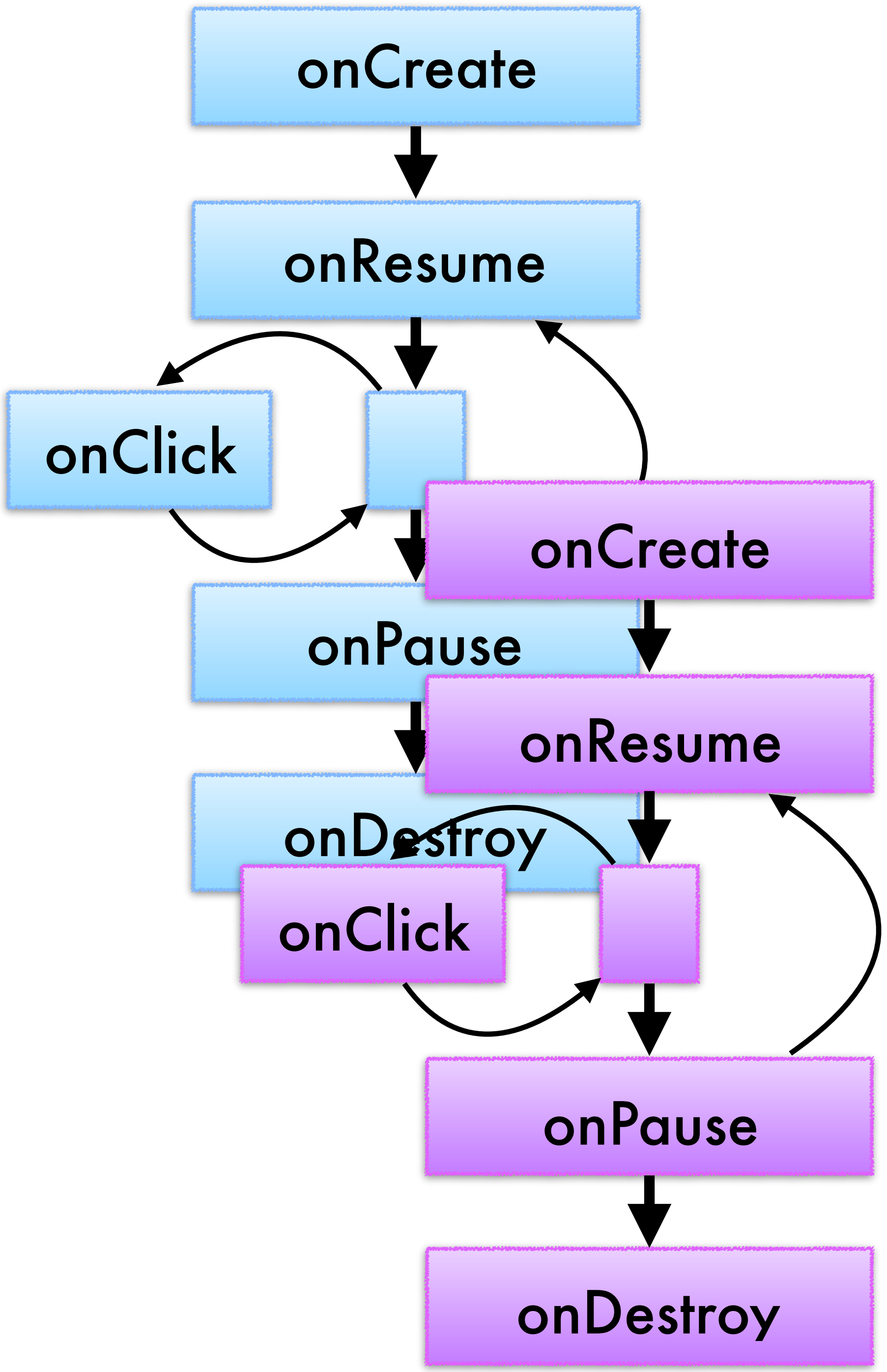


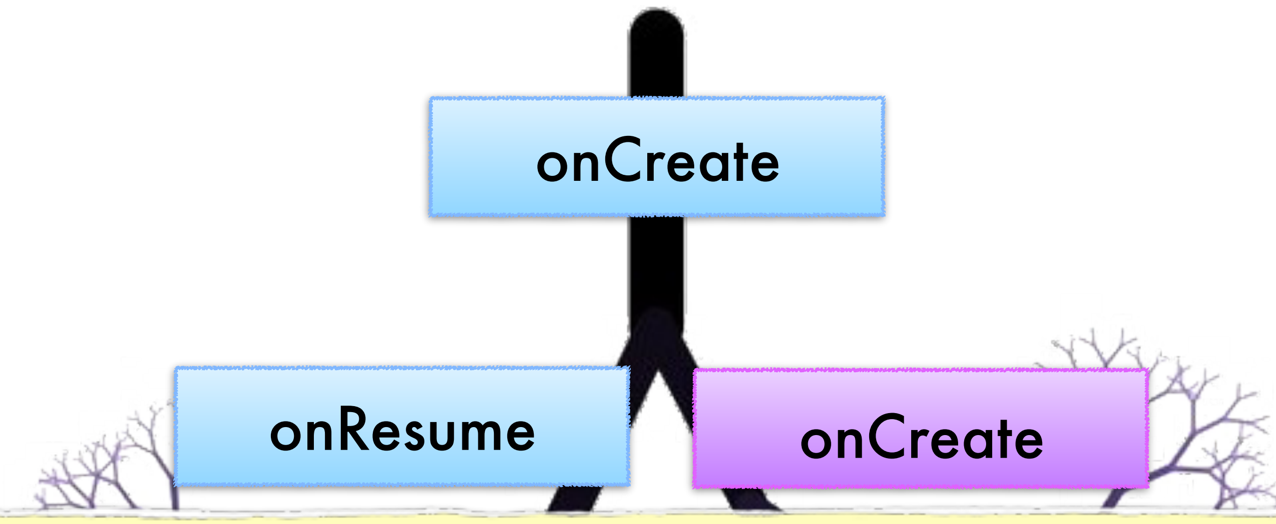
Explore callback interleavings ...





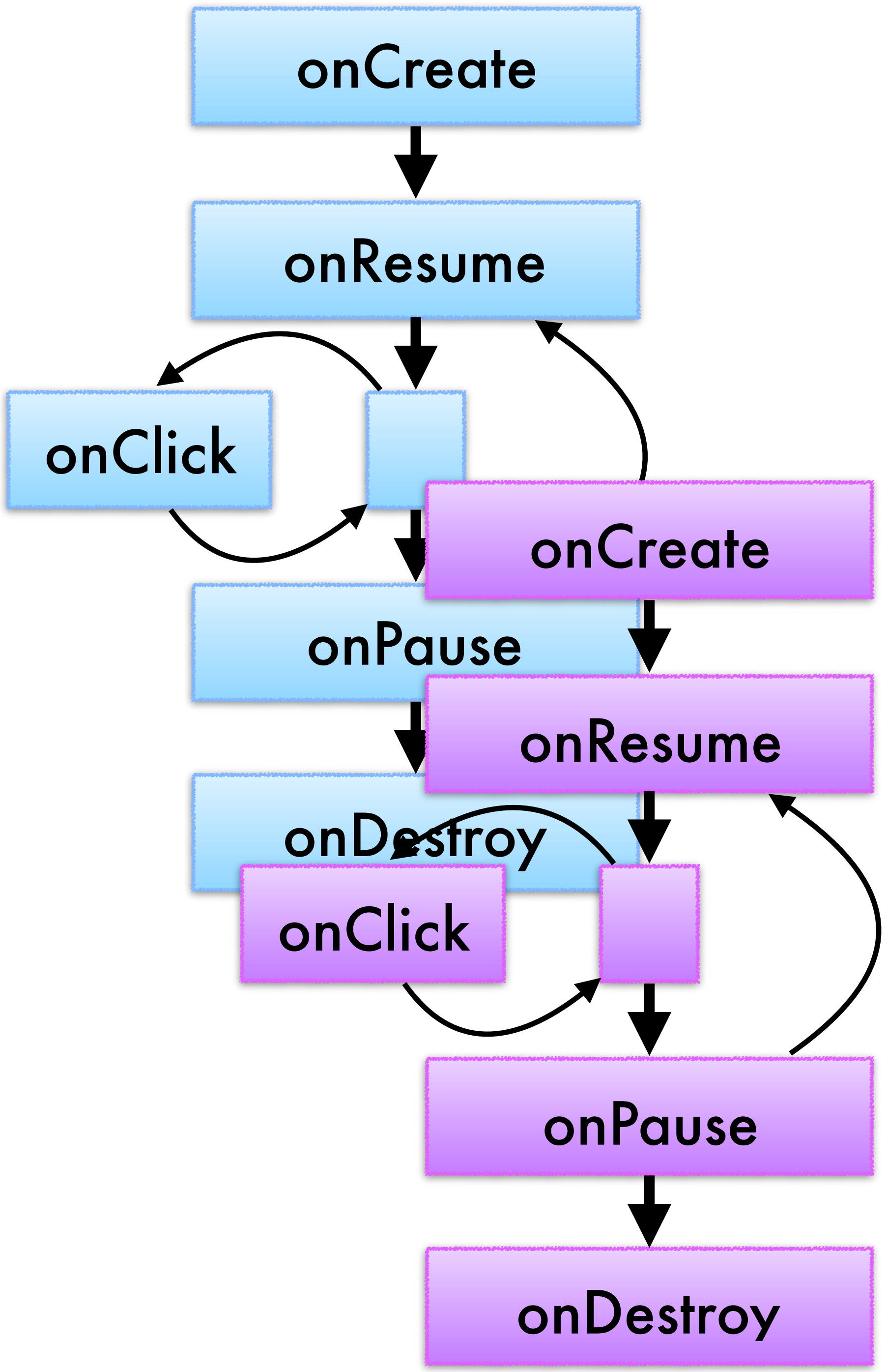
Explore callback interleavings …

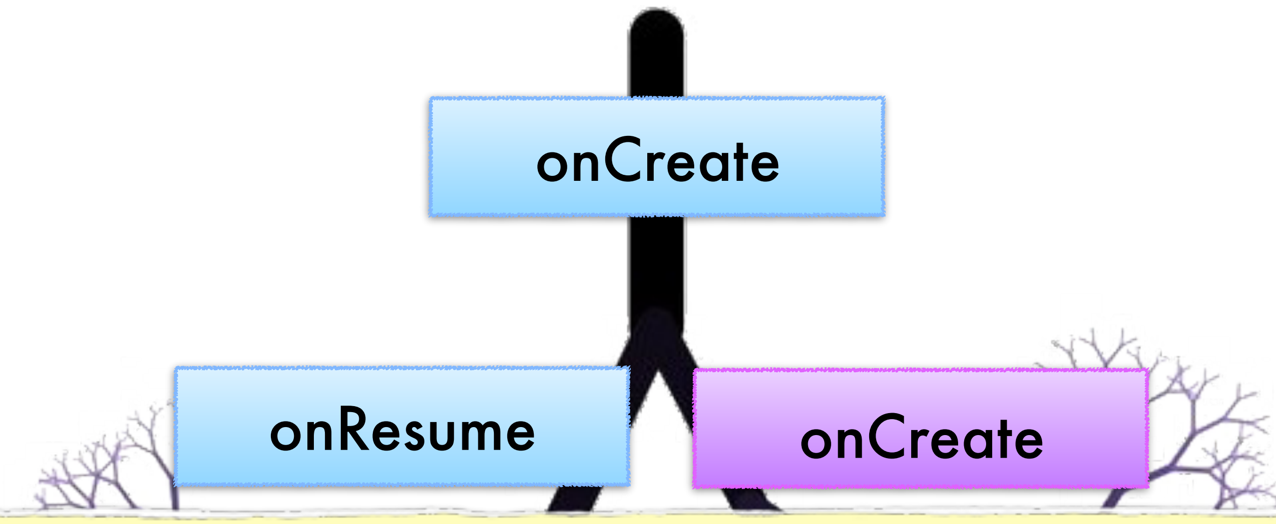




Previous analyses do not consider inter-component interleavings in a flow-sensitive way

Explore callback interleavings …

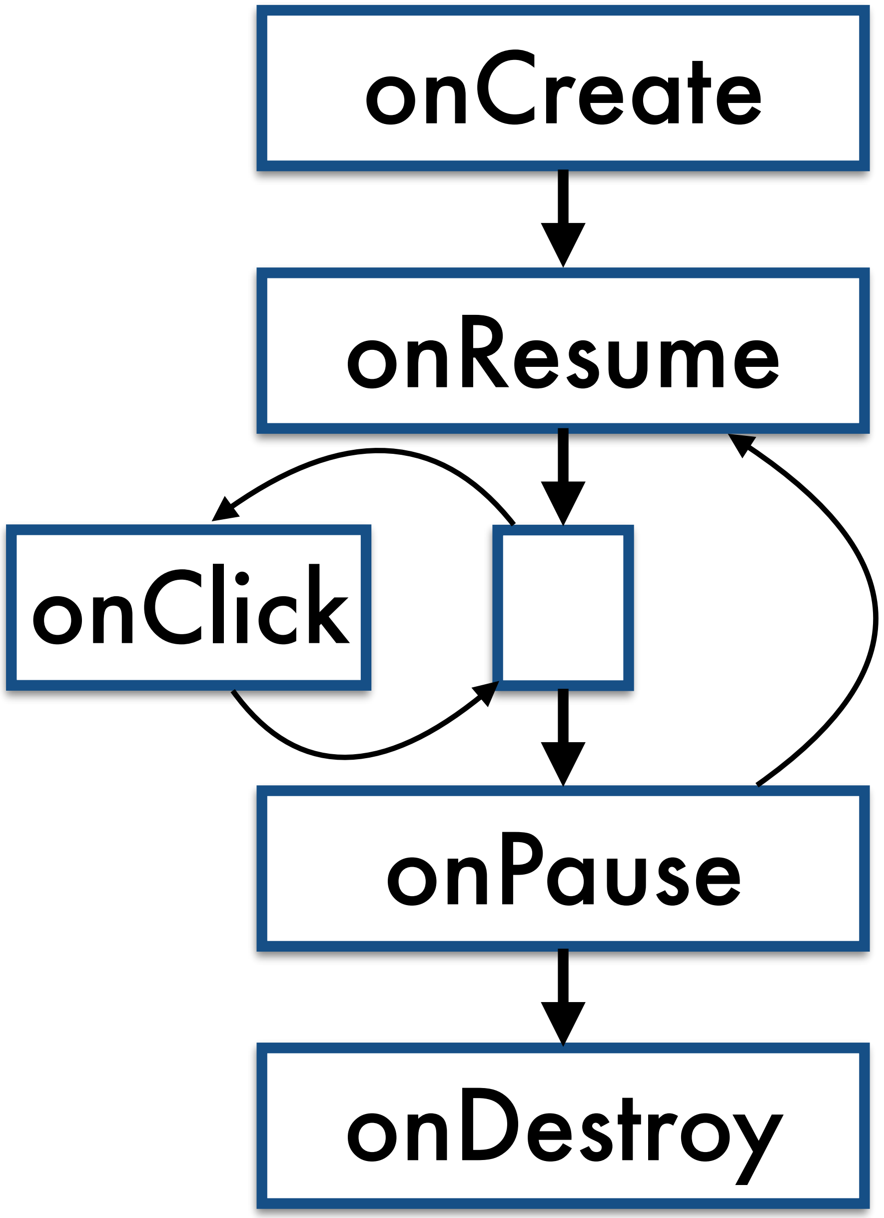




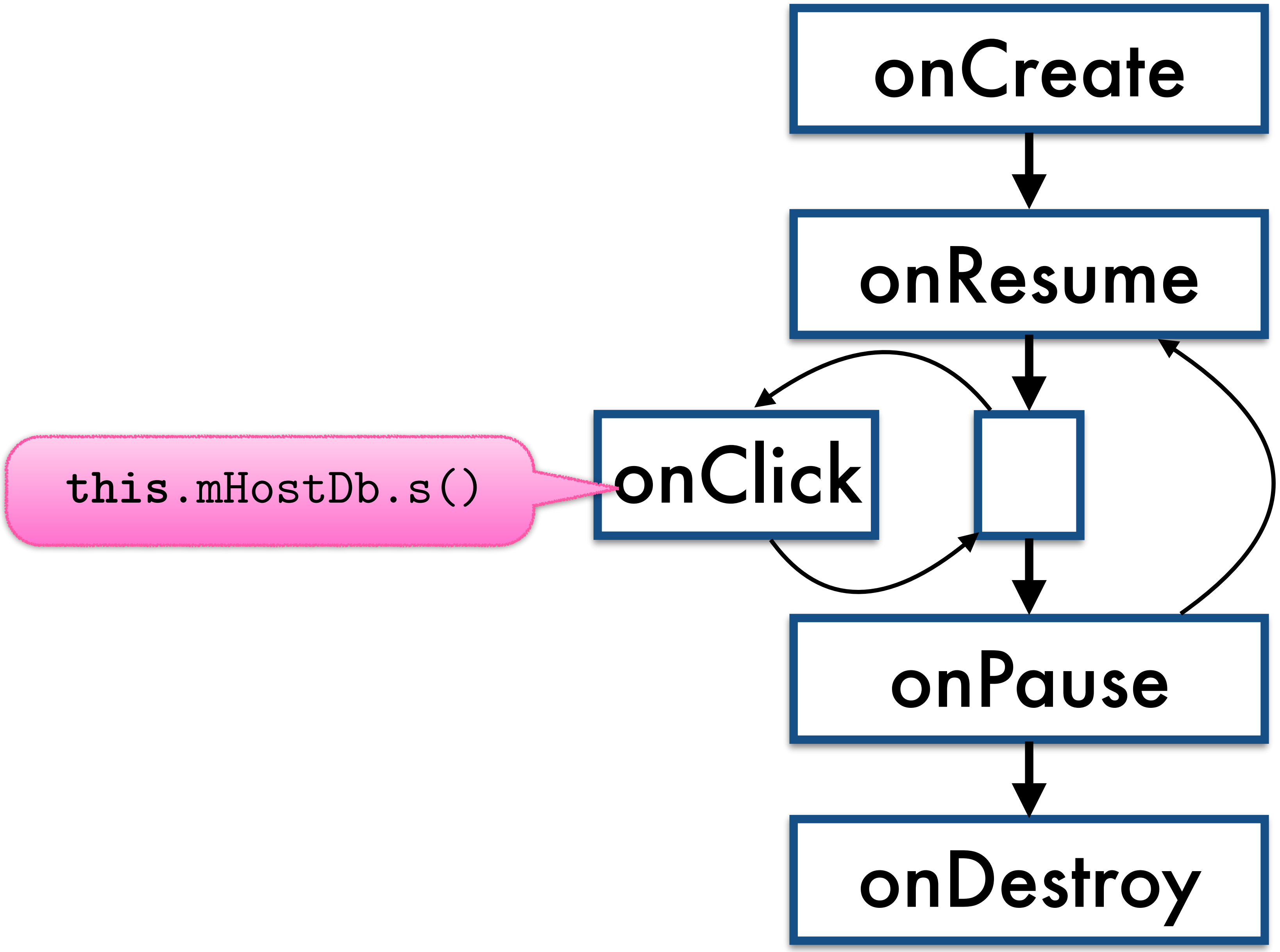
Previous analyses do not consider inter-component interleavings in a flow-sensitive way

An app with 1,320 callbacks would have created a product automaton with 10^{111} nodes (with unsoundly one instance per class)

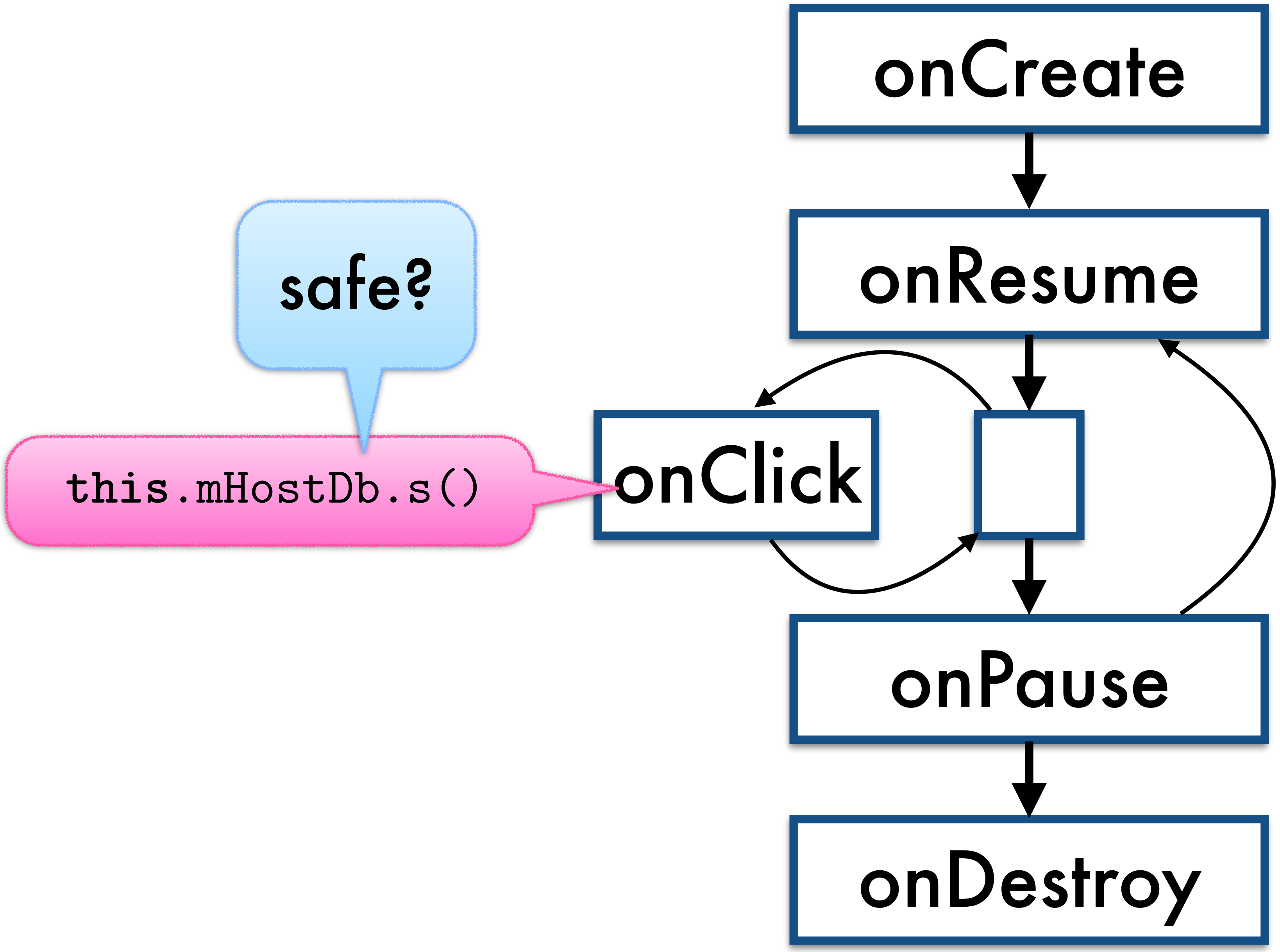
… it shouldn’t be so hard



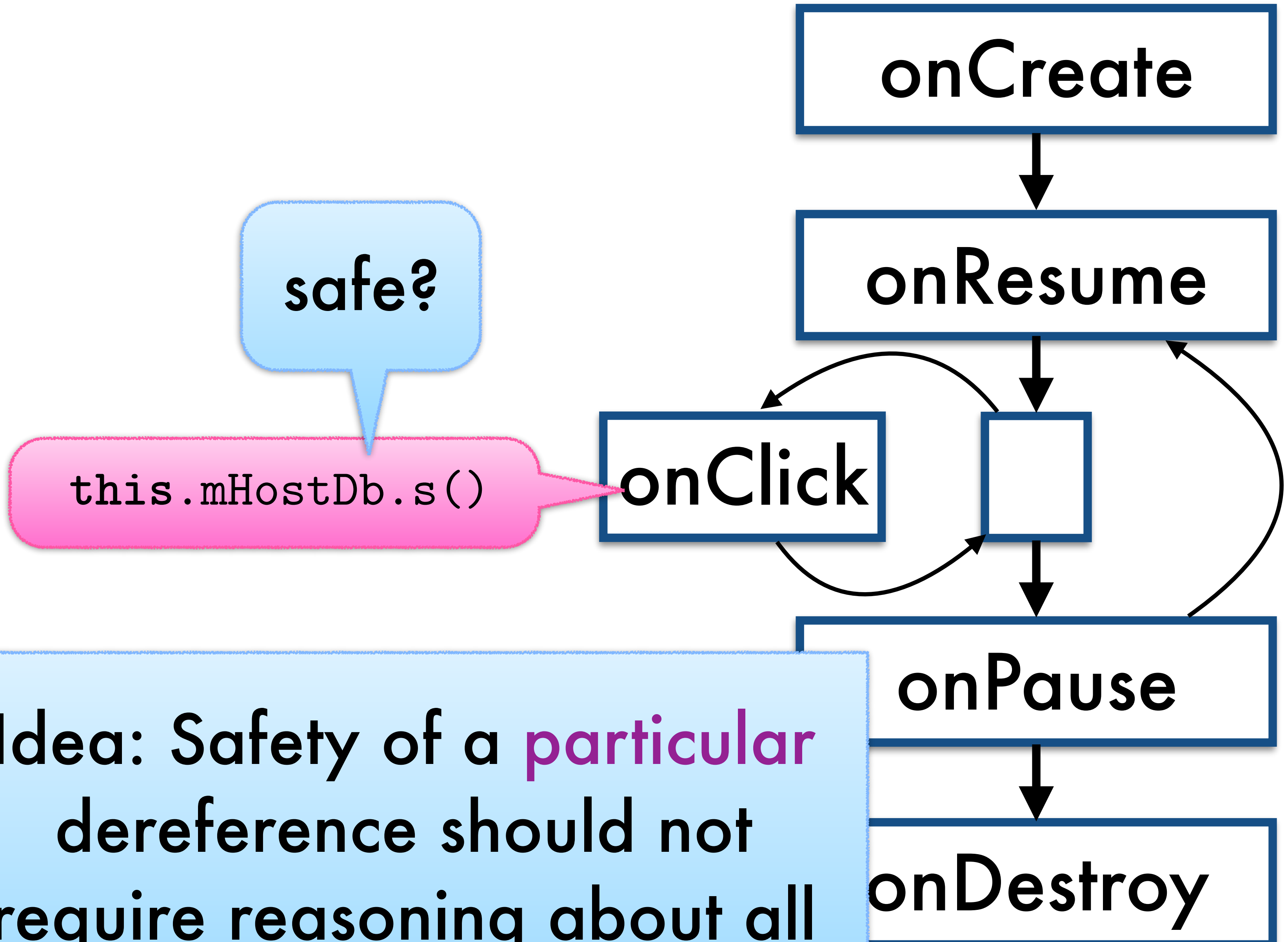
… it shouldn't be so hard



… it shouldn't be so hard

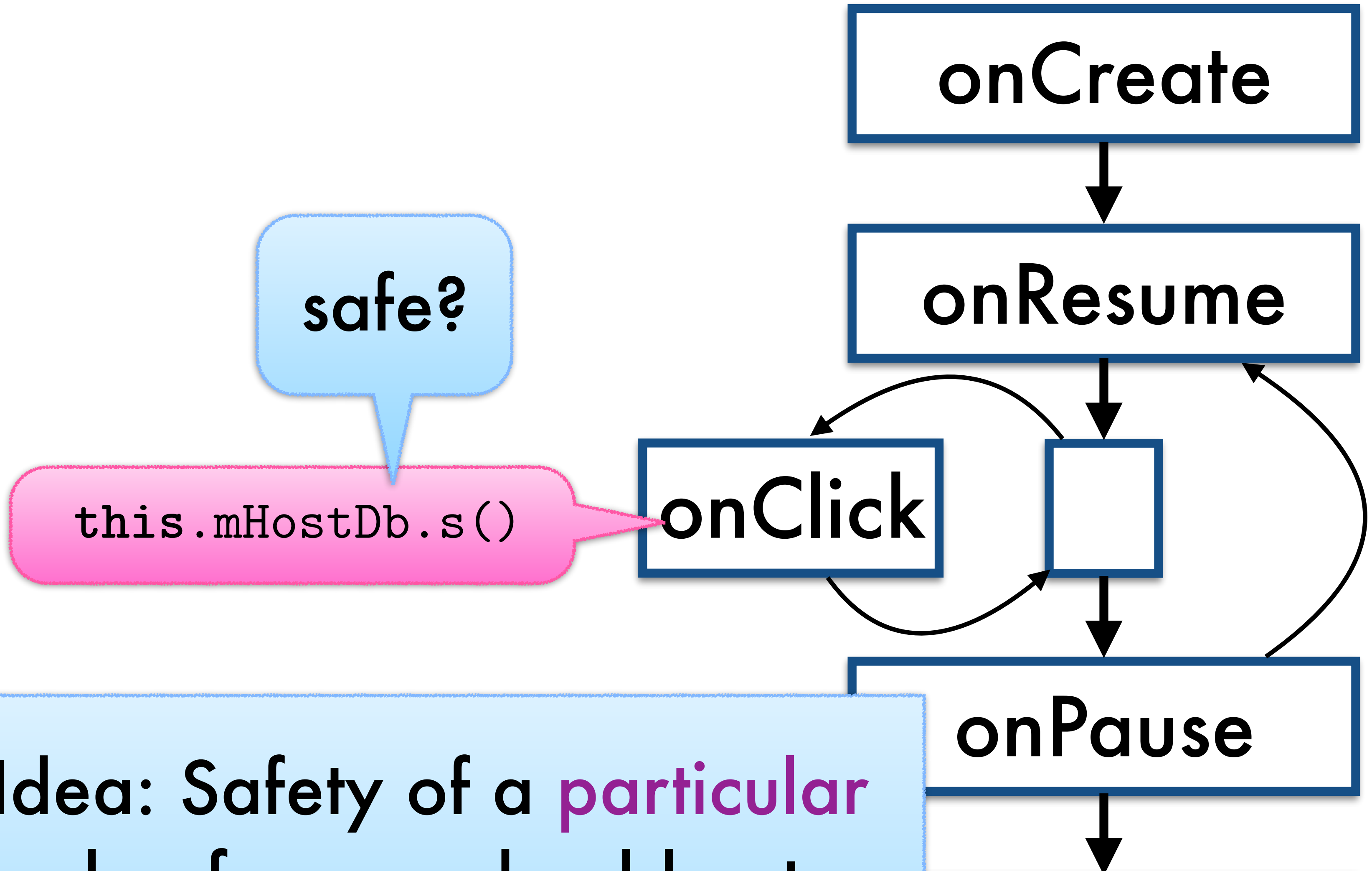


… it shouldn't be so hard



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

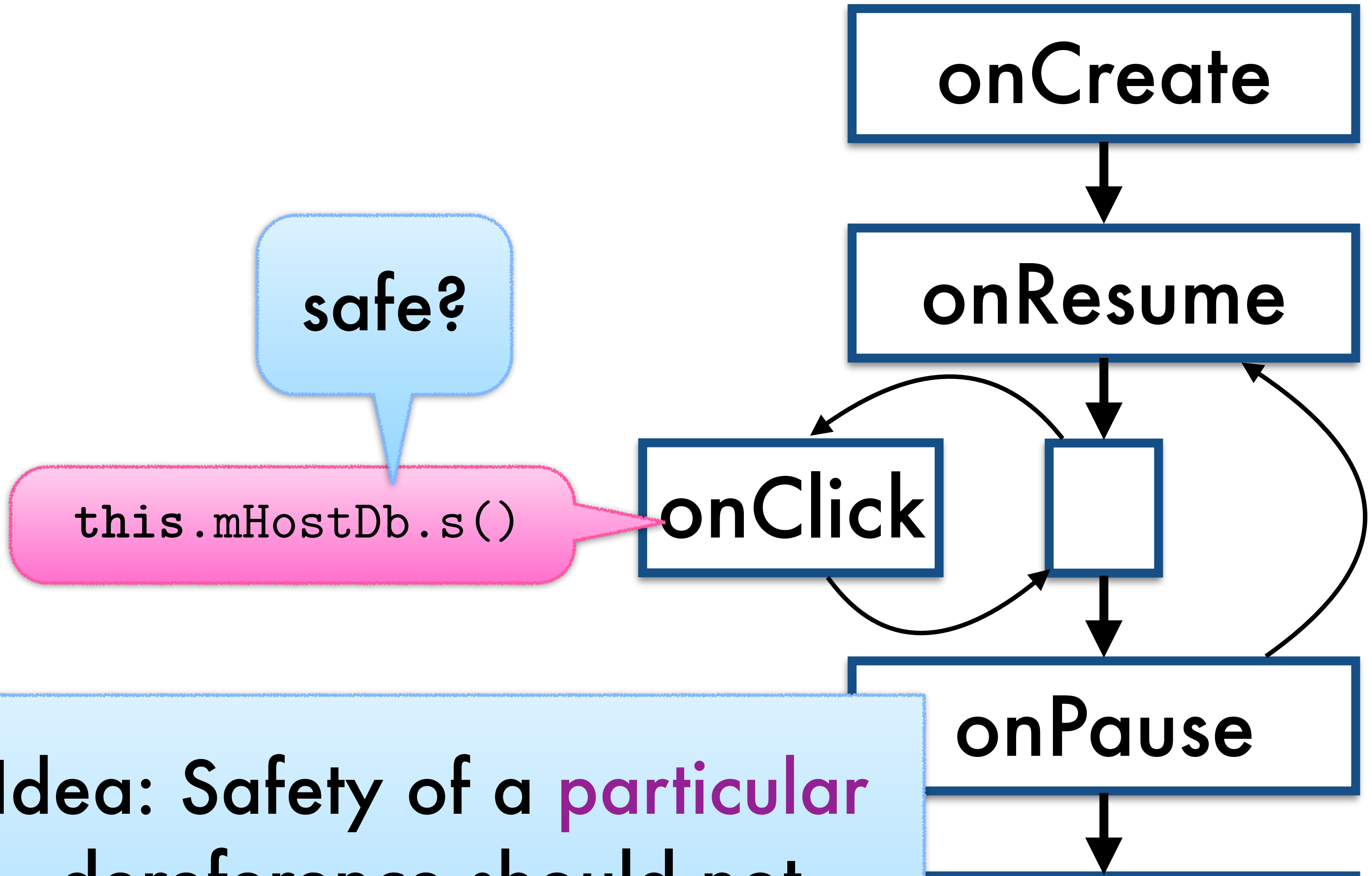
… it shouldn't be so hard



Idea: Safety of a particular dereference should not

A "smart" goal-directed analysis could consider relevant callback orderings without considering all of them

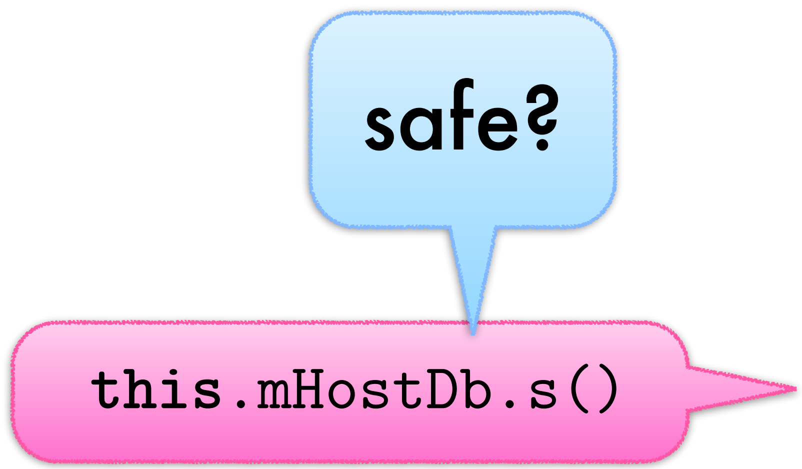
… it shouldn't be so hard



Idea: Safety of a particular dereference should not

A "smart" goal-directed analysis could consider relevant callback orderings without considering all of them

Goal-directed program analysis



Goal-directed program analysis

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

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()`

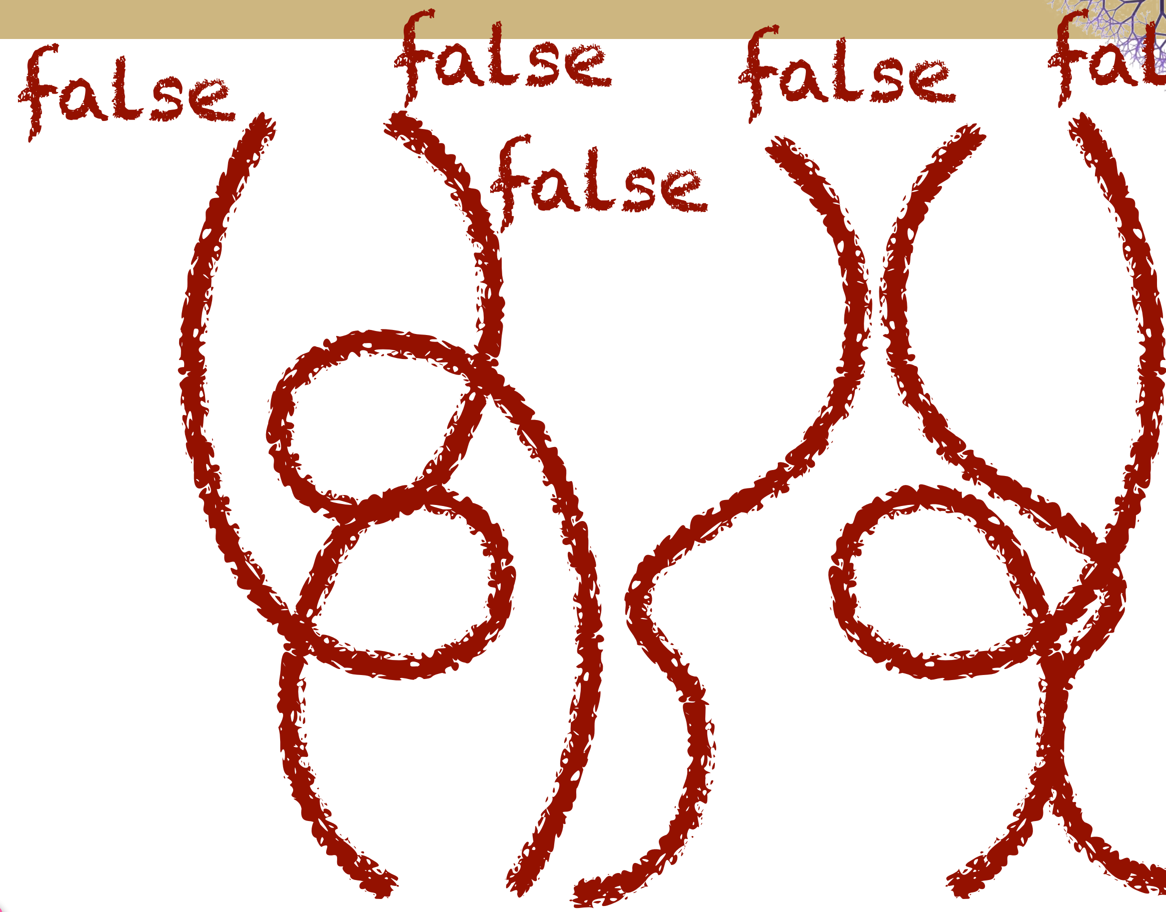
`mHostDb == null`

Goal-directed program analysis

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

safe?

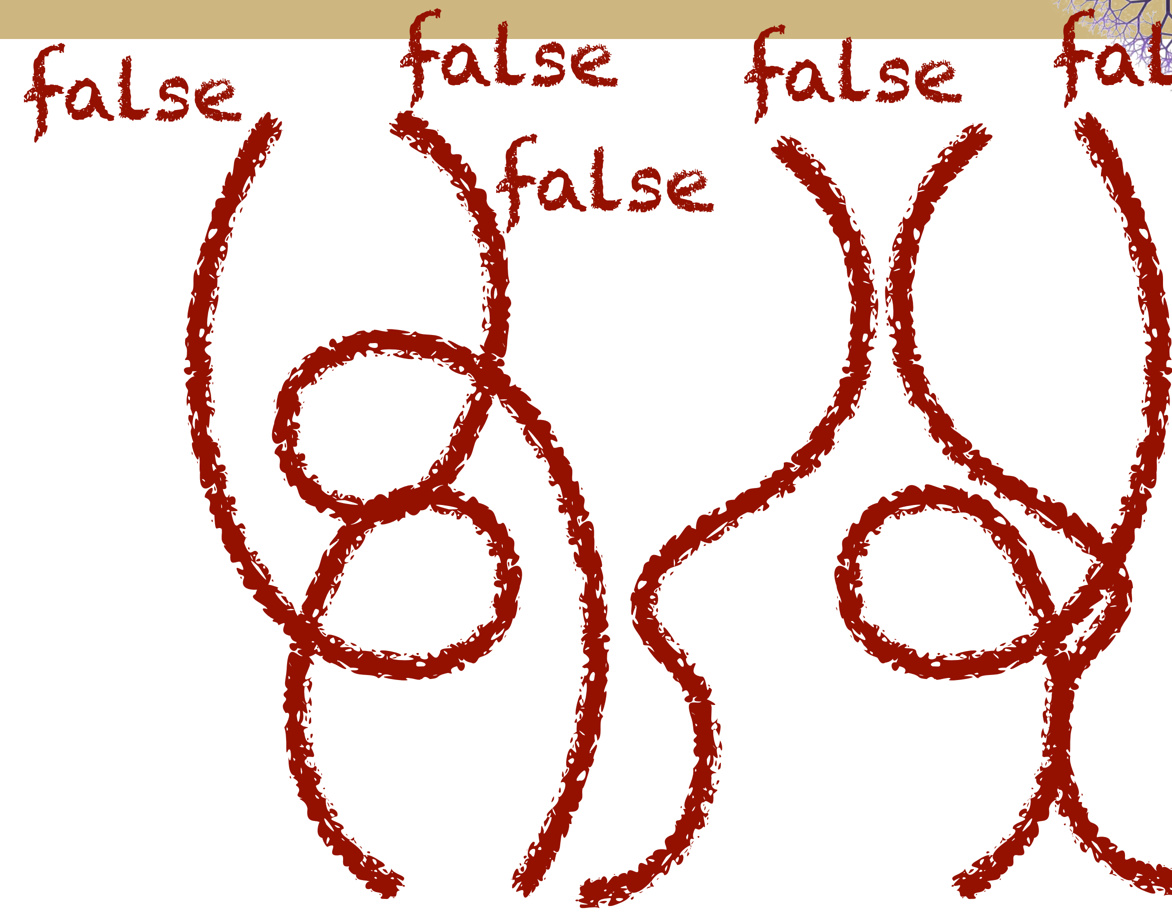
`this.mHostDb.s()`



`mHostDb == null`

Goal-directed program analysis

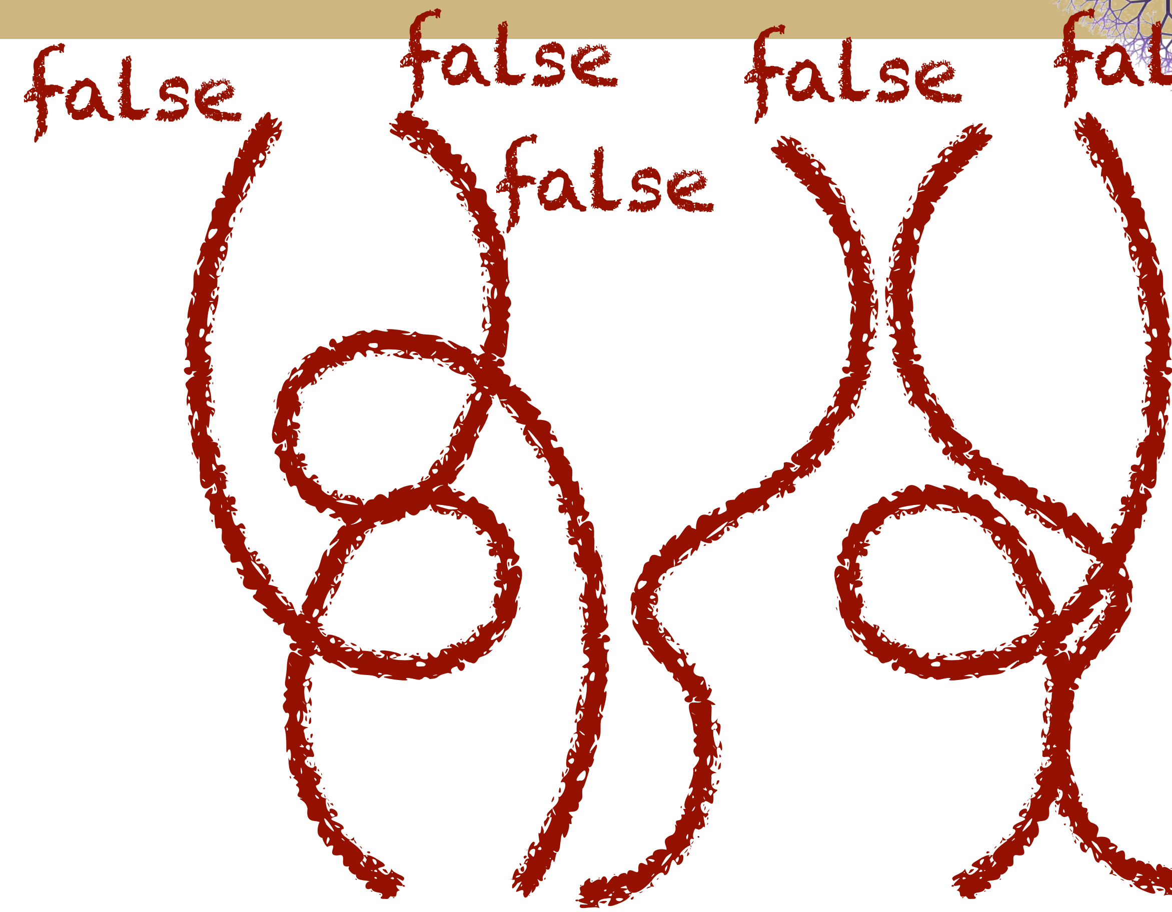
Given a program configuration **goal**, derive a **contradiction** w.r.t. its reachability



$$(\mathbf{this} \mapsto \widehat{t} * \widehat{t} \cdot \texttt{mHostDb} \mapsto \widehat{a} * \mathsf{true}) \wedge \widehat{a} = \mathsf{null}$$

Goal-directed program analysis

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

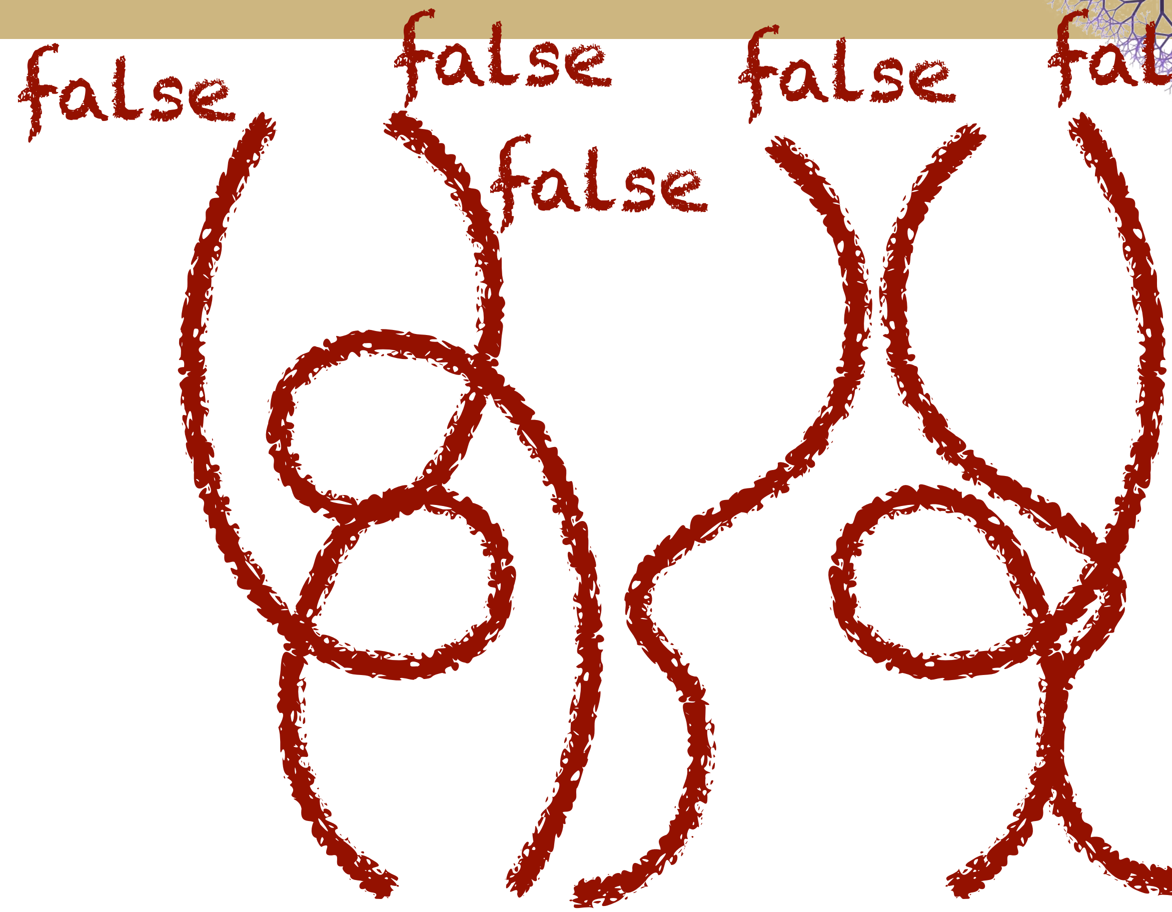


$$(\mathbf{this} \mapsto \widehat{t} * \widehat{t}\cdot\mathtt{mHostDb} \mapsto \widehat{a} * \mathsf{true}) \wedge \widehat{a} = \mathsf{null}$$

Thresher: A backwards abstract interpretation with separation logic constraints to refute error conditions [PLDI'13]

Goal-directed program analysis

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

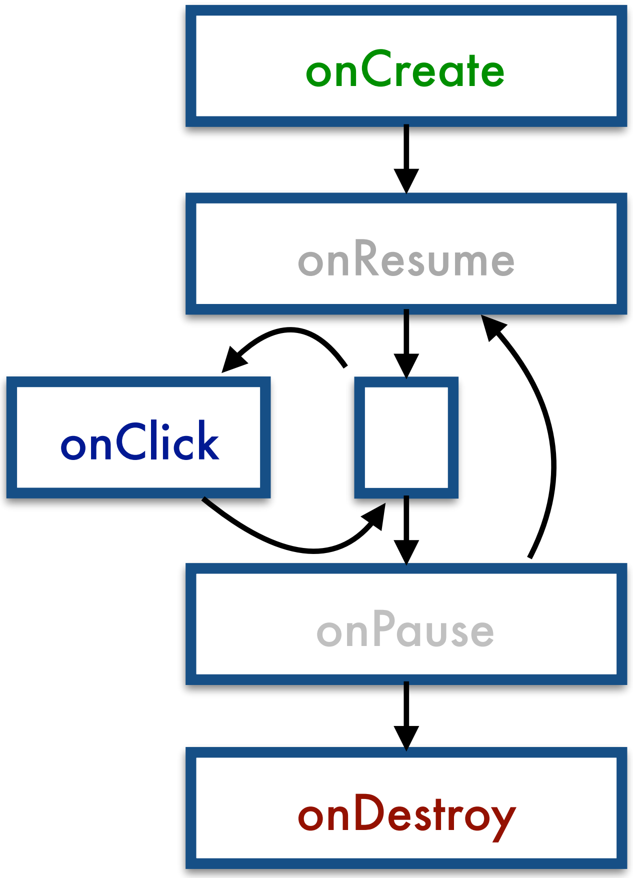


$$(\mathbf{this} \mapsto \widehat{t} * \widehat{t} \cdot \texttt{mHostDb} \mapsto \widehat{a} * \mathsf{true}) \wedge \widehat{a} = \mathsf{null}$$

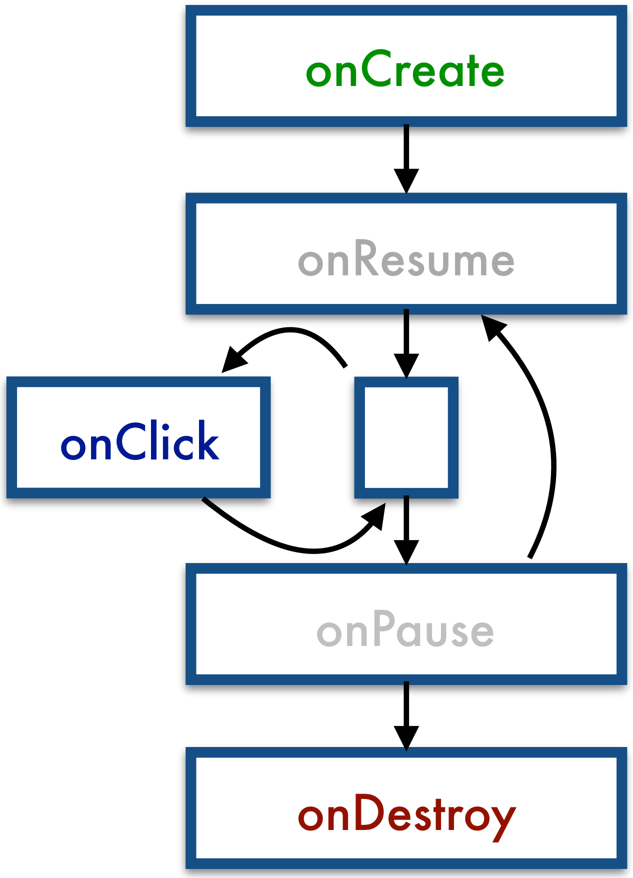
over-approximate

Thresher: A backwards abstract interpretation with separation logic constraints to refute error conditions [PLDI'13]

Being smart …
Two dereferences: one safe and one buggy

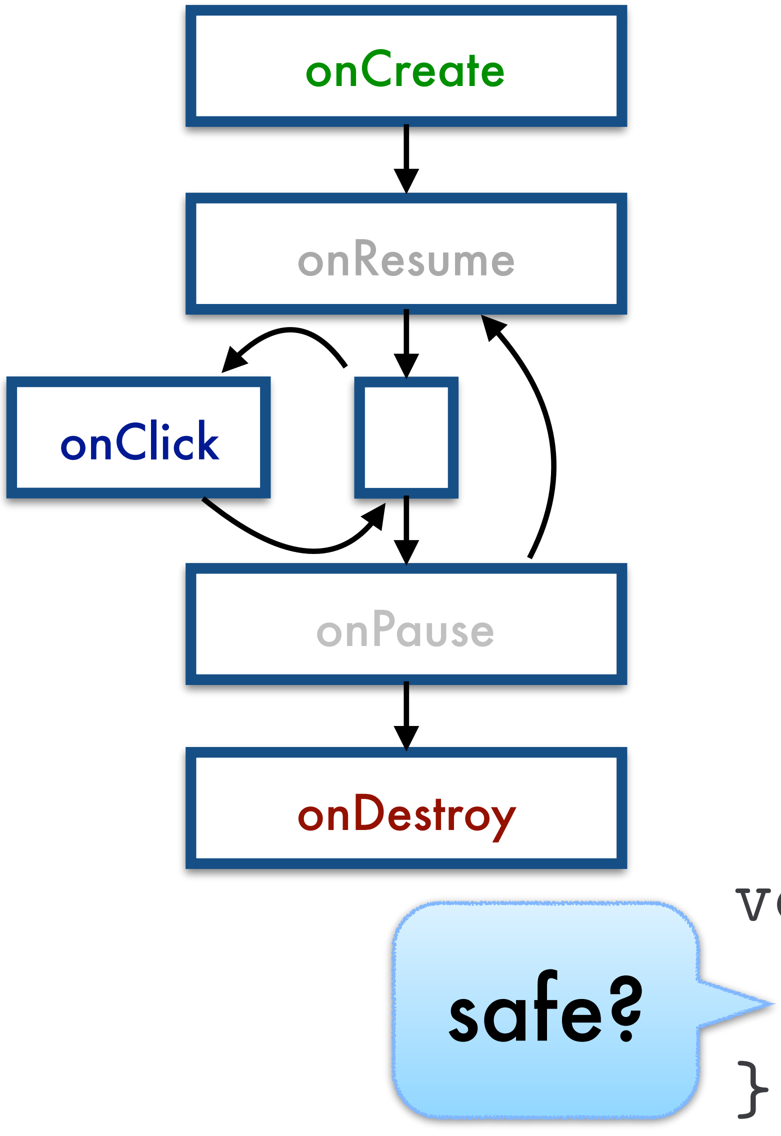


Being smart …
Two dereferences: one safe and one buggy



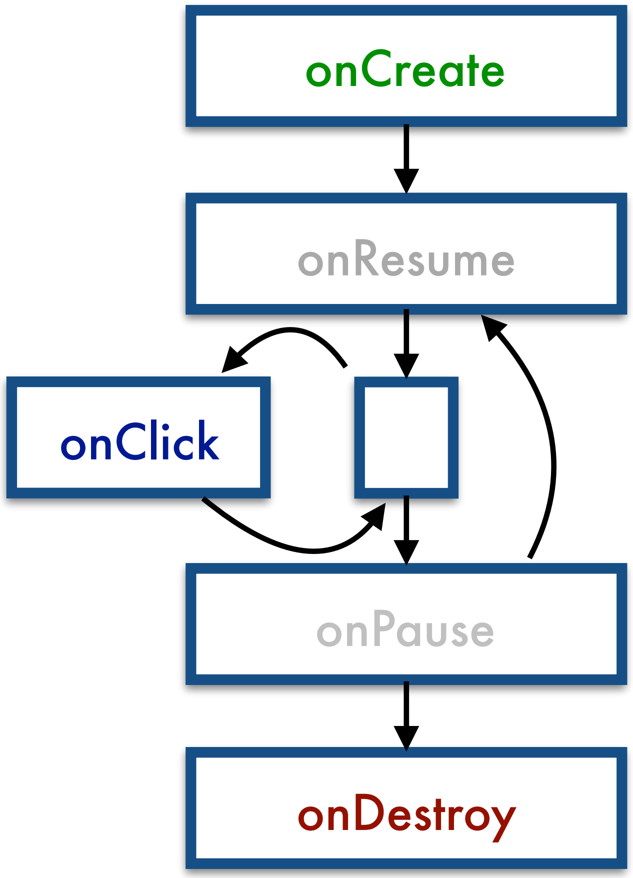

```
void onClick(…) {
  this.mHostDb.s(this.mService.g());
}
```

Being smart …
Two dereferences: one safe and one buggy




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

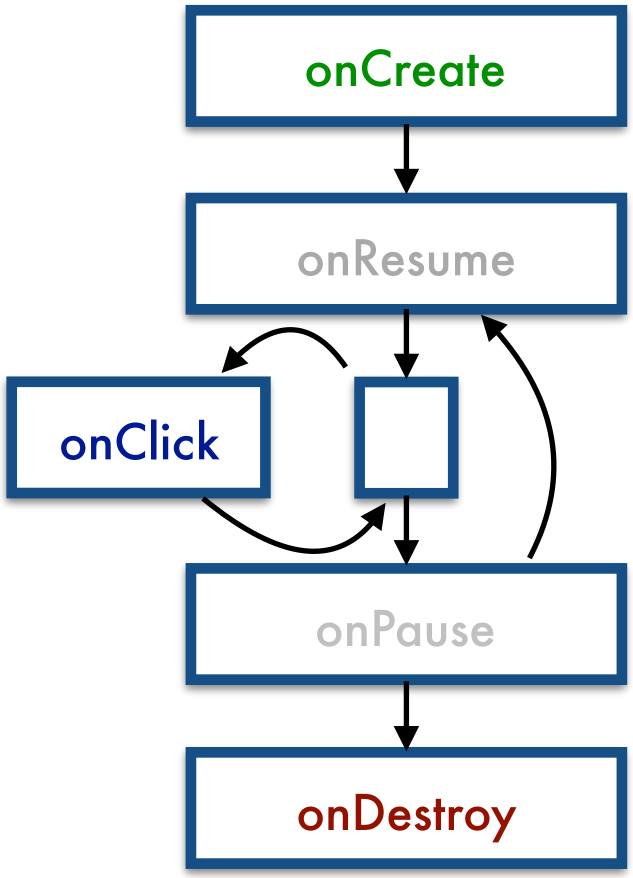
Being smart …
Two dereferences: one safe and one buggy



safe?

```
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

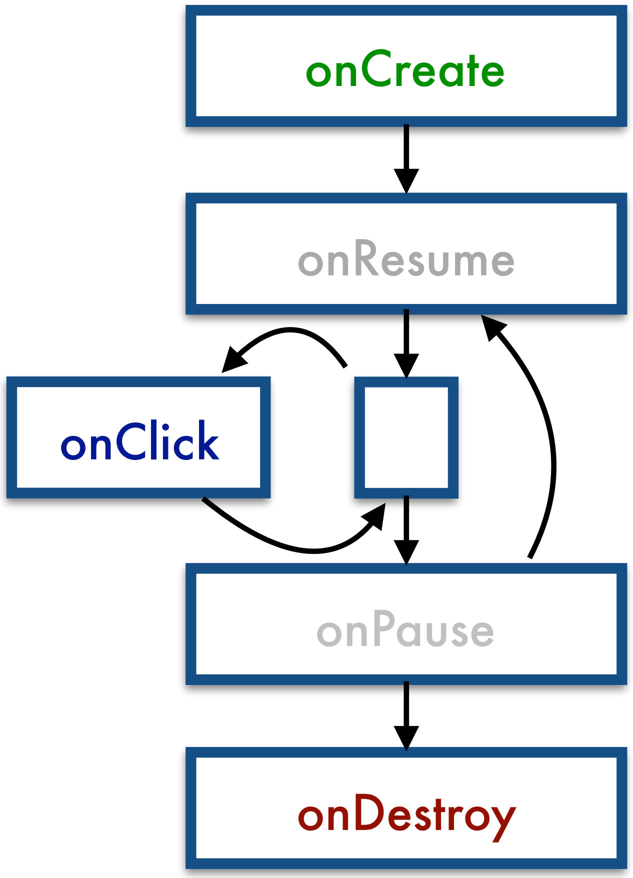


safe?

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

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

1 2

Being smart …
Two dereferences: one safe and one buggy



safe?

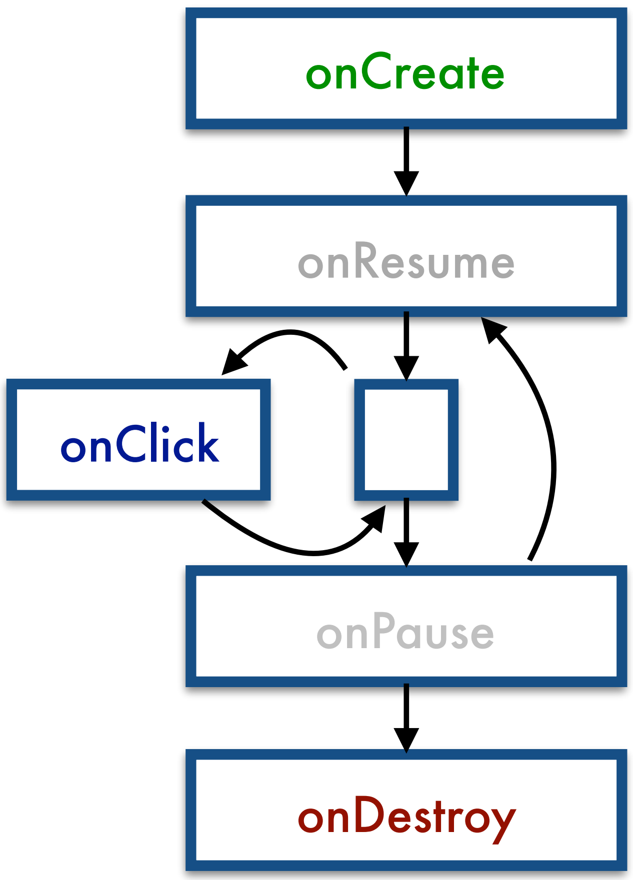
```
void onClick(…) {
  this.mHostDb.s(this.mService.g());
}
```

1 2

lifecycle constraints relevant

```
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();
}
```

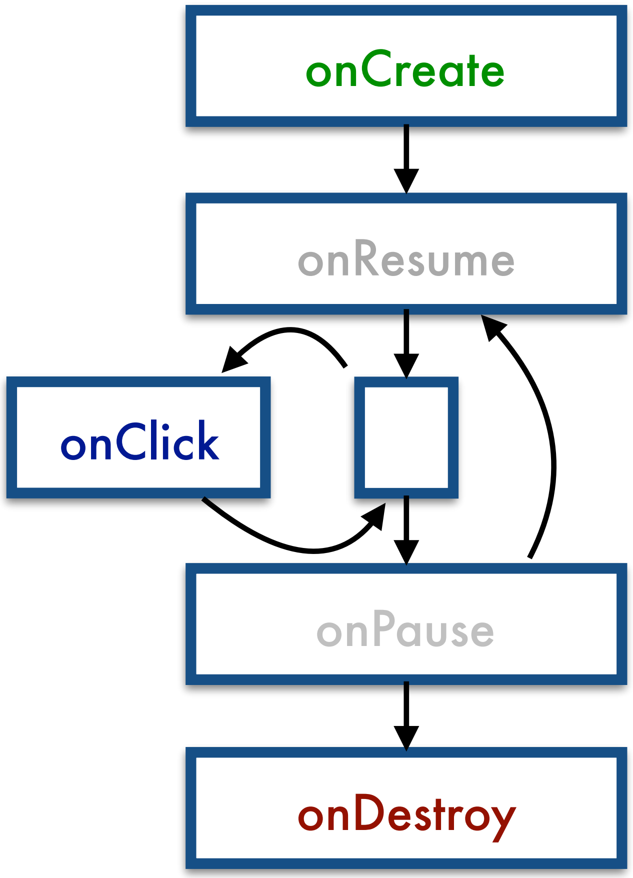
safe?

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

lifecycle constraints relevant

```
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();
}
```

safe?

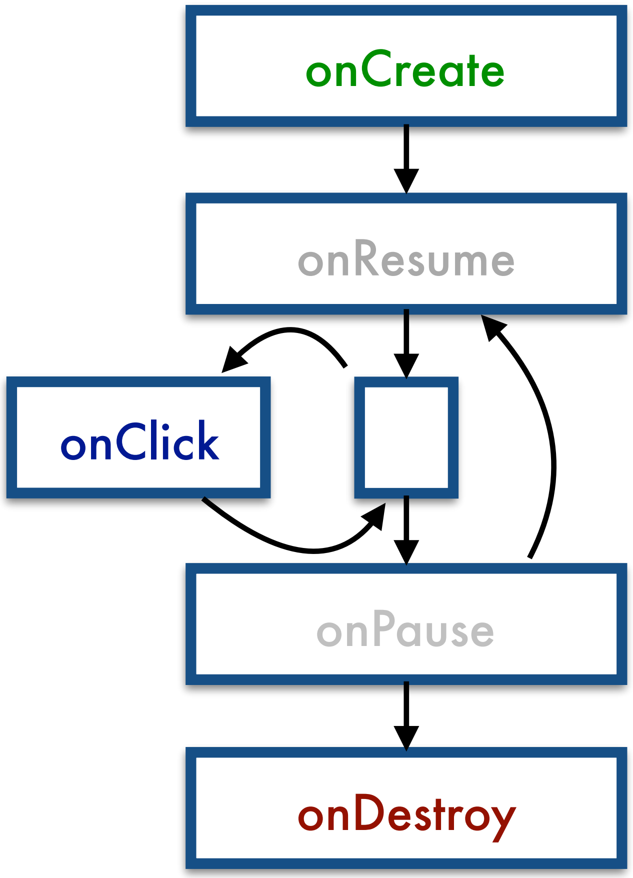
```
void onClick(…) {
  this.mHostDb.s(this.mService.g());
}
```

1 2

lifecycle constraints relevant

```
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();
}
```

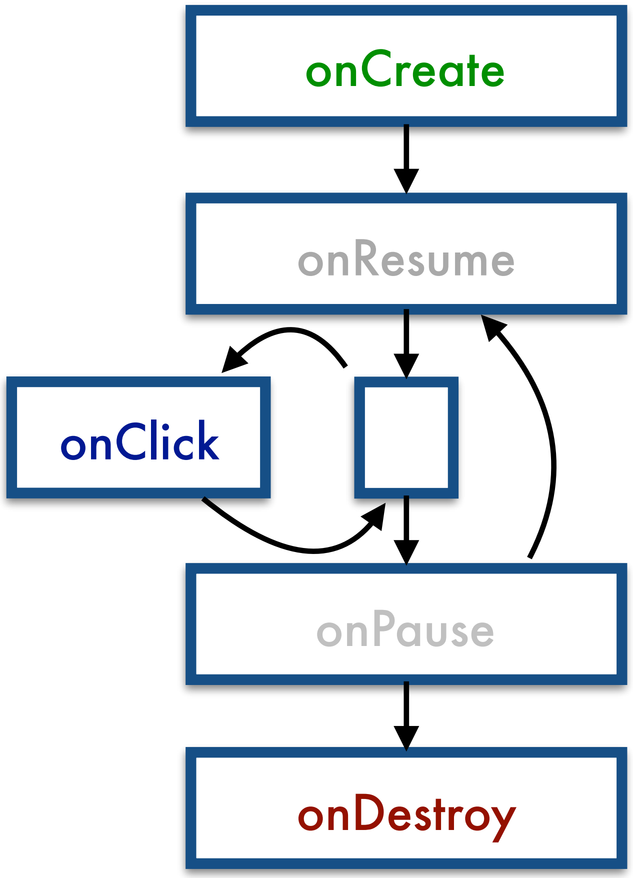
safe?

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

lifecycle constraints relevant

```
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();
}
```

safe?

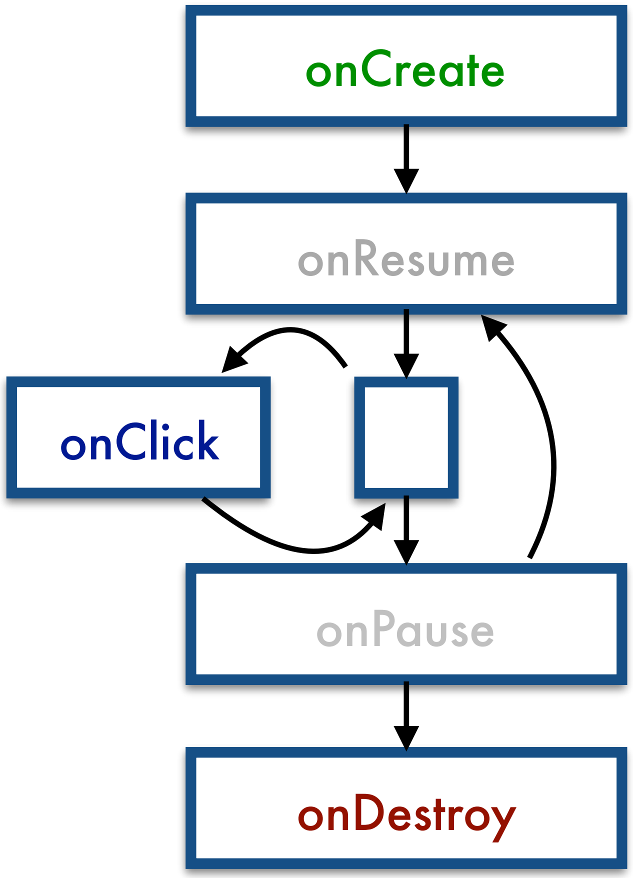
```
void onClick(…) {
  this.mHostDb.s(this.mService.g());
}
```

1 ✔ 2

lifecycle constraints relevant

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

Being smart …
Two dereferences: one safe and one buggy

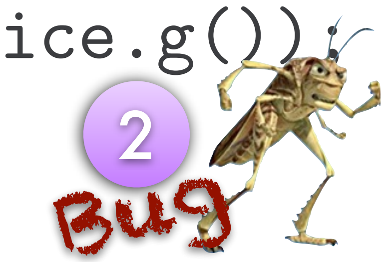



```
void onCreate() {
  bindService(…, new ServiceConn {
    void onConnected(@Nonnull Service s) {
      this.mService = s;
    }
  });
  this.mHostDb = new Db();
}

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

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

safe?

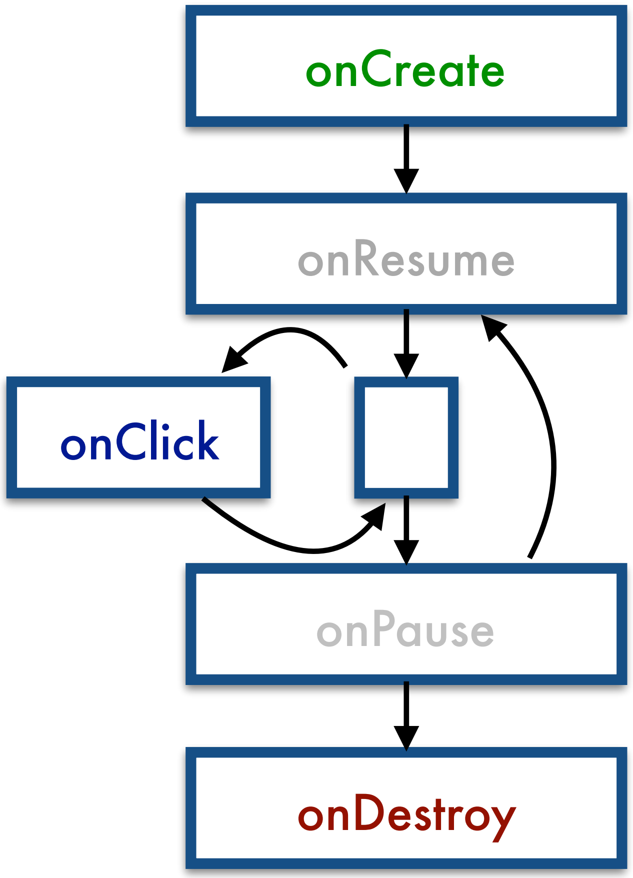
1 ✔

2



lifecycle constraints relevant

Being smart …
Two dereferences: one safe and one buggy



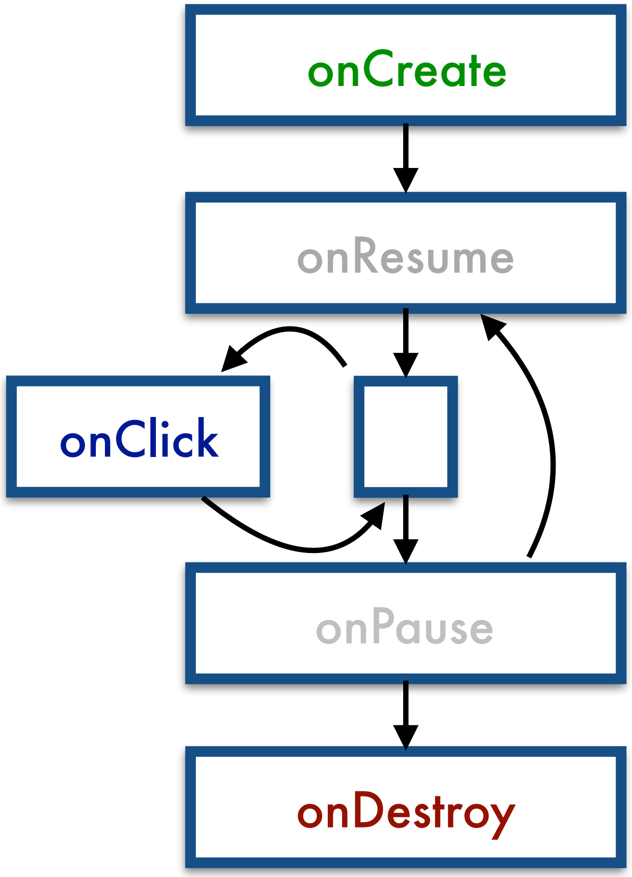

```
void onCreate() {
  bindService(…, new ServiceConn {
    void onConnected(@Nonnull Service s) {
      this.mService = s;
    }
  });
  this.mHostDb = new Db();
}

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

safe?

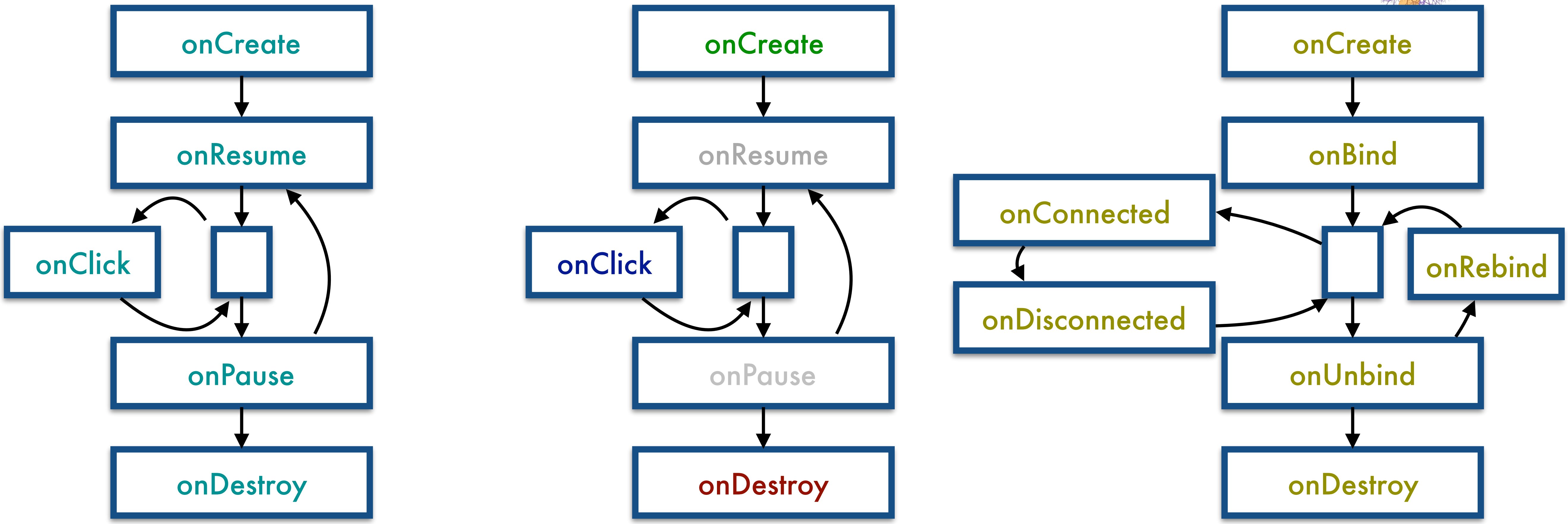
lifecy
constr
relev

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

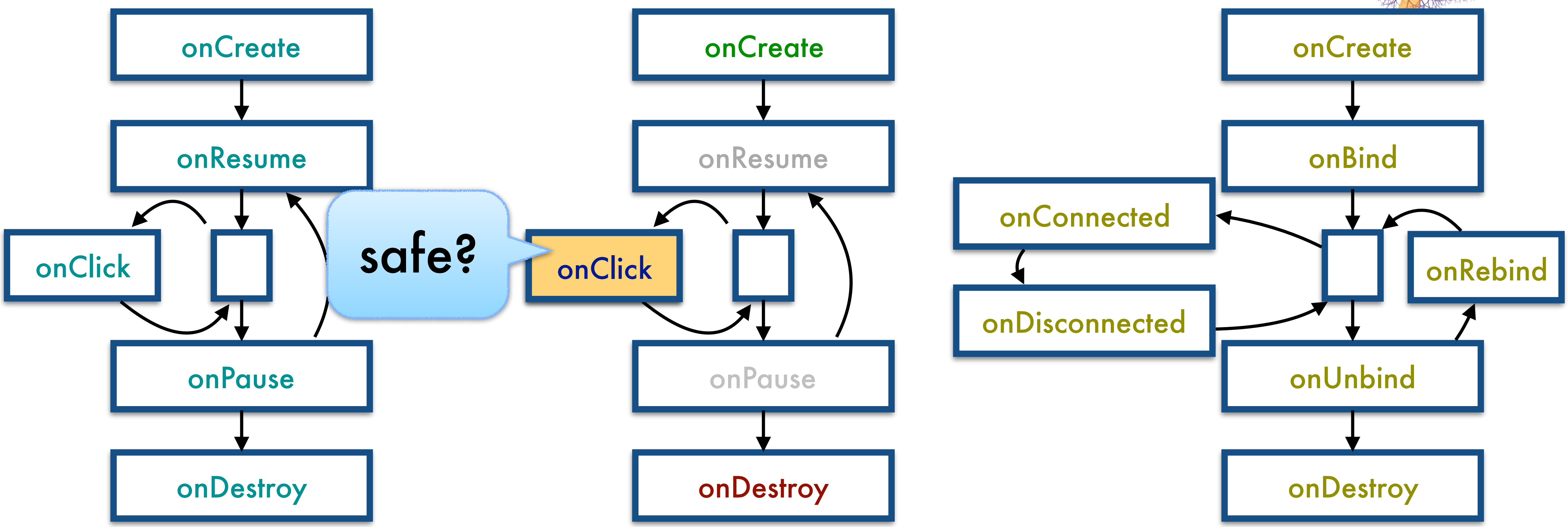
Idea: Jumping to relevant callbacks



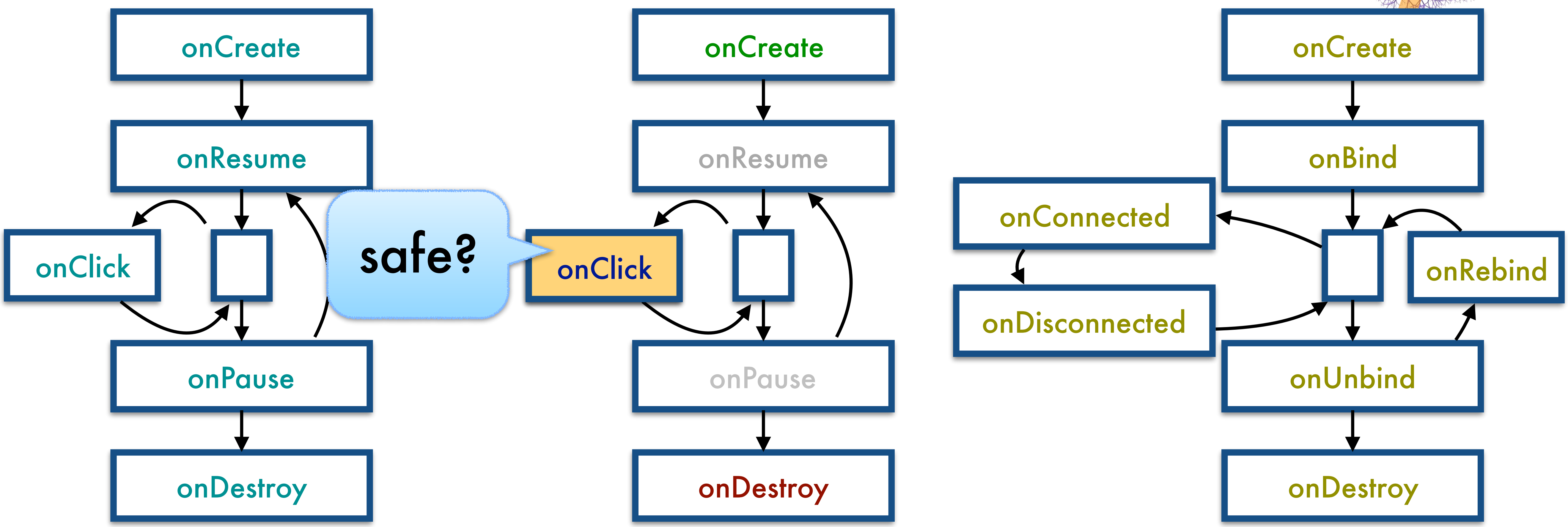
Idea: Jumping to relevant callbacks



Idea: Jumping to relevant callbacks

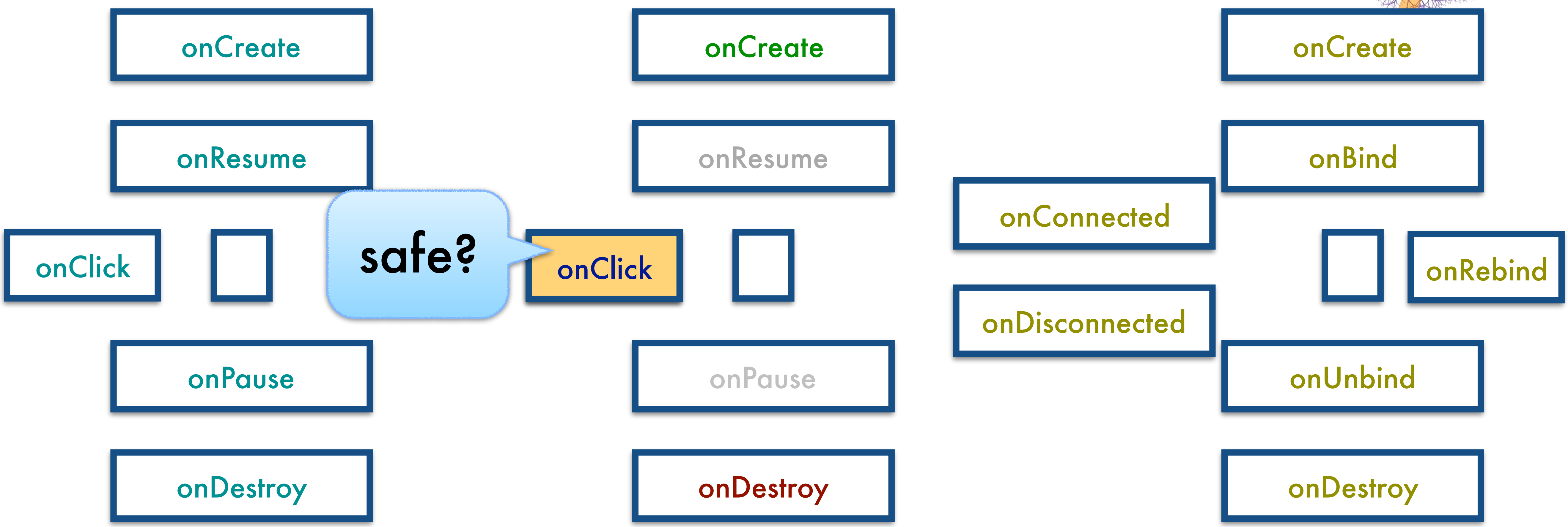


Idea: Jumping to relevant callbacks



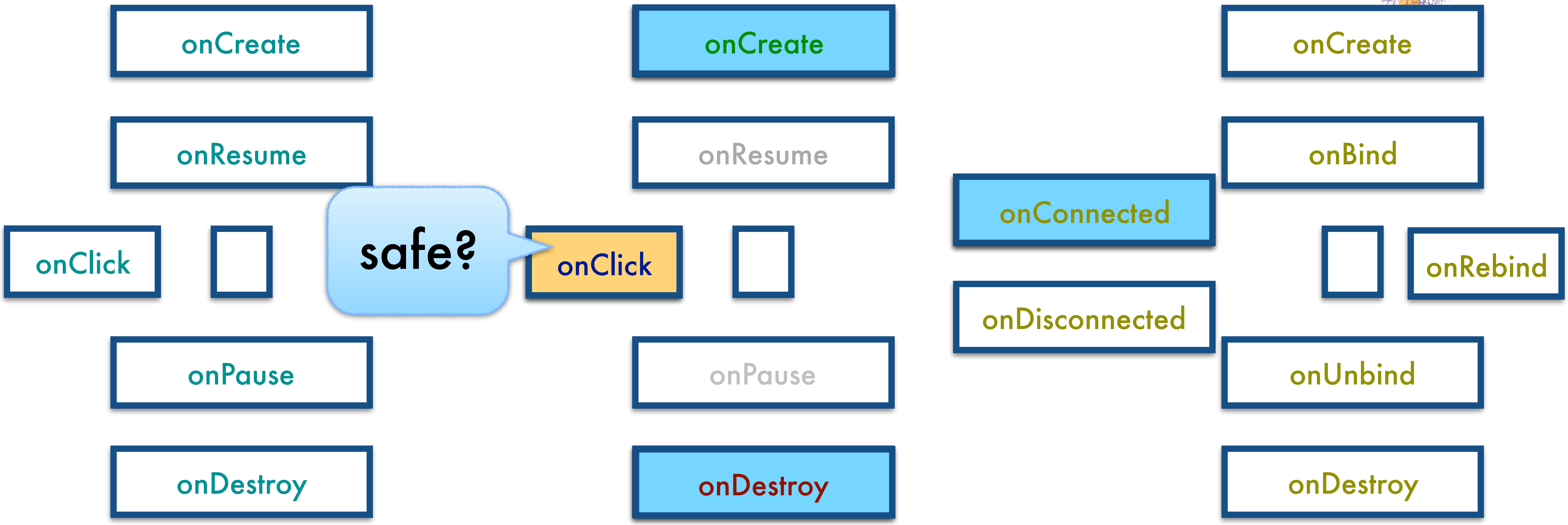
Find data-relevant callbacks

Idea: Jumping to relevant callbacks



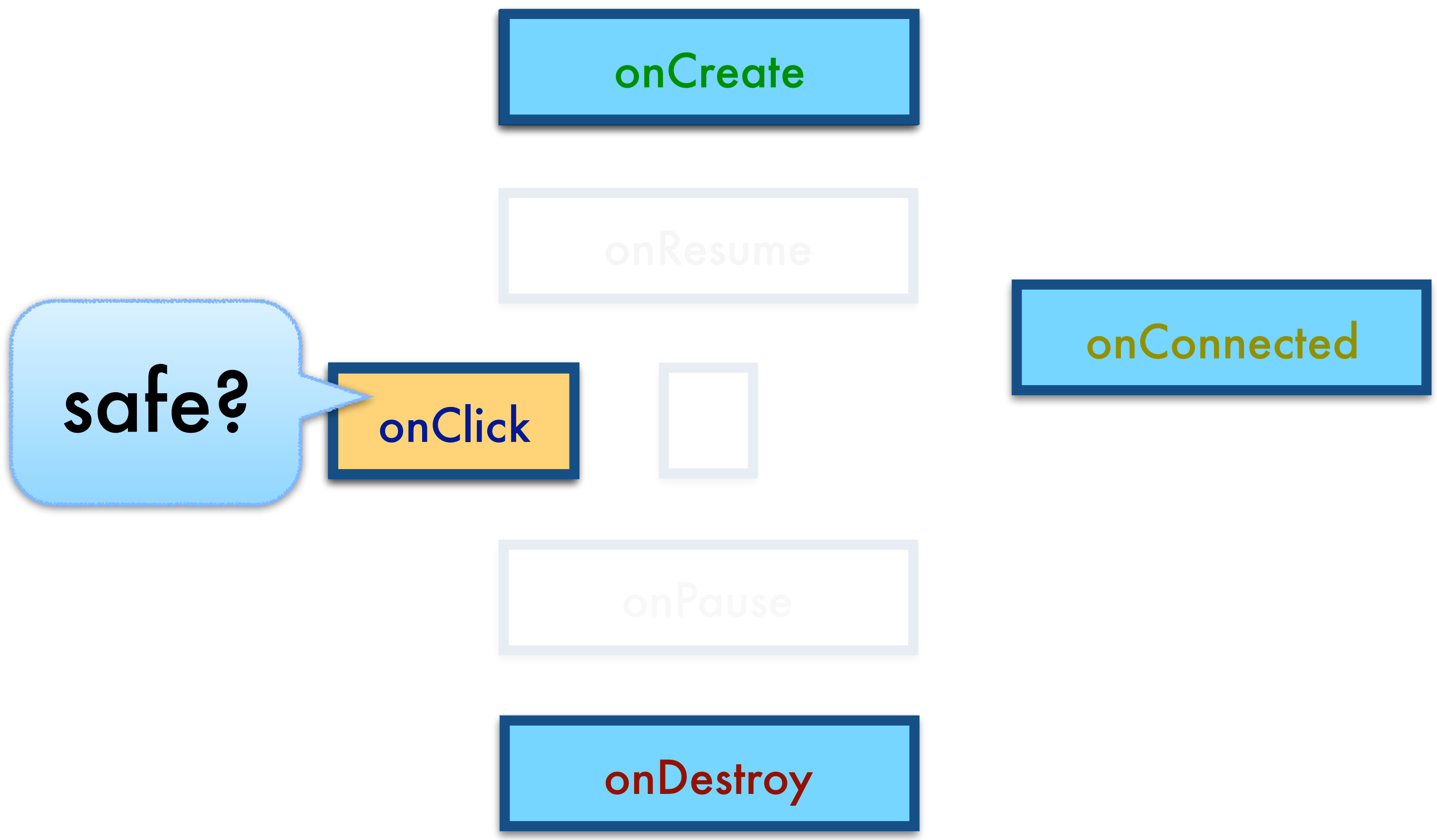
Find data-relevant callbacks

Idea: Jumping to relevant callbacks



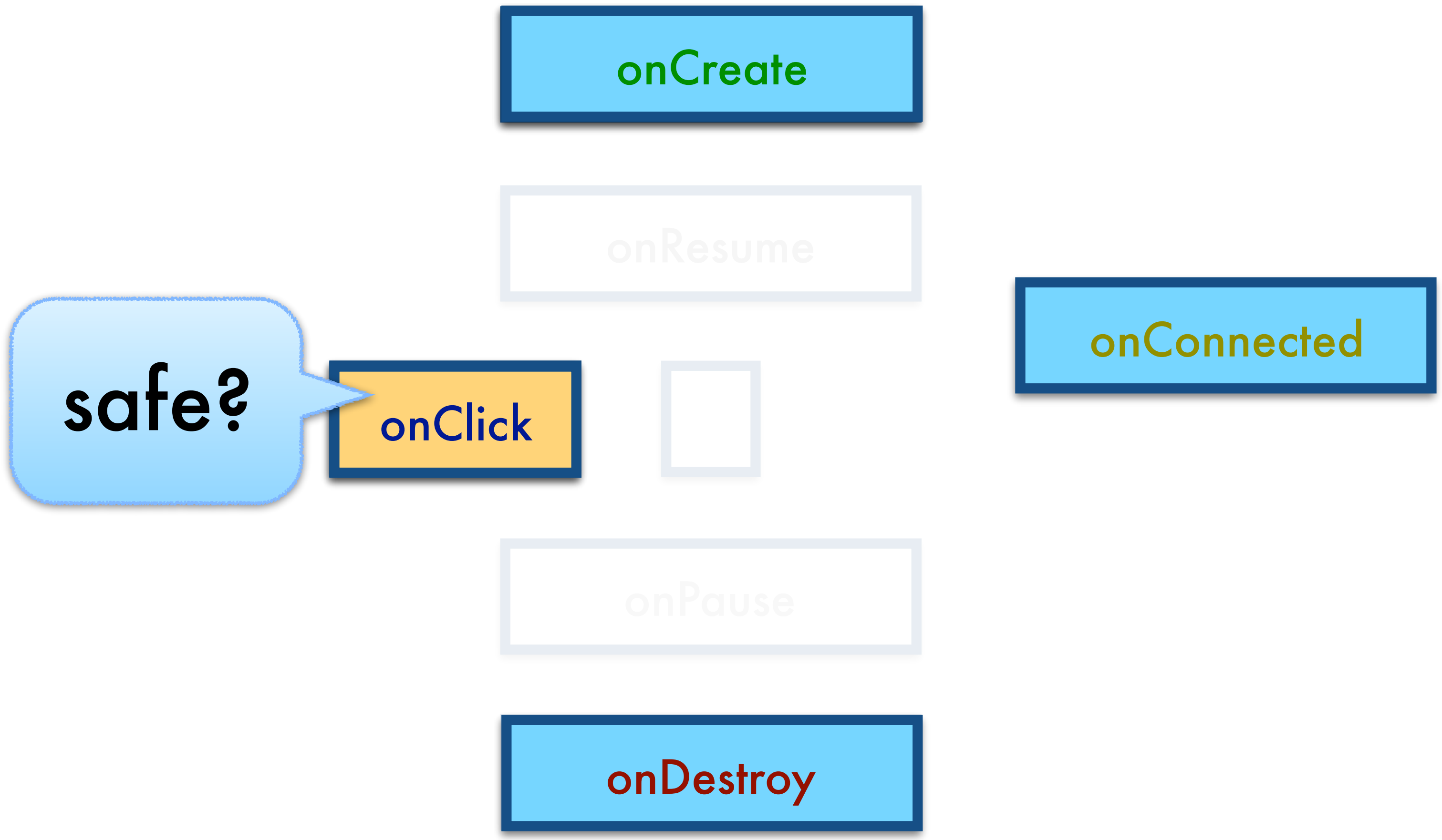
Find data-relevant callbacks

Idea: Jumping to relevant callbacks



Find data-relevant callbacks

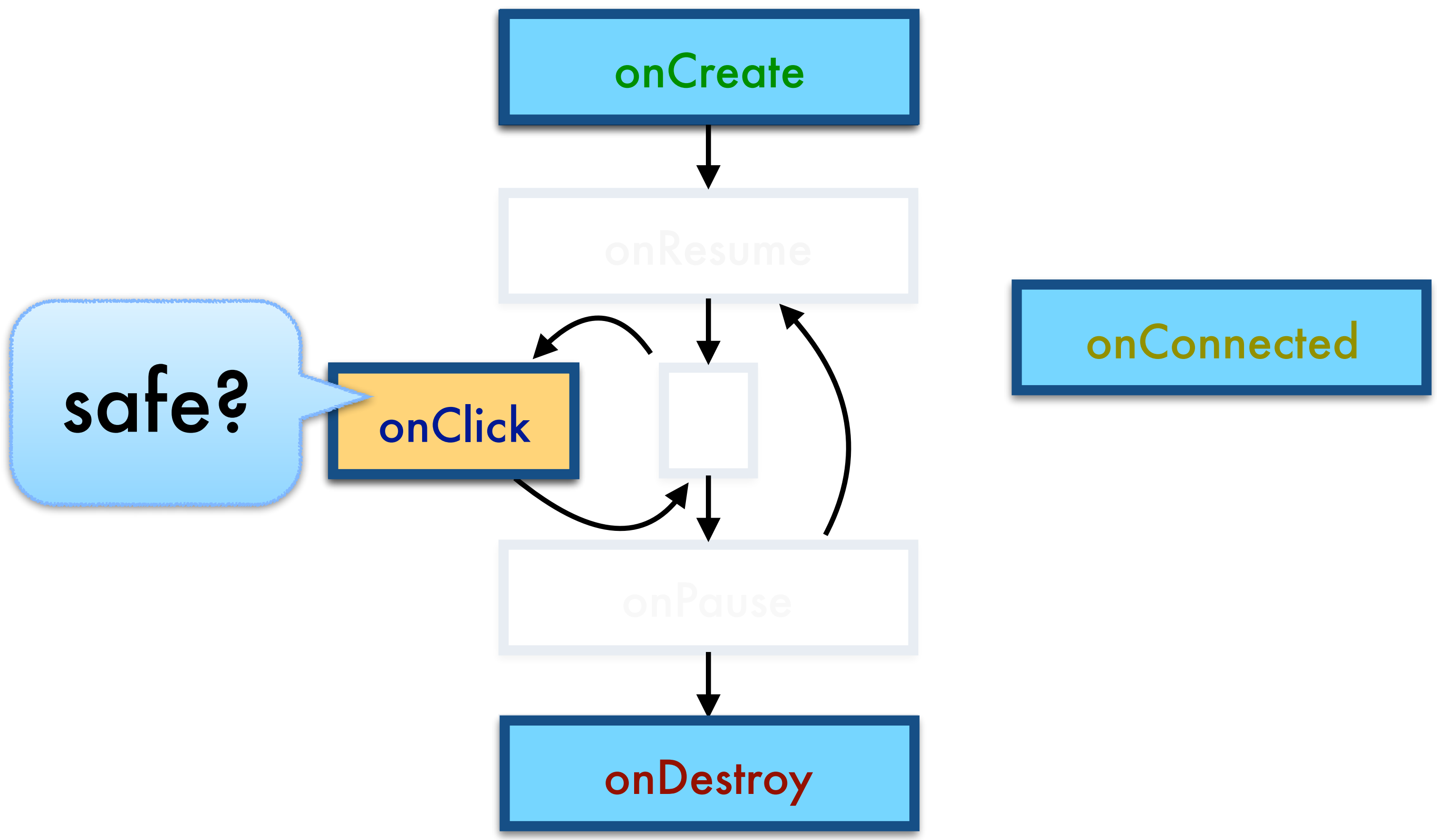
Idea: Jumping to relevant callbacks



Find data-relevant callbacks

Filter using control-feasibility

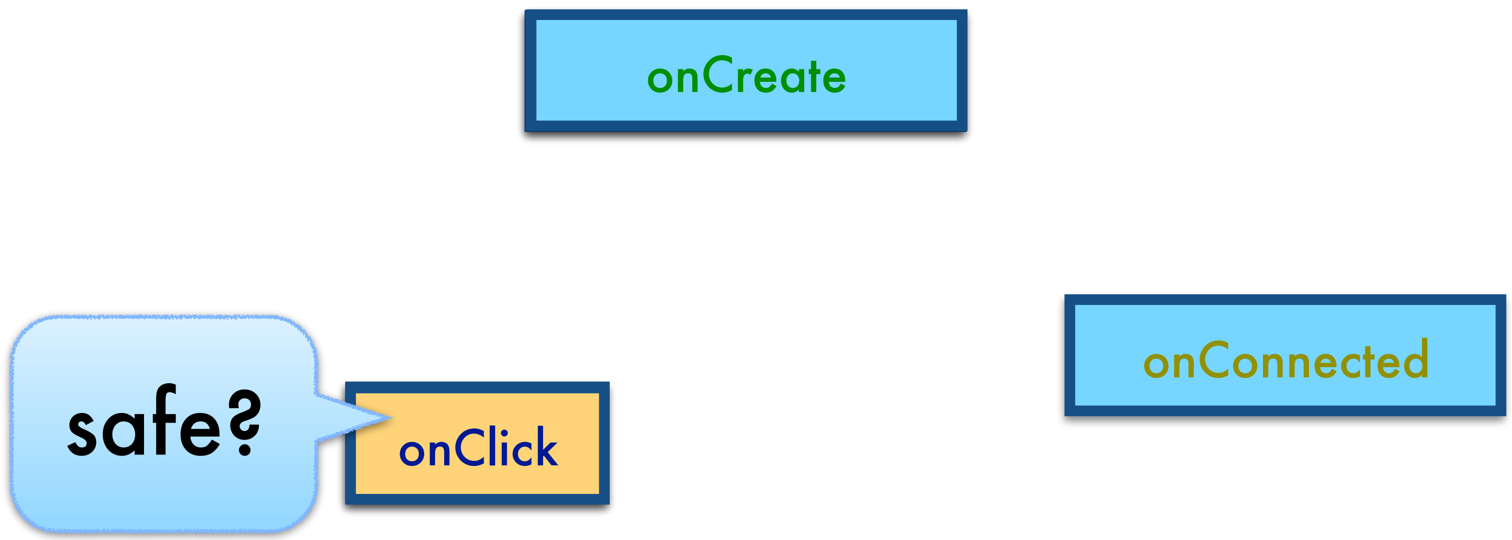
Idea: Jumping to relevant callbacks



Find data-relevant callbacks

Filter using control-feasibility

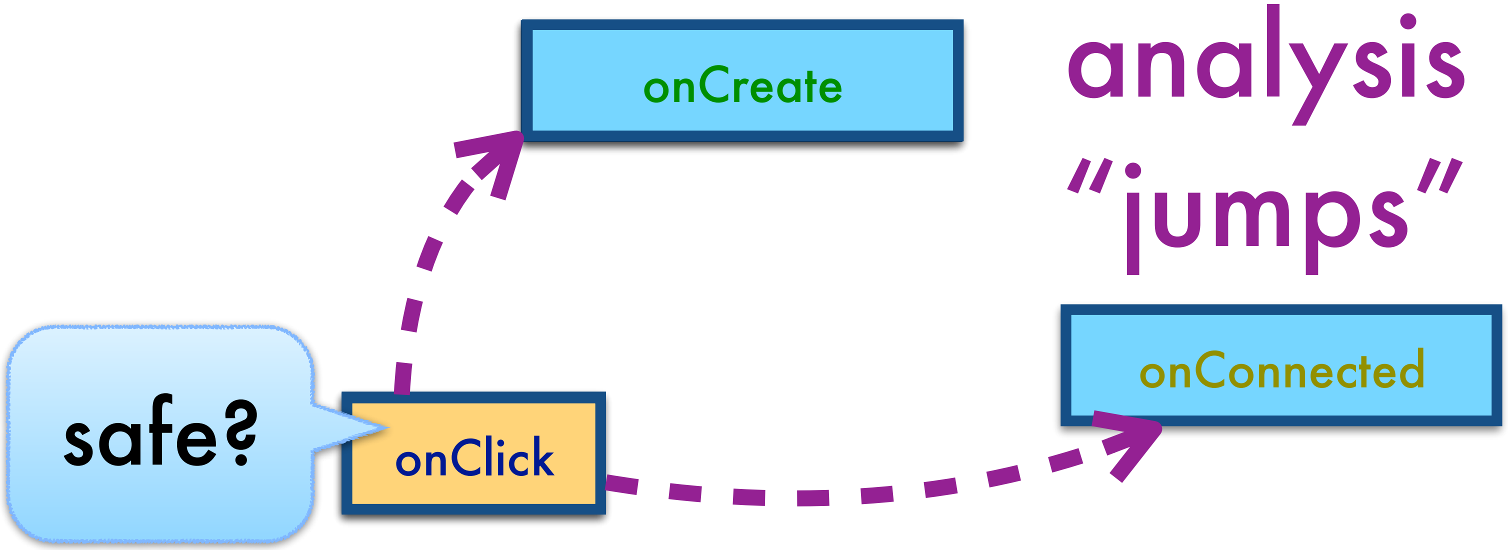
Idea: Jumping to relevant callbacks



Find data-relevant callbacks

Filter using control-feasibility

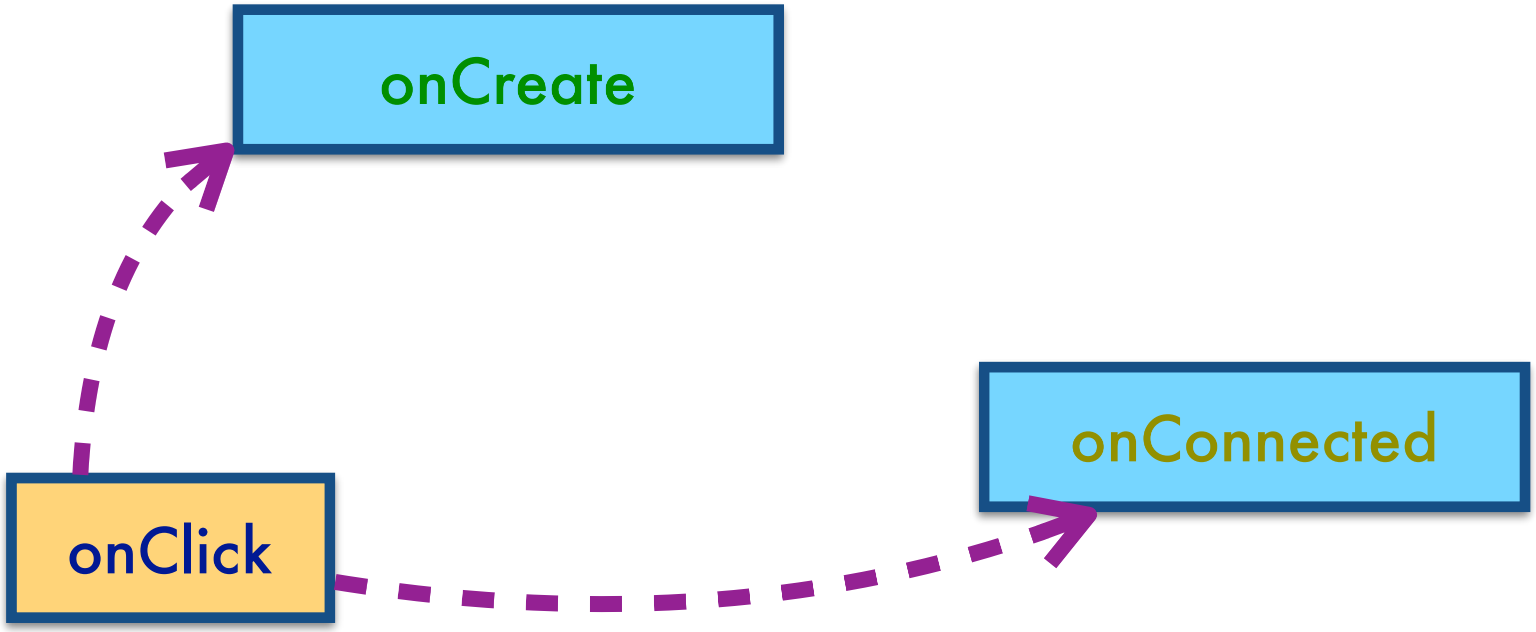
Idea: Jumping to relevant callbacks



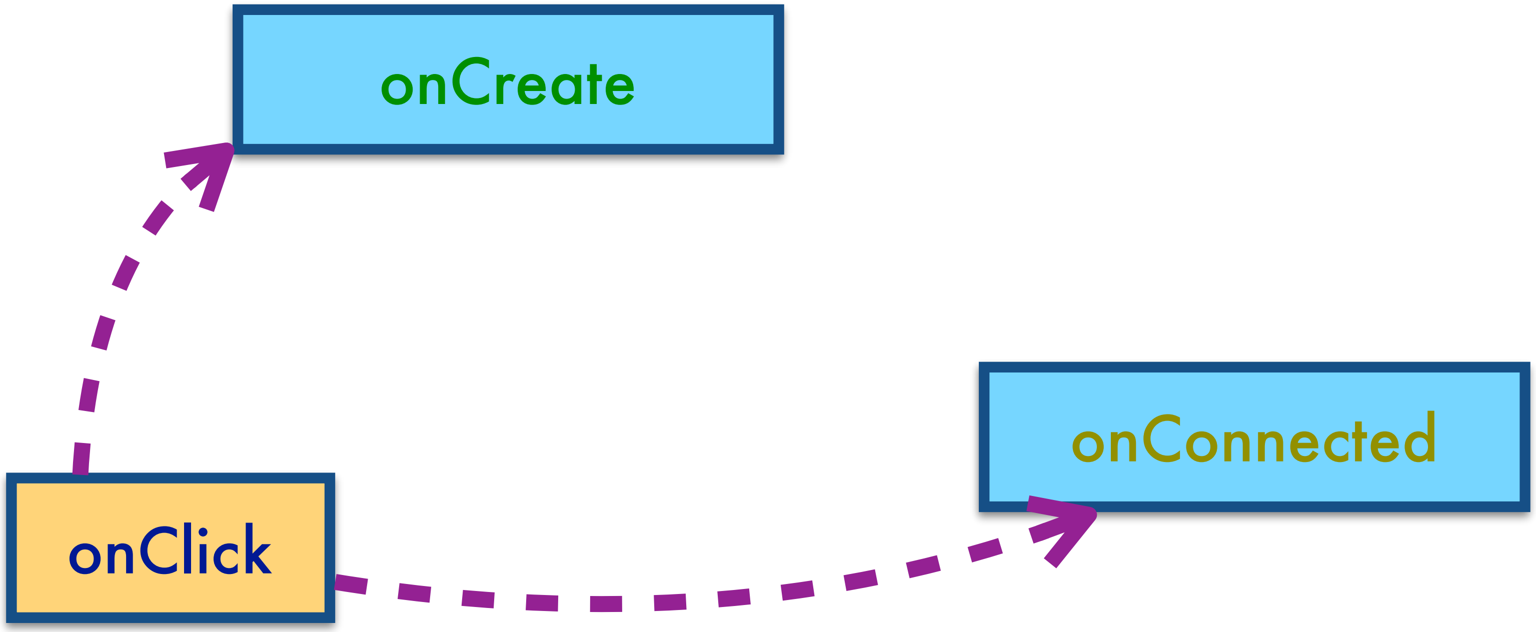
Find data-relevant callbacks

Filter using control-feasibility

Contributions: Hopper is an analysis that jumps

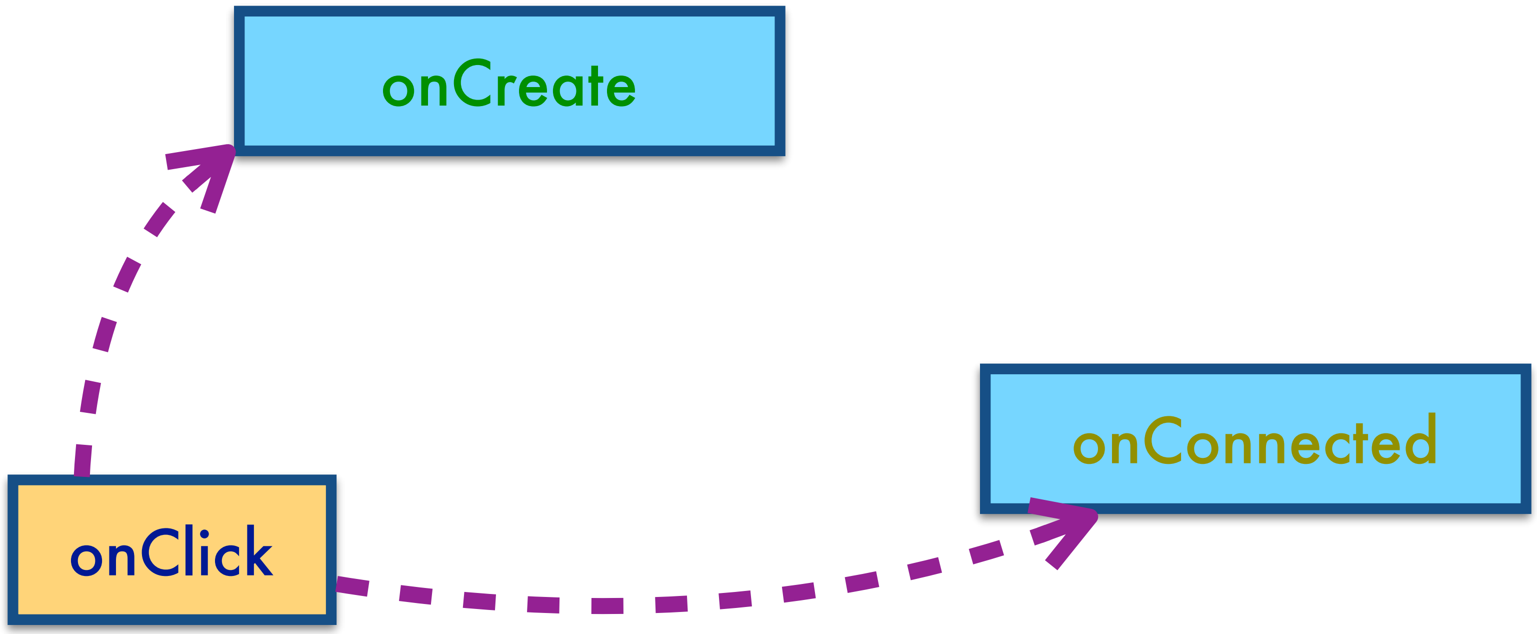


Contributions: Hopper is an analysis that jumps



1 $\langle Q, \ell \rangle \rightsquigarrow T$ Framework for sound jumping analyses

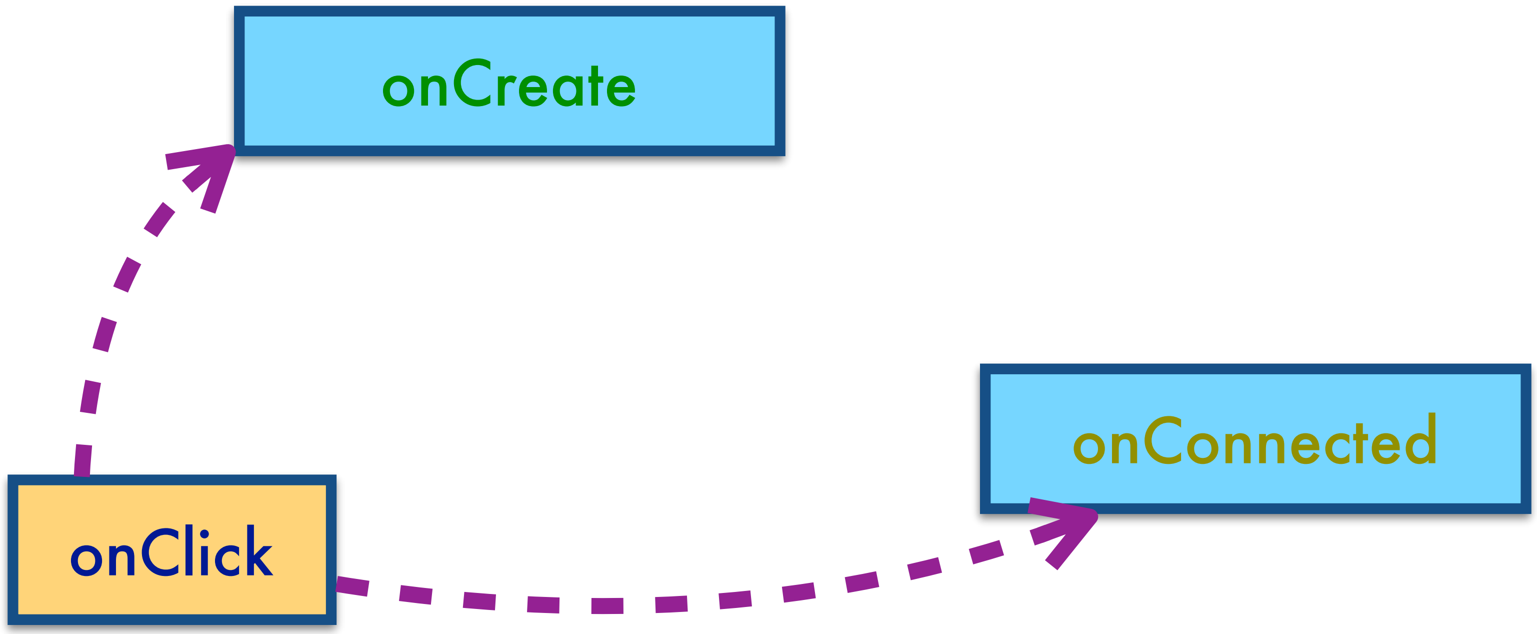
Contributions: Hopper is an analysis that jumps



1 $\langle Q, \ell \rangle \rightsquigarrow T$ Framework for sound jumping analyses

2  Applied to Android lifecycles

Contributions: Hopper is an analysis that jumps



1 $\langle Q, \ell \rangle \rightsquigarrow T$ **Framework for sound jumping analyses**

Applied to Android lifecycles

Interleave data-relevance with control-feasibility to realize jumping

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

Interleave data-relevance with control-feasibility to realize jumping

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

ℓ

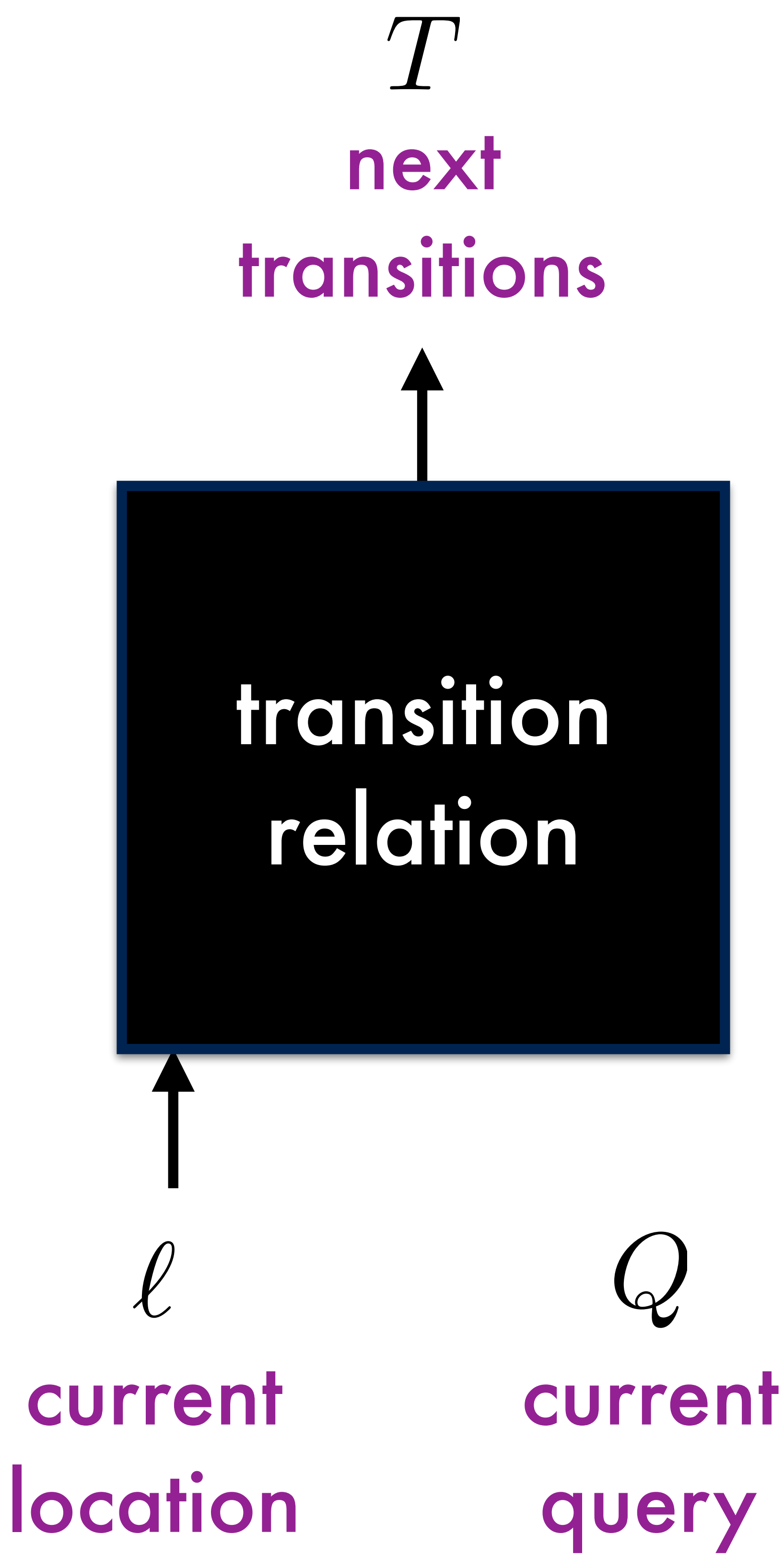
current location

Q

current query

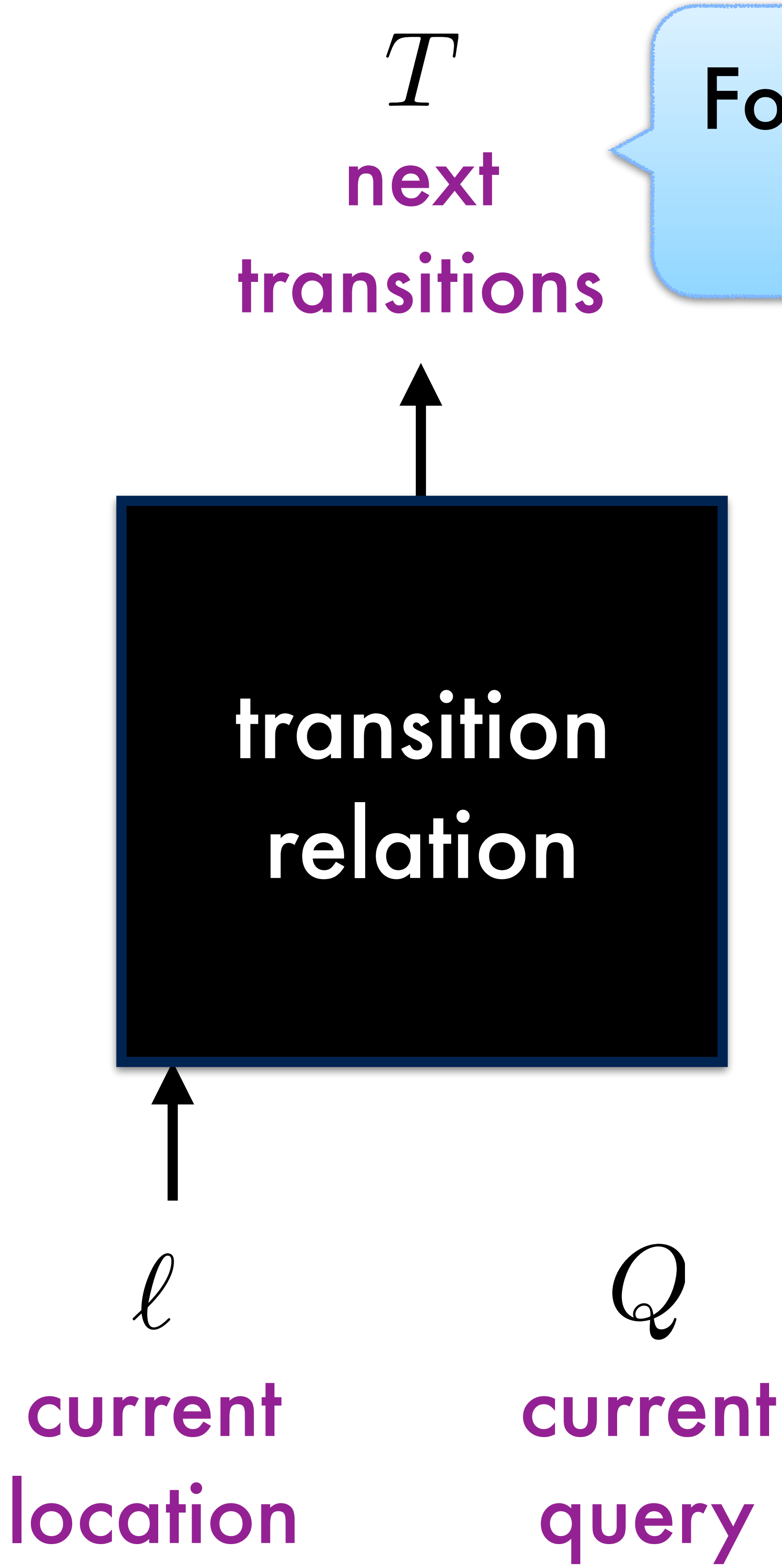
Interleave data-relevance with control-feasibility to realize jumping

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



Interleave data-relevance with control-feasibility to realize jumping

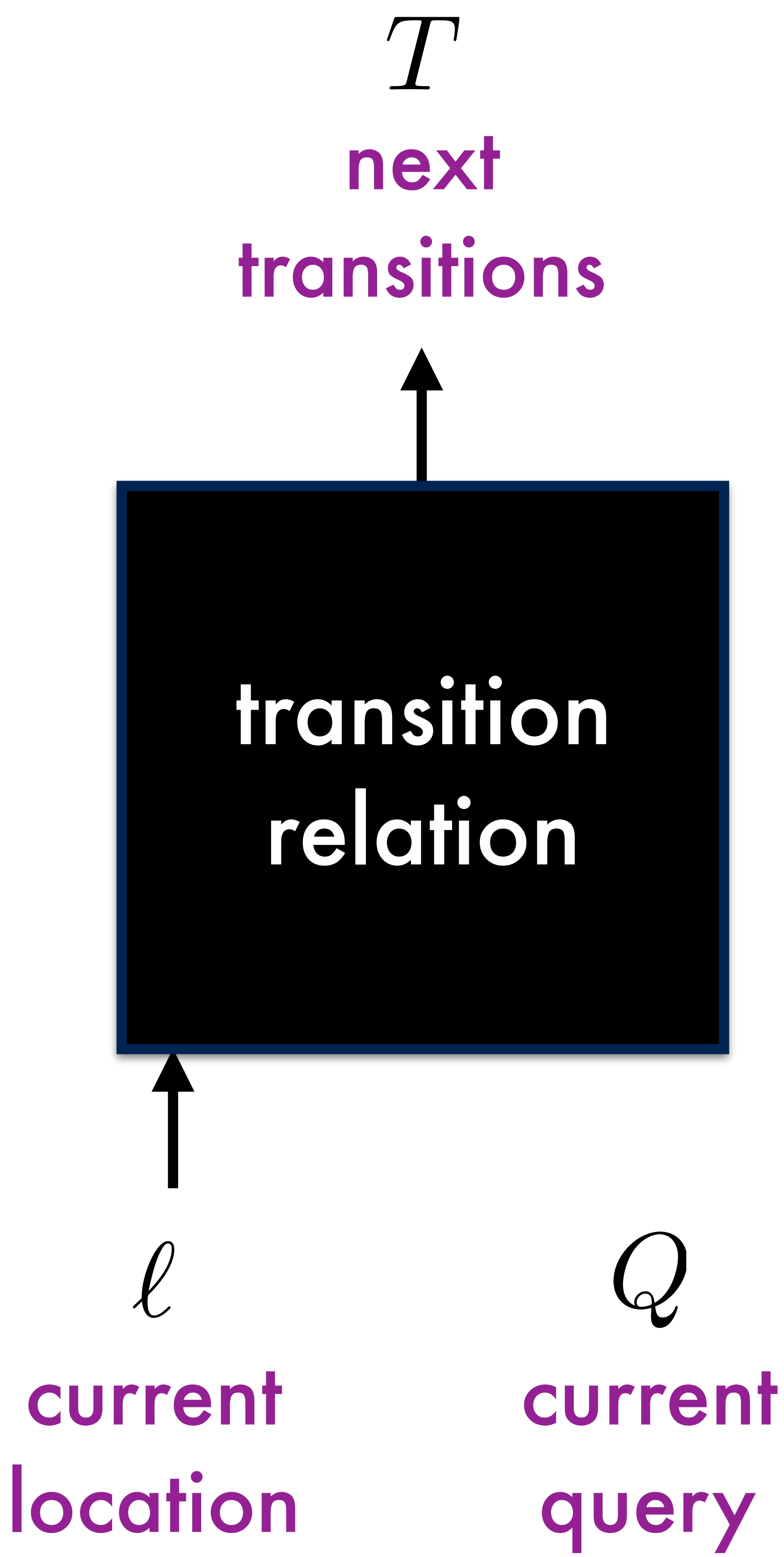
$\langle Q, \ell \rangle \rightsquigarrow T$



Follow the normal control-flow of the the program

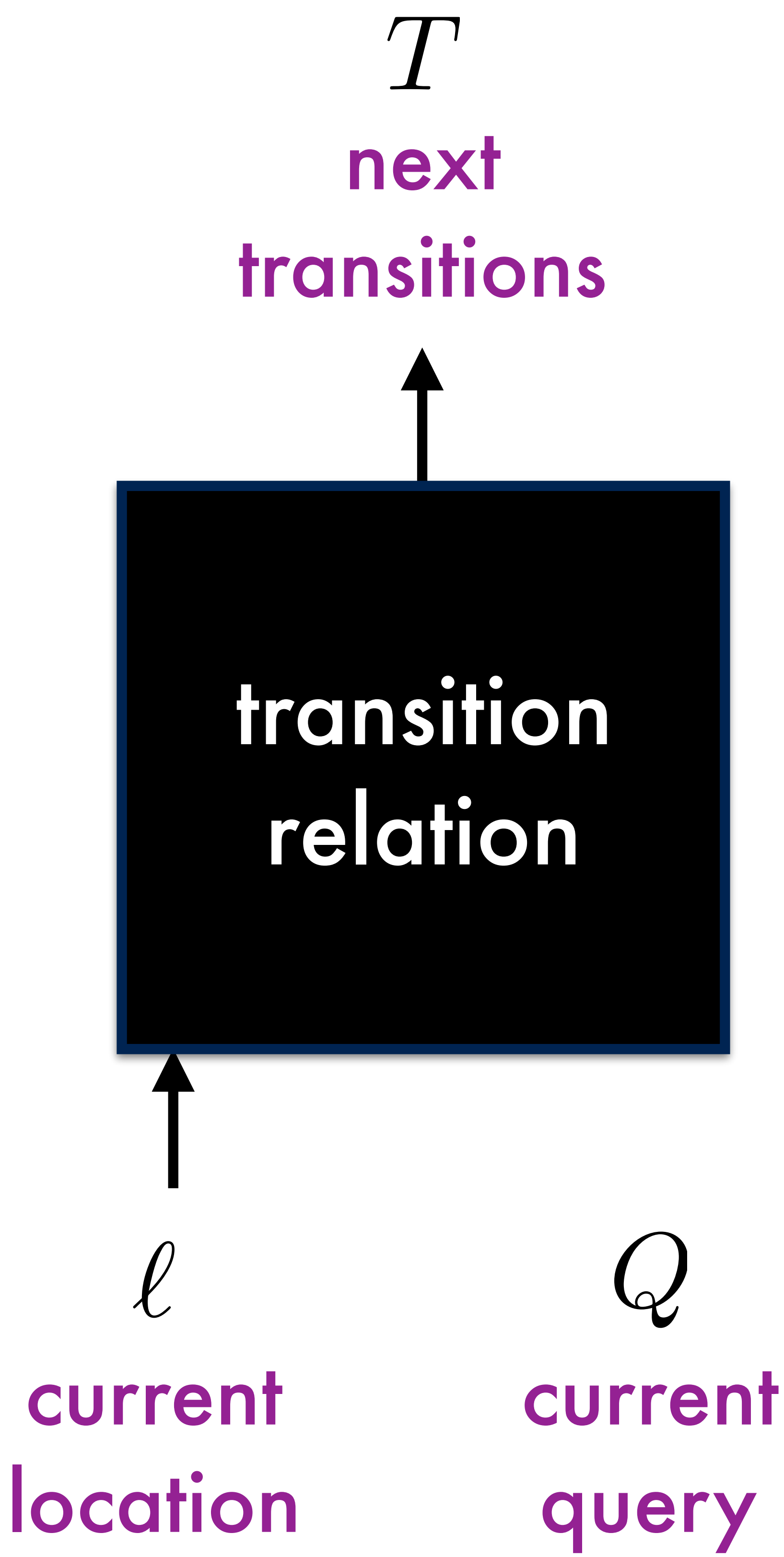
Interleave data-relevance with control-feasibility to realize jumping

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



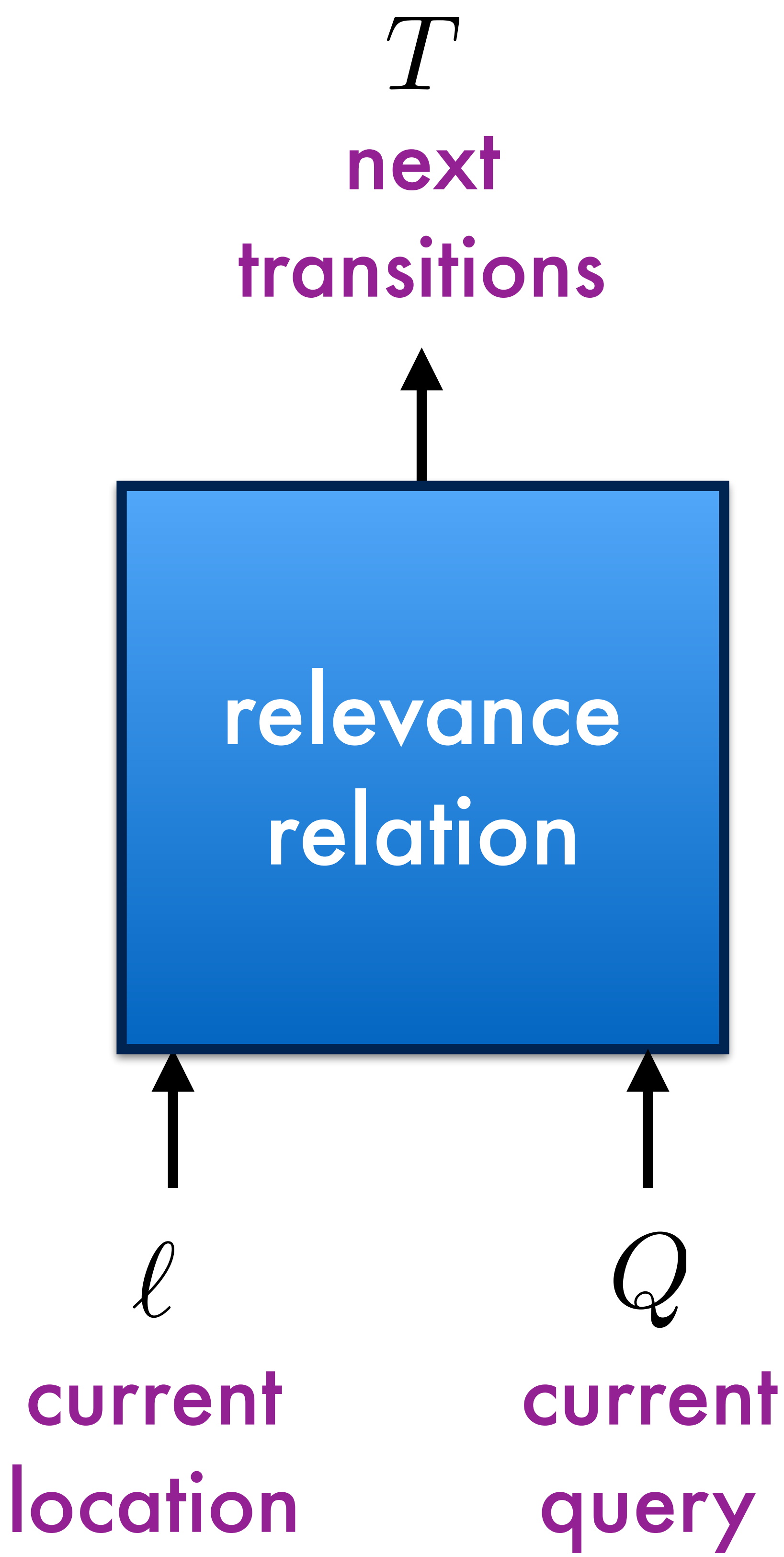
Interleave data-relevance with control-feasibility to realize jumping

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



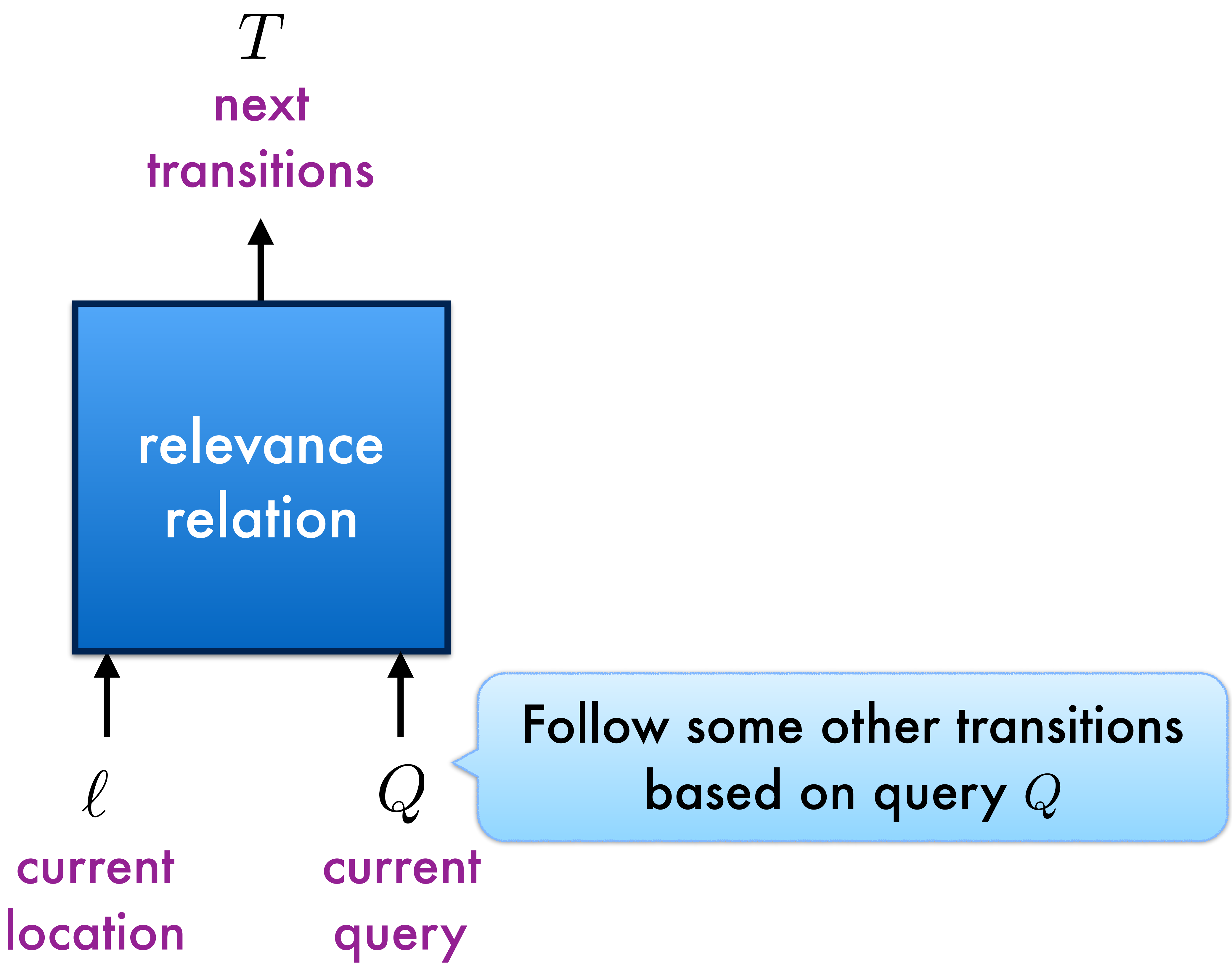
Interleave data-relevance with control-feasibility to realize jumping

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



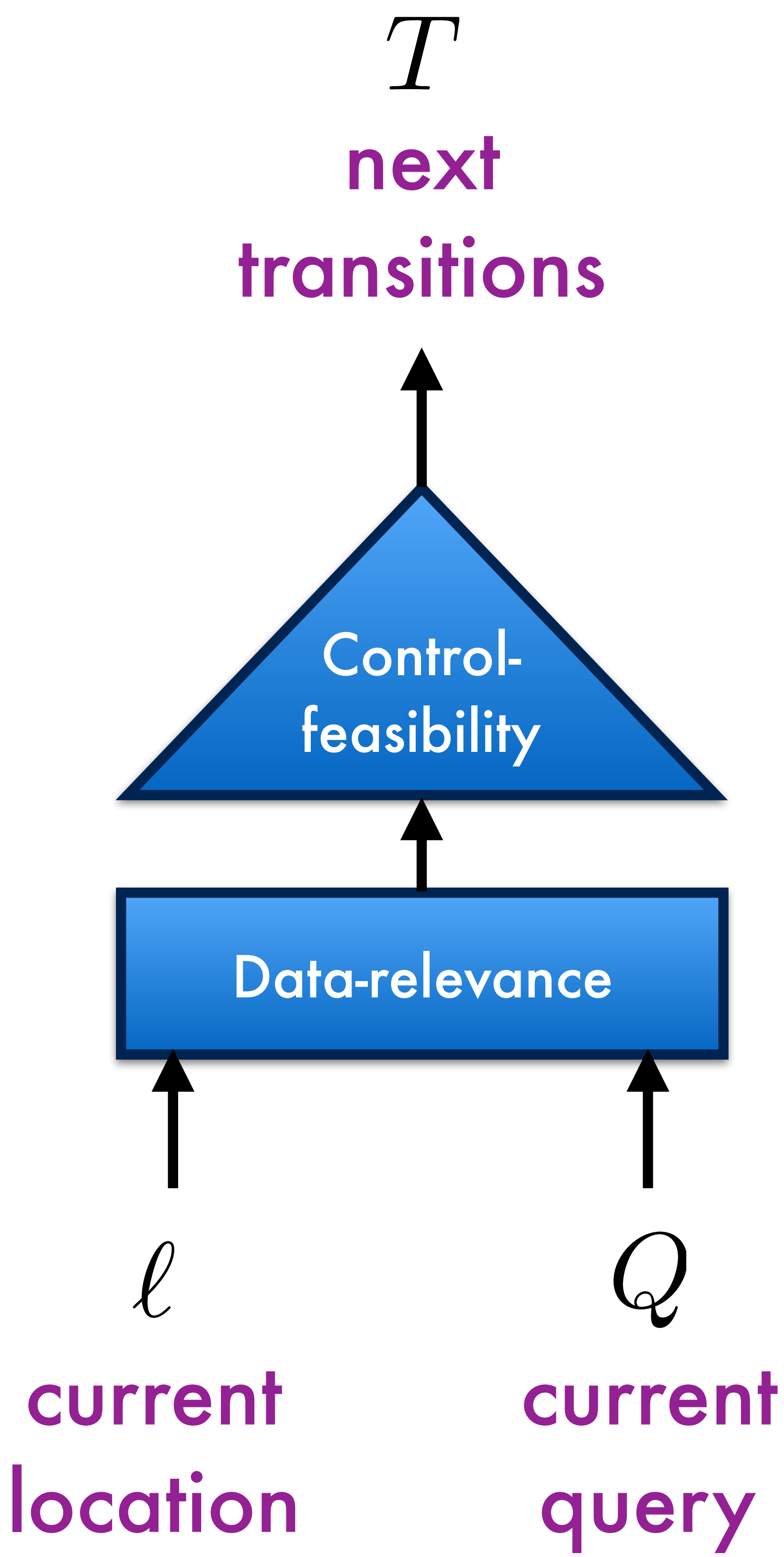
Interleave data-relevance with control-feasibility to realize jumping

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



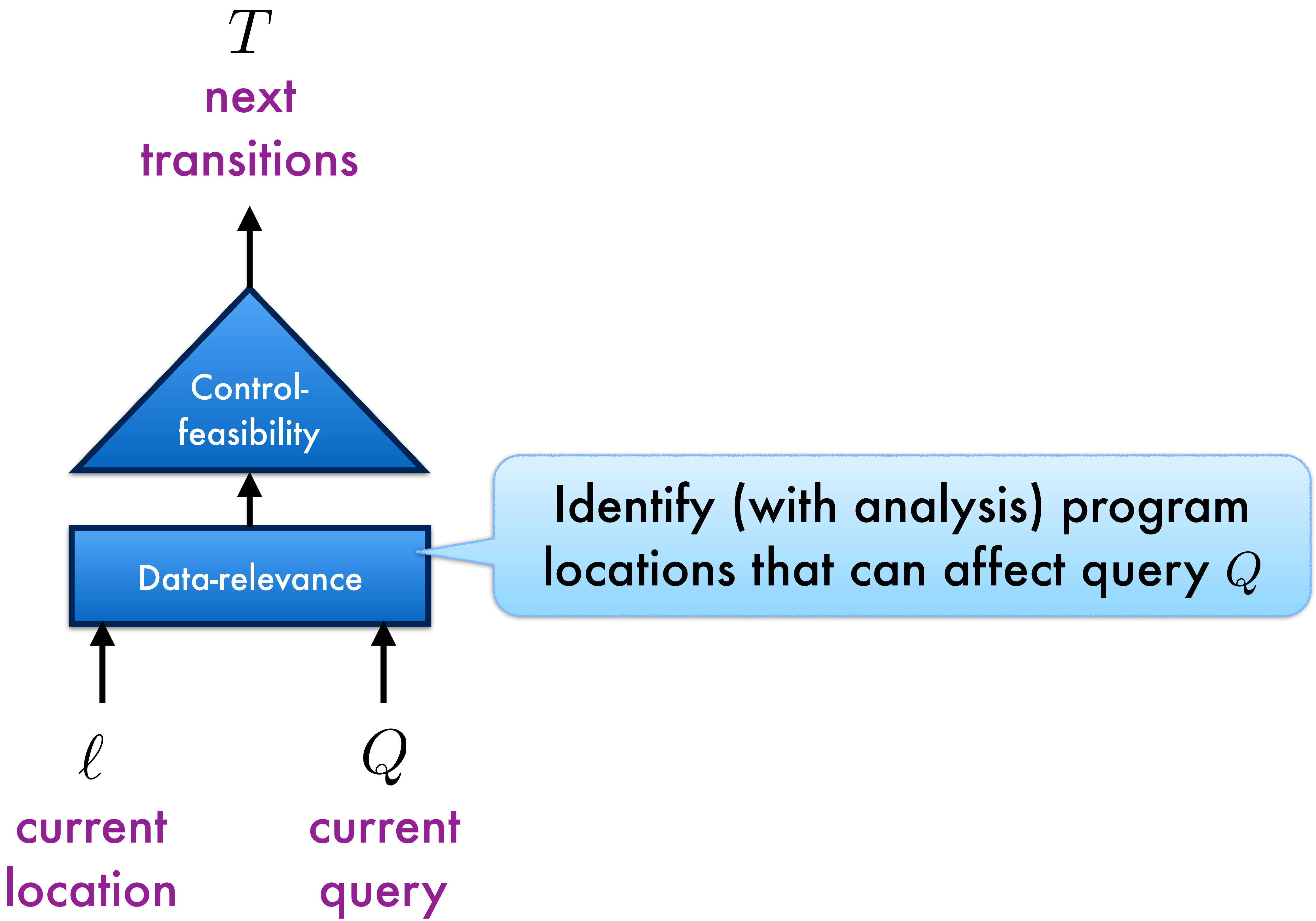
Interleave data-relevance with control-feasibility to realize jumping

1 $\langle Q, \ell \rangle \leadsto T$



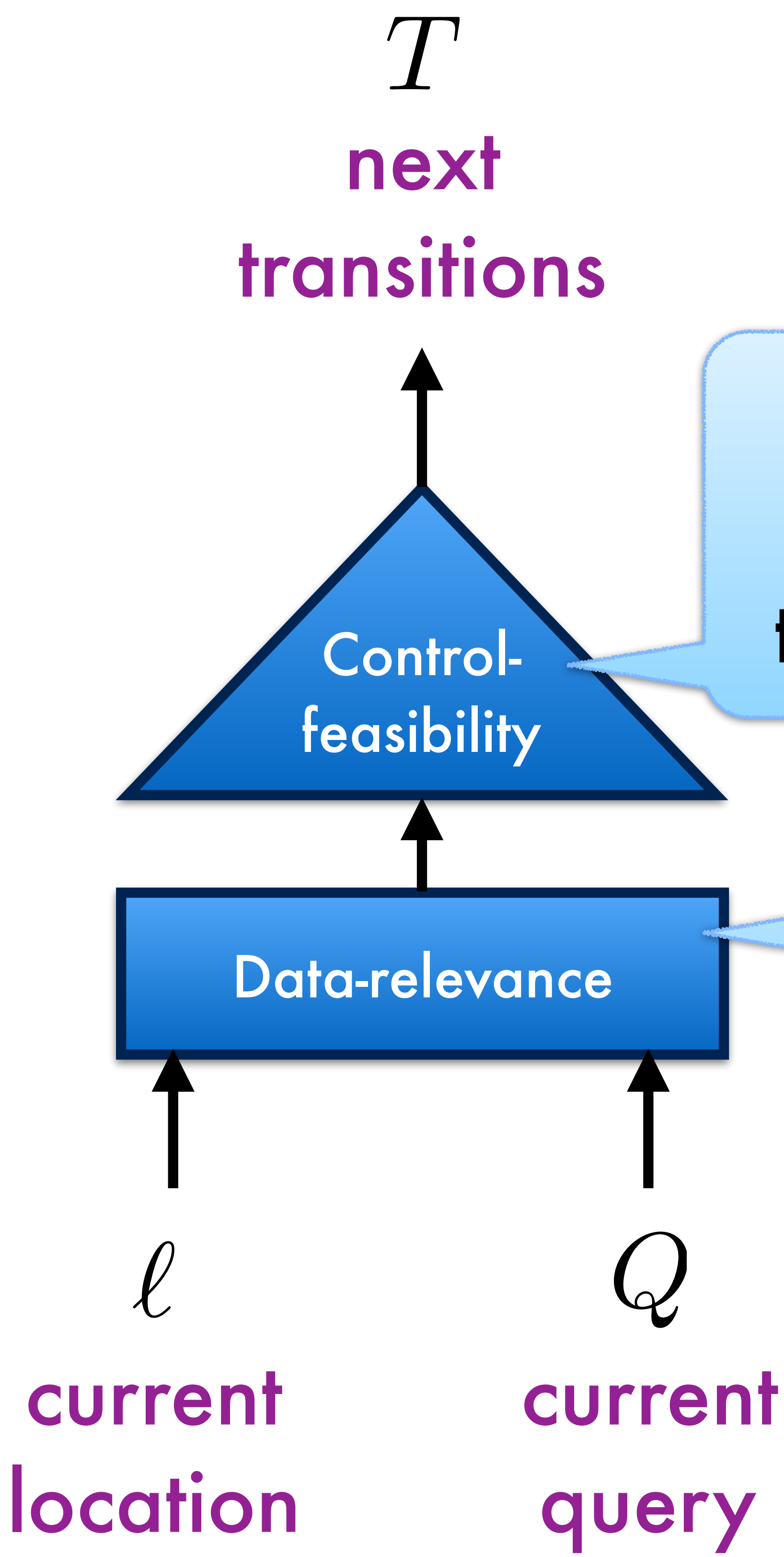
Interleave data-relevance with control-feasibility to realize jumping

$$\textcircled{1}\ \langle Q, \ell \rangle \rightsquigarrow T$$



Interleave data-relevance with control-feasibility to realize jumping

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

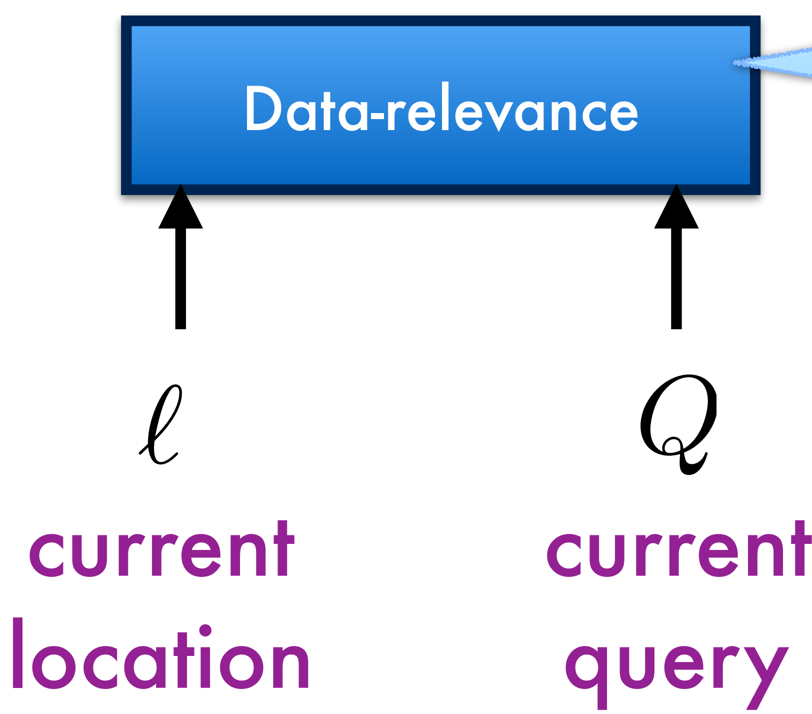


Filter to the locations that can feasibly reach ℓ (without "going through" any other transition in T)

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

Data-relevance identifies relevant writes

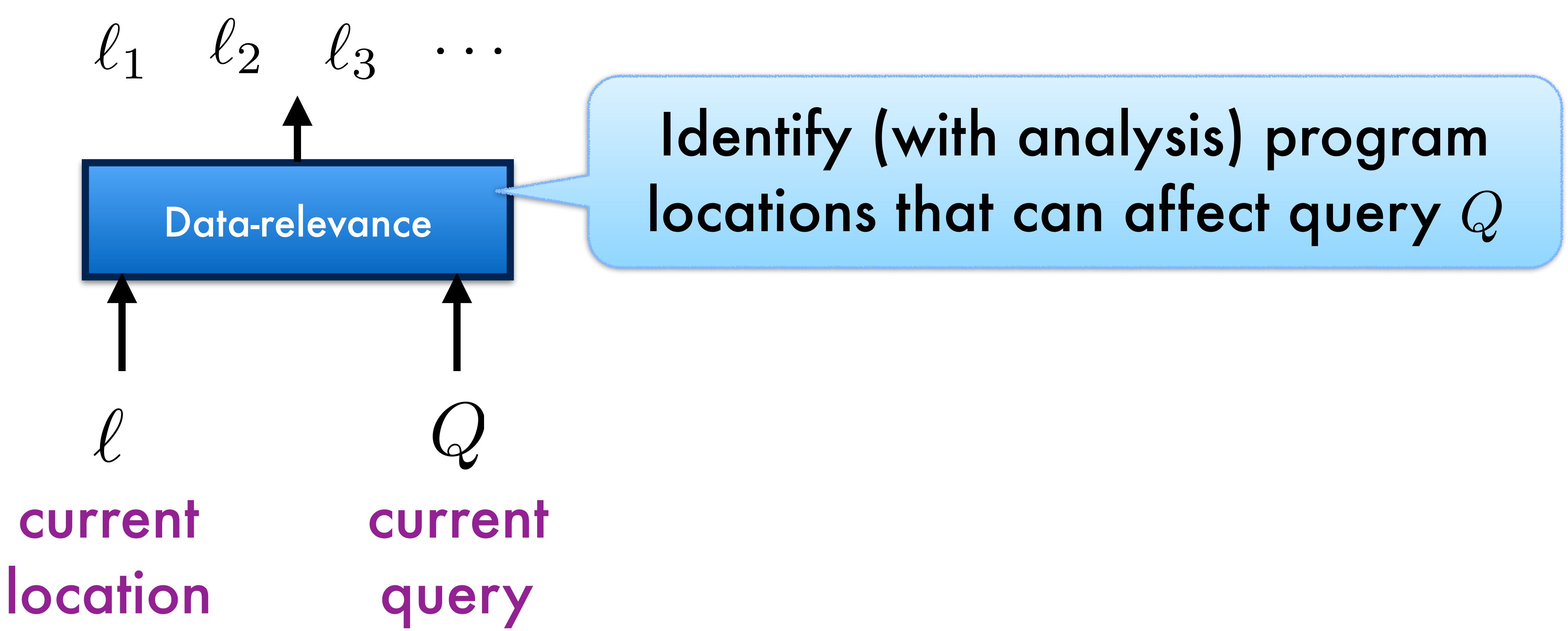
1 $\langle Q, \ell \rangle \rightsquigarrow T$



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

Data-relevance identifies relevant writes

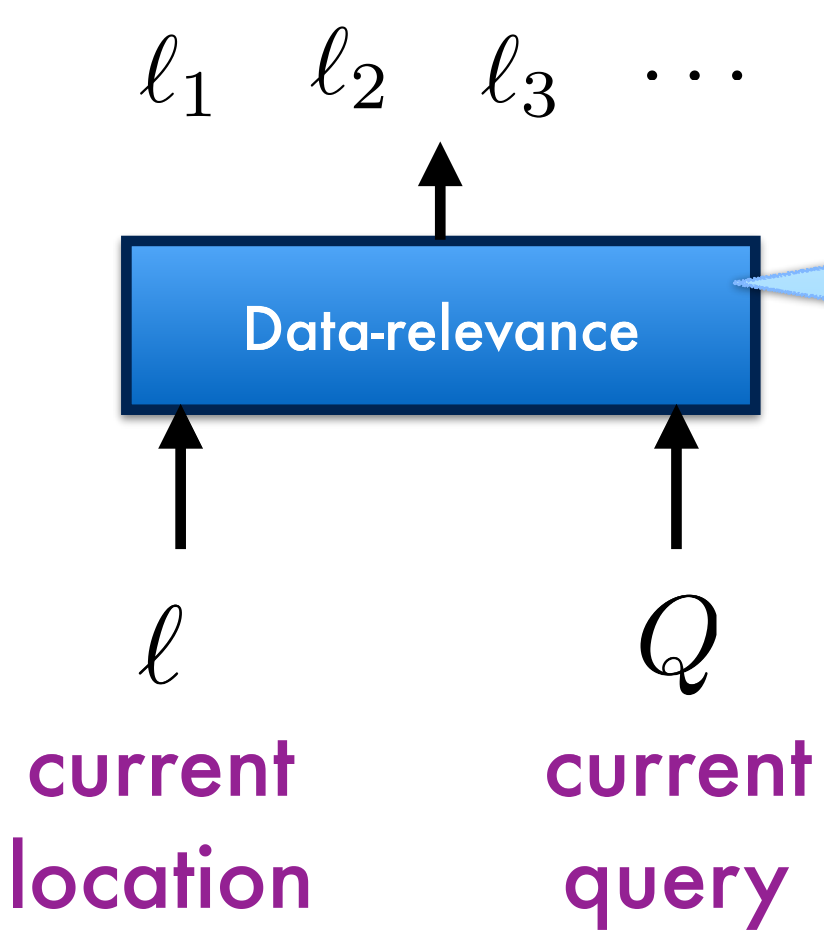
1 $\langle Q, \ell \rangle \rightsquigarrow T$



Data-relevance identifies relevant writes

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

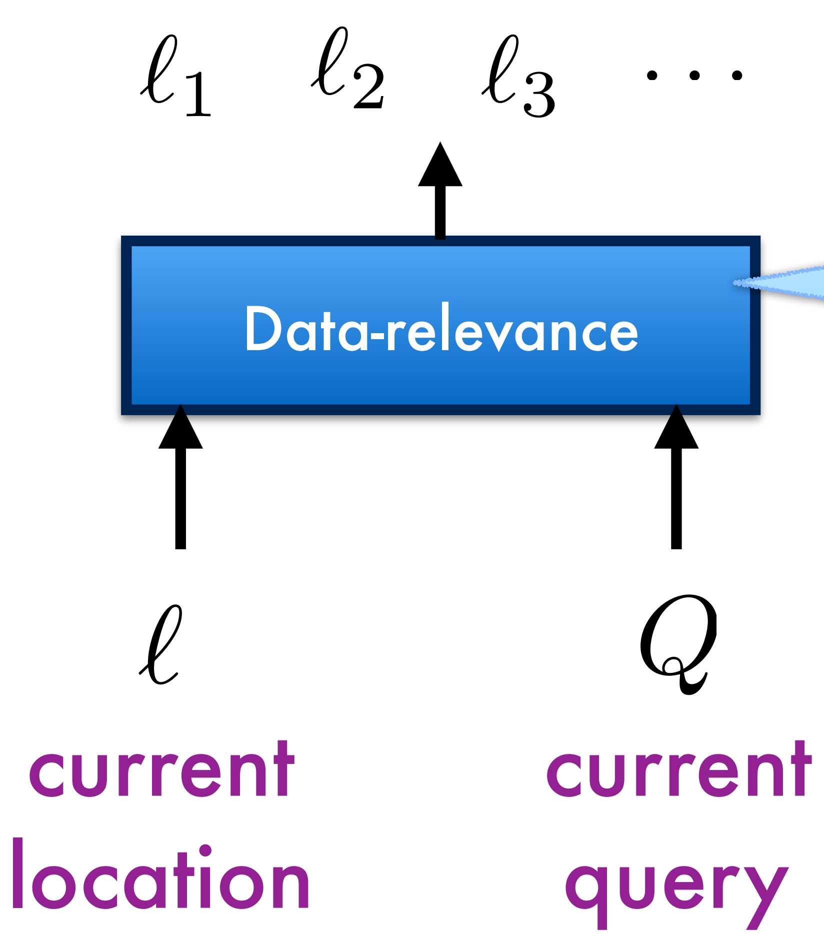
Computed using pre-pass points-to analysis, types, field-based, ...



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

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

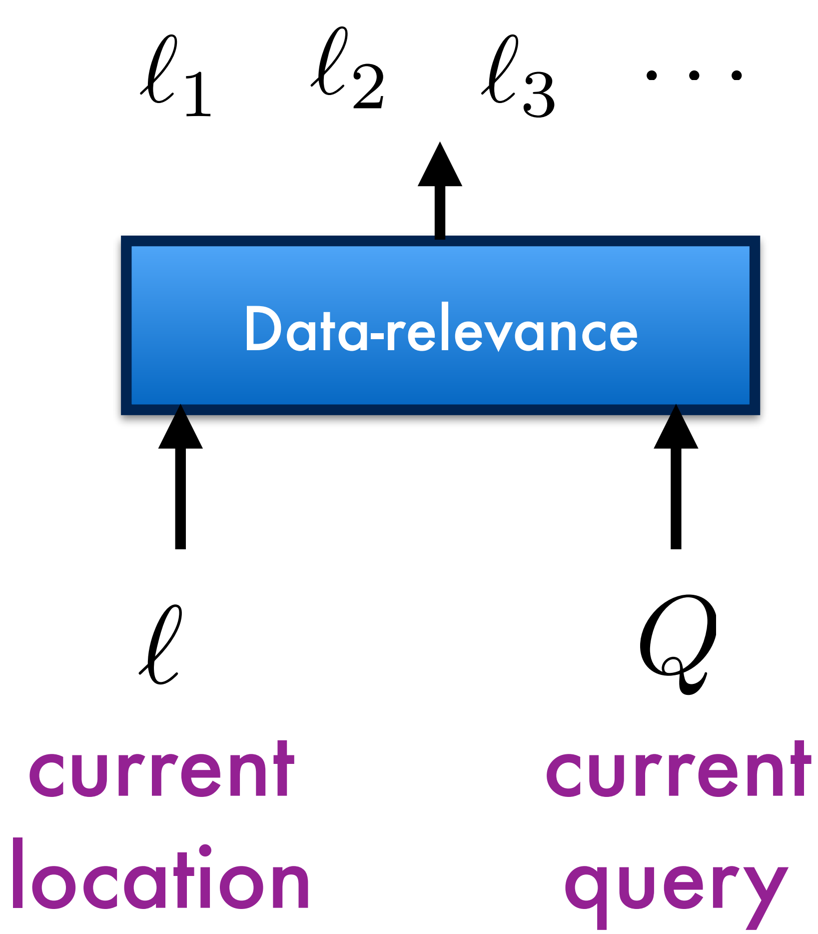
Computed using pre-pass points-to analysis, types, field-based, …



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

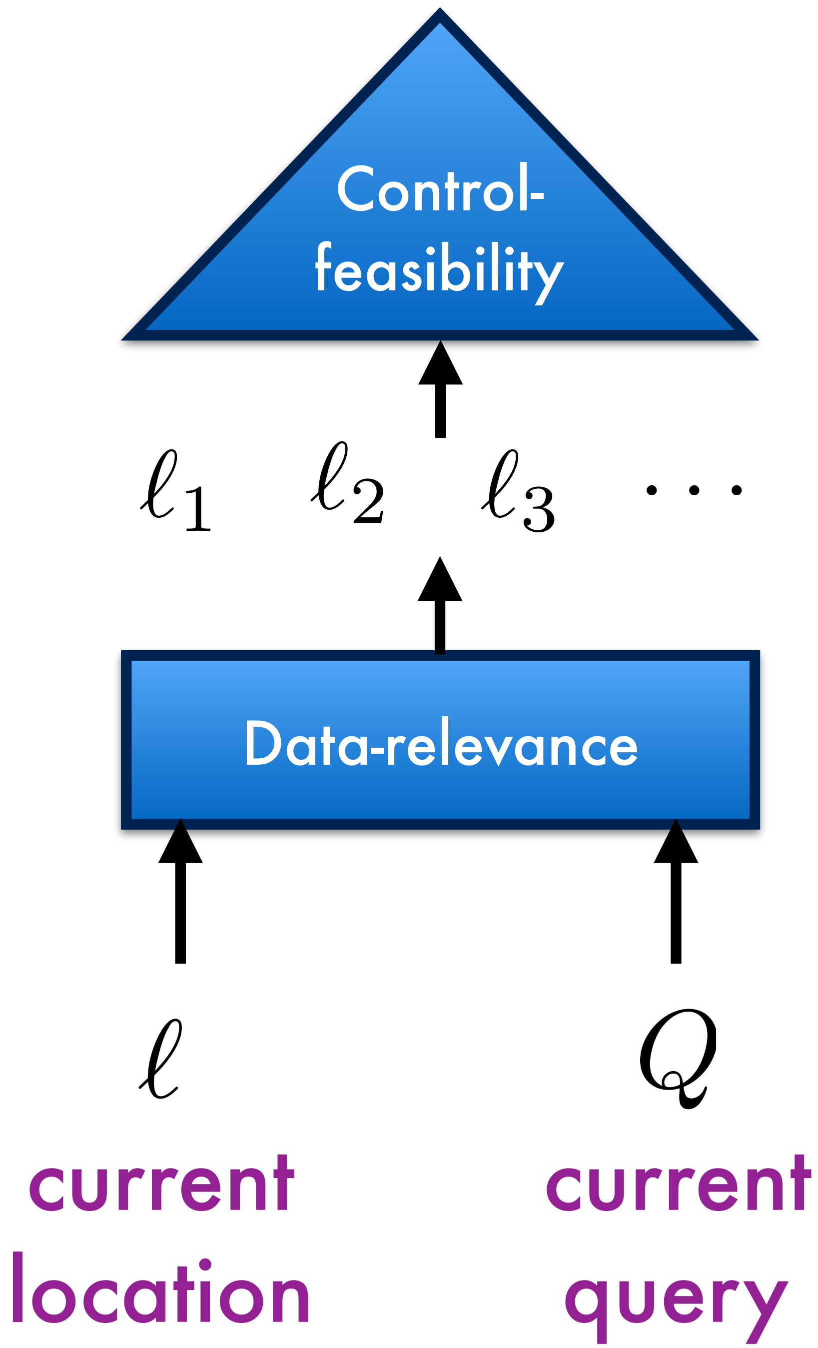
Control-feasibility recovers flow-sensitivity

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



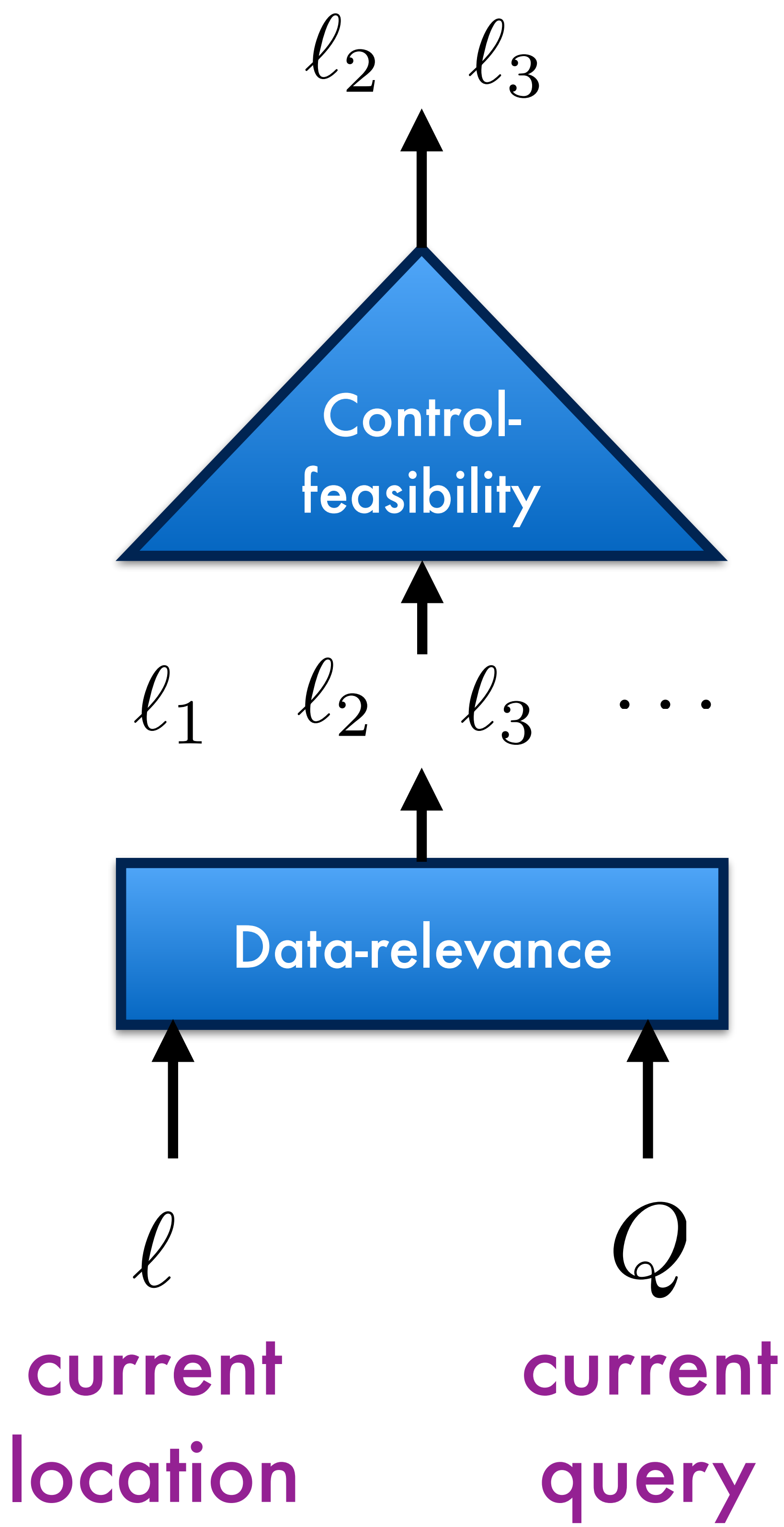
Control-feasibility recovers flow-sensitivity

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



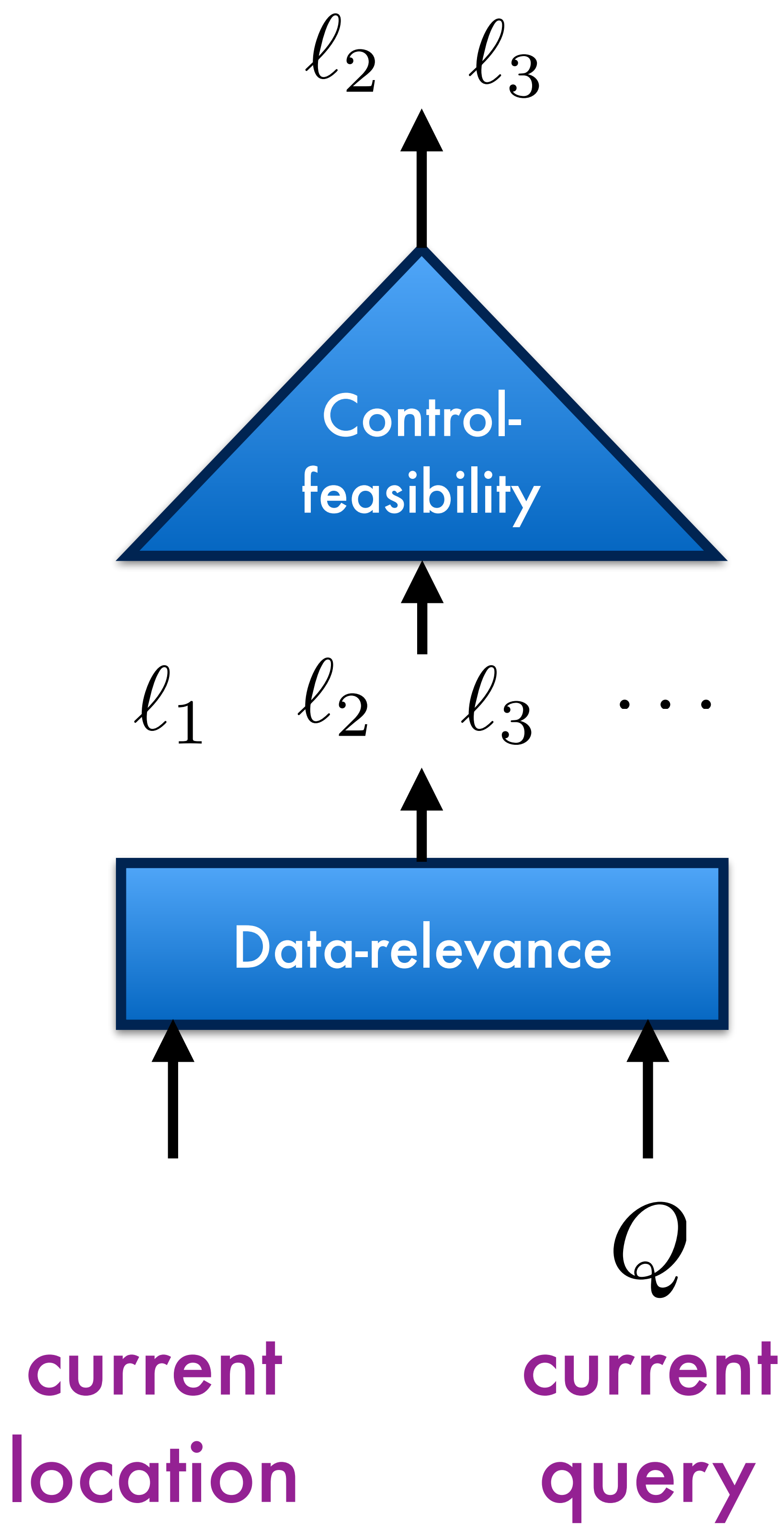
Control-feasibility recovers flow-sensitivity

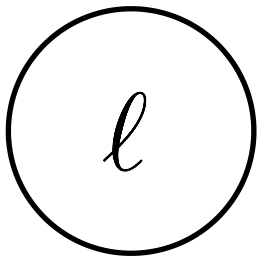
1 $\langle Q, \ell \rangle \rightsquigarrow T$



Control-feasibility recovers flow-sensitivity

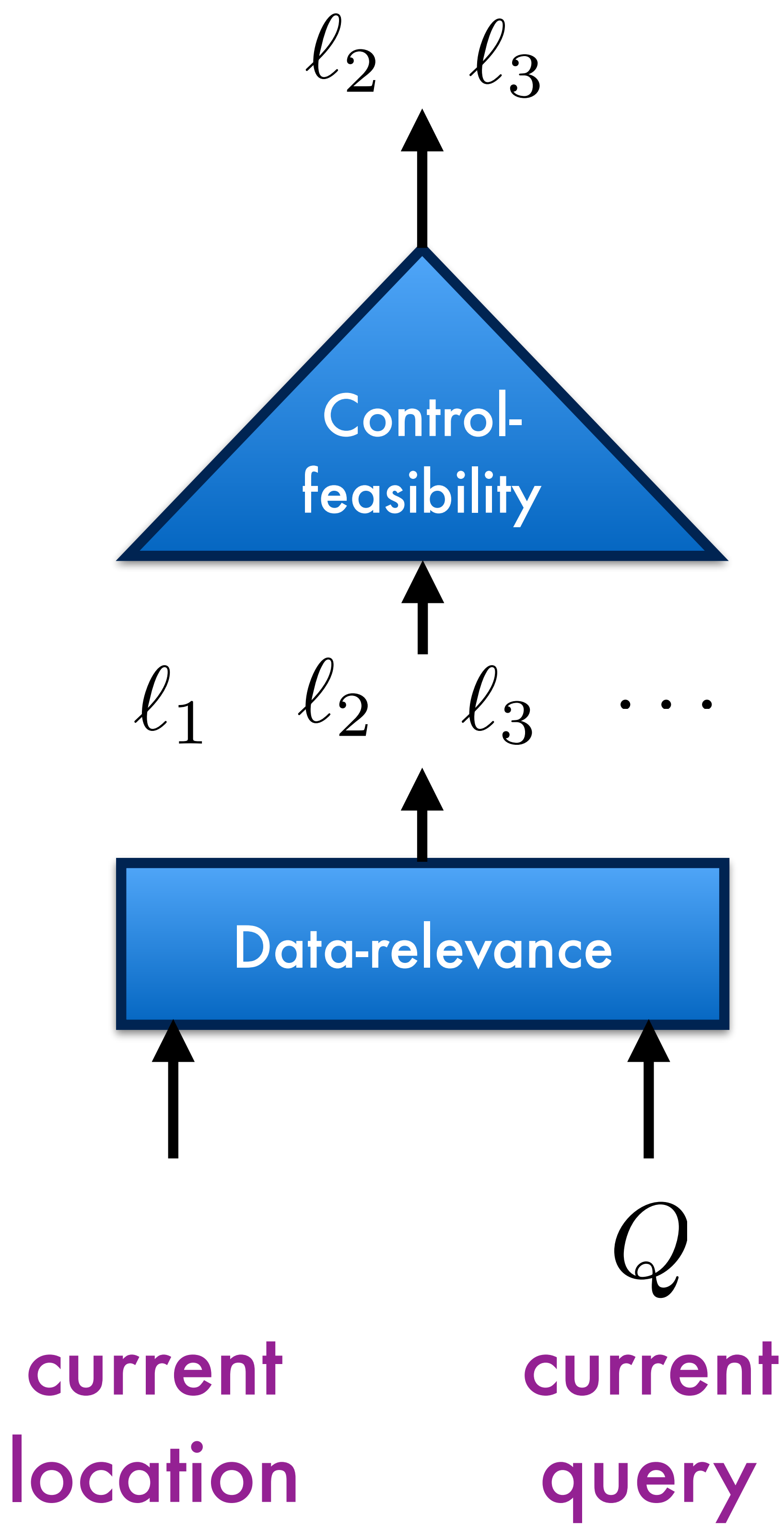
1 $\langle Q, \ell \rangle \rightsquigarrow T$



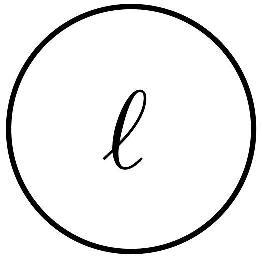


Control-feasibility recovers flow-sensitivity

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

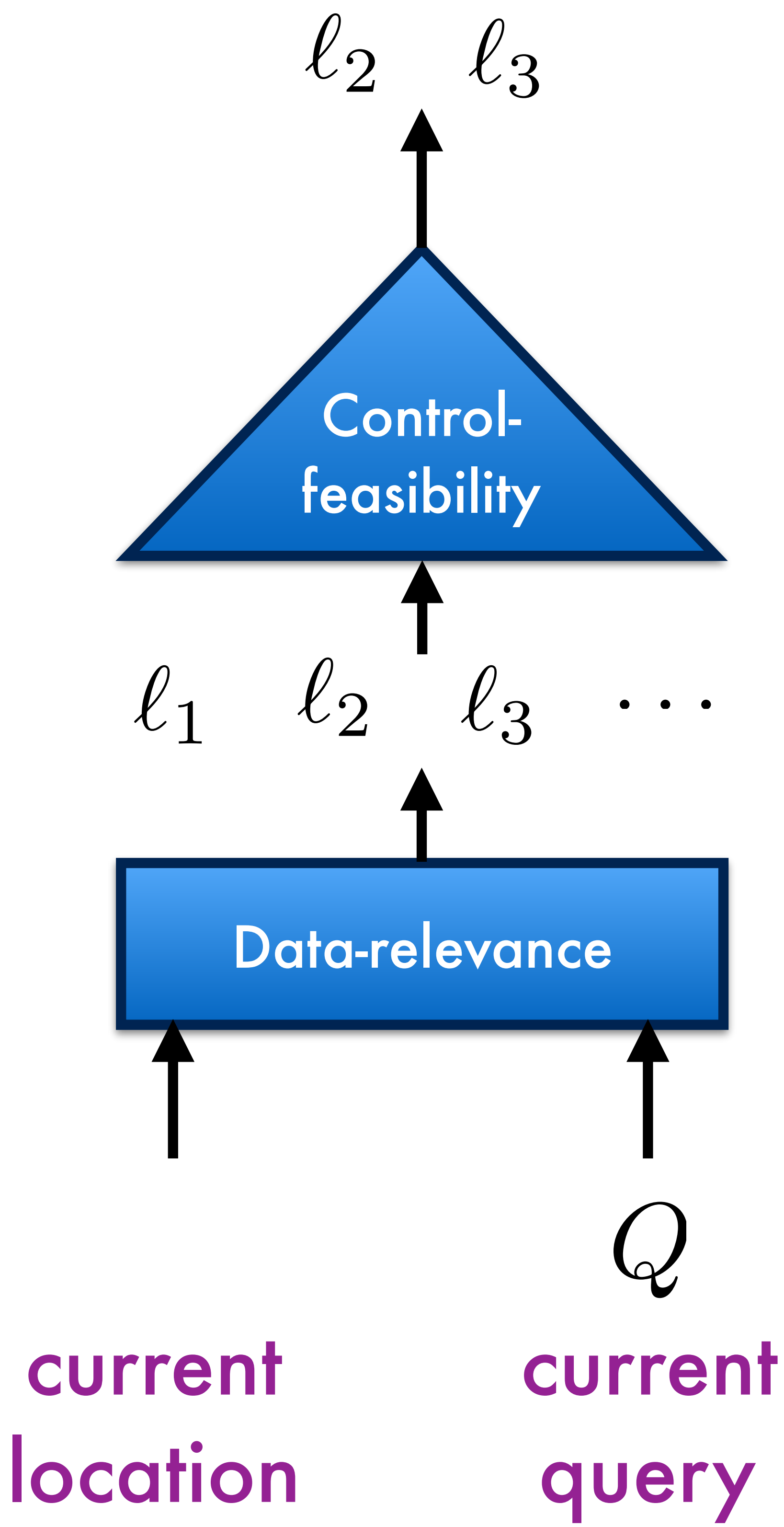


Filter the set of data-relevant locations using control flow and the current program point

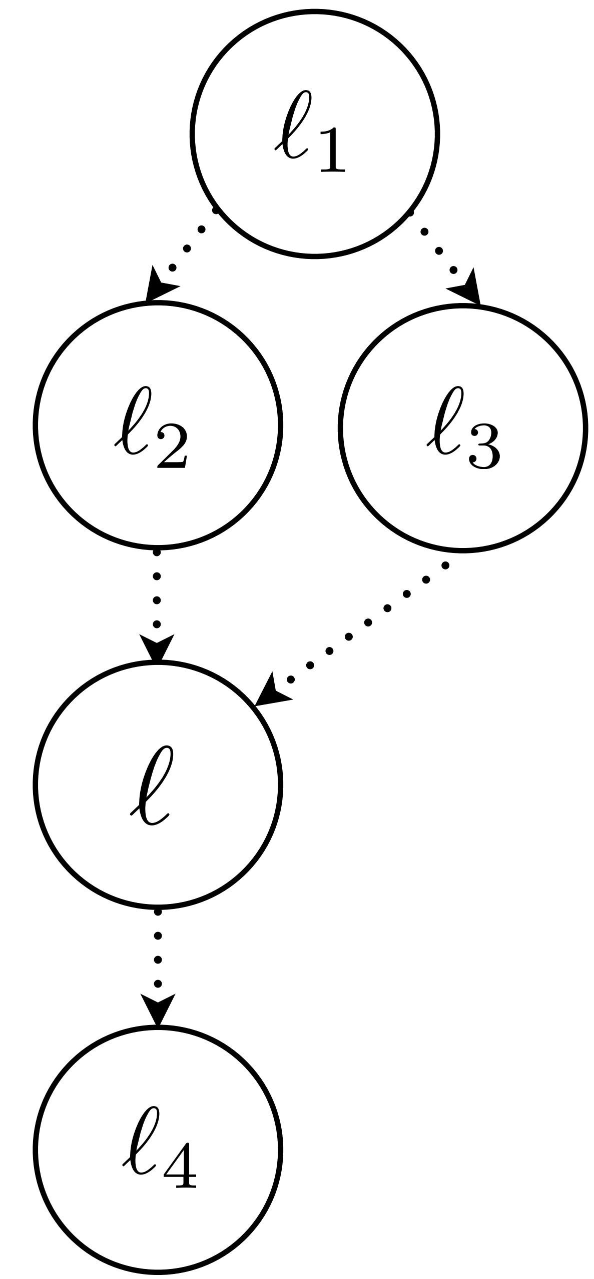


Control-feasibility recovers flow-sensitivity

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

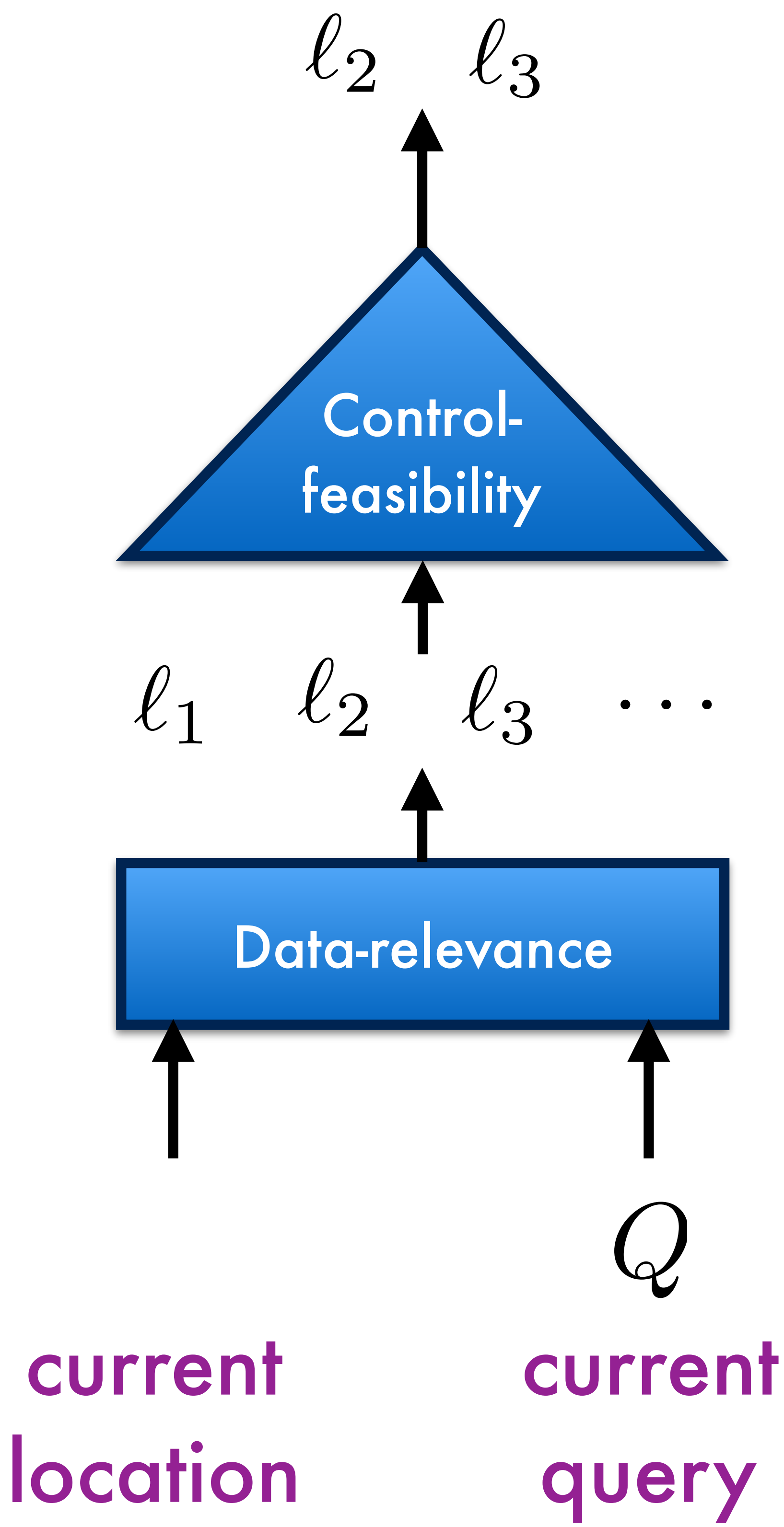


Filter the set of data-relevant locations using control flow and the current program point



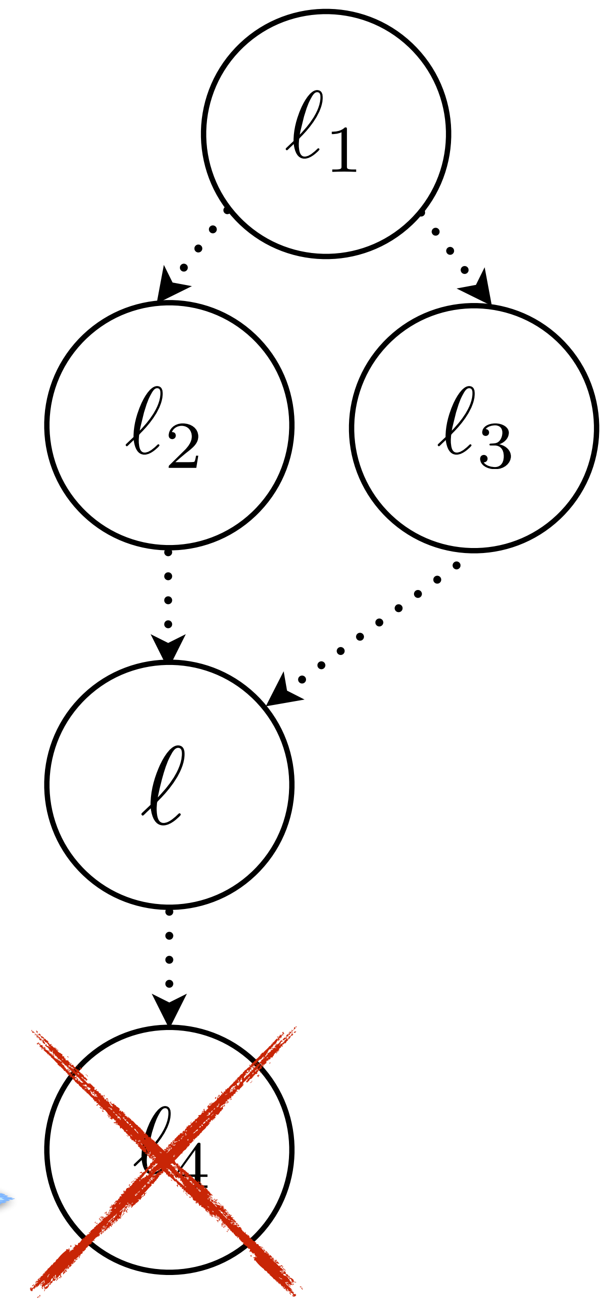
Control-feasibility recovers flow-sensitivity

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



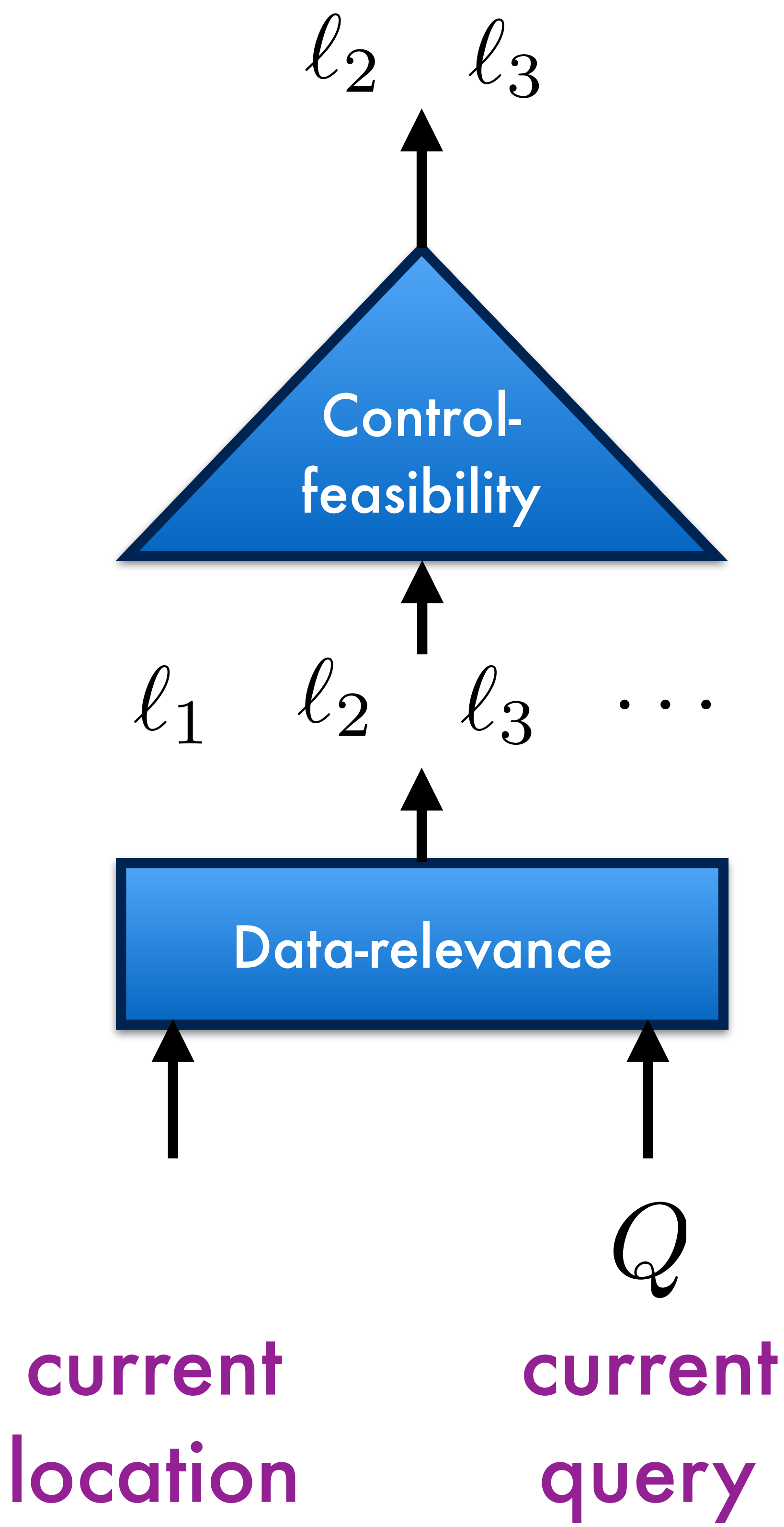
Filter the set of data-relevant locations using control flow and the current program point

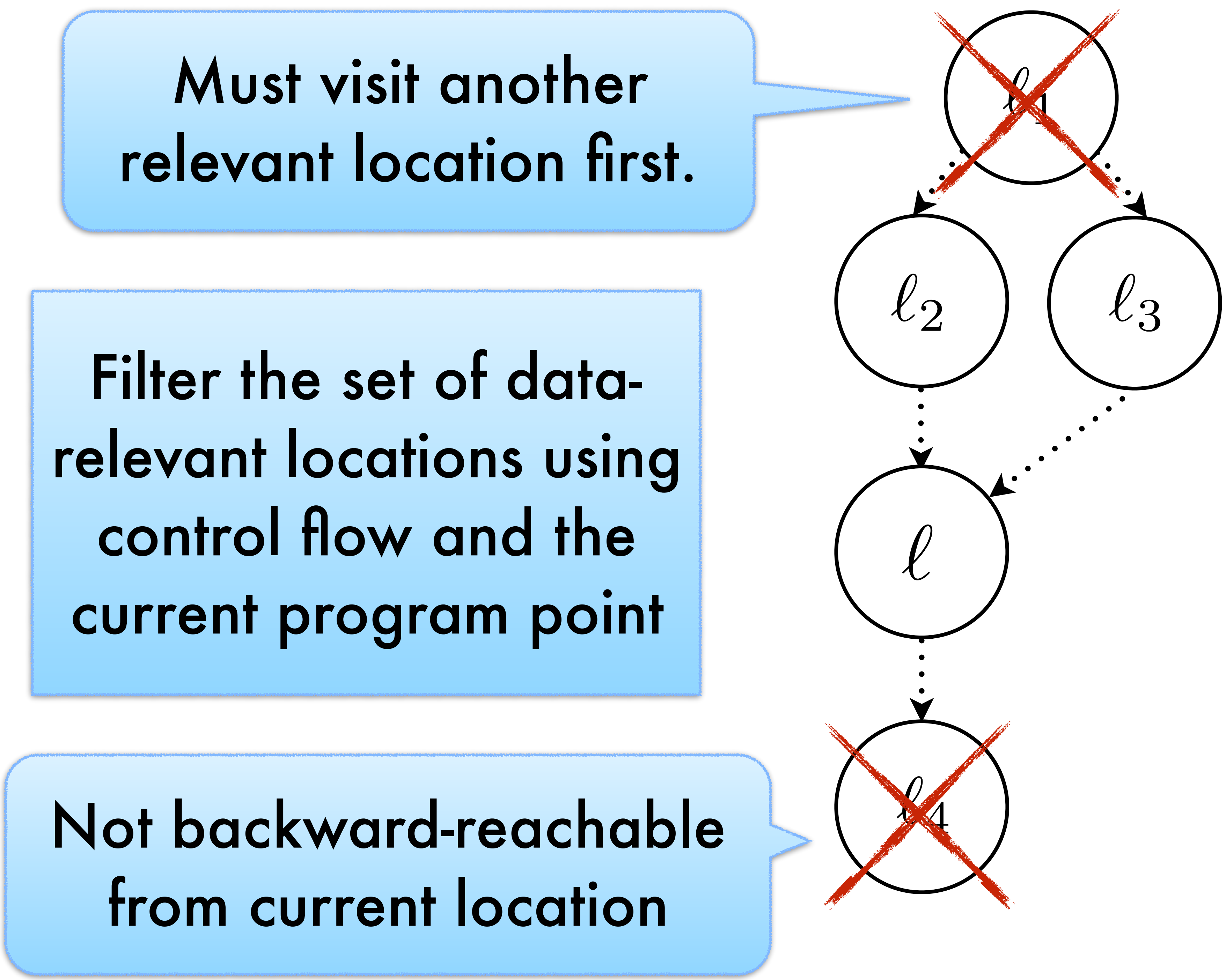
Not backward-reachable from current location



Control-feasibility recovers flow-sensitivity

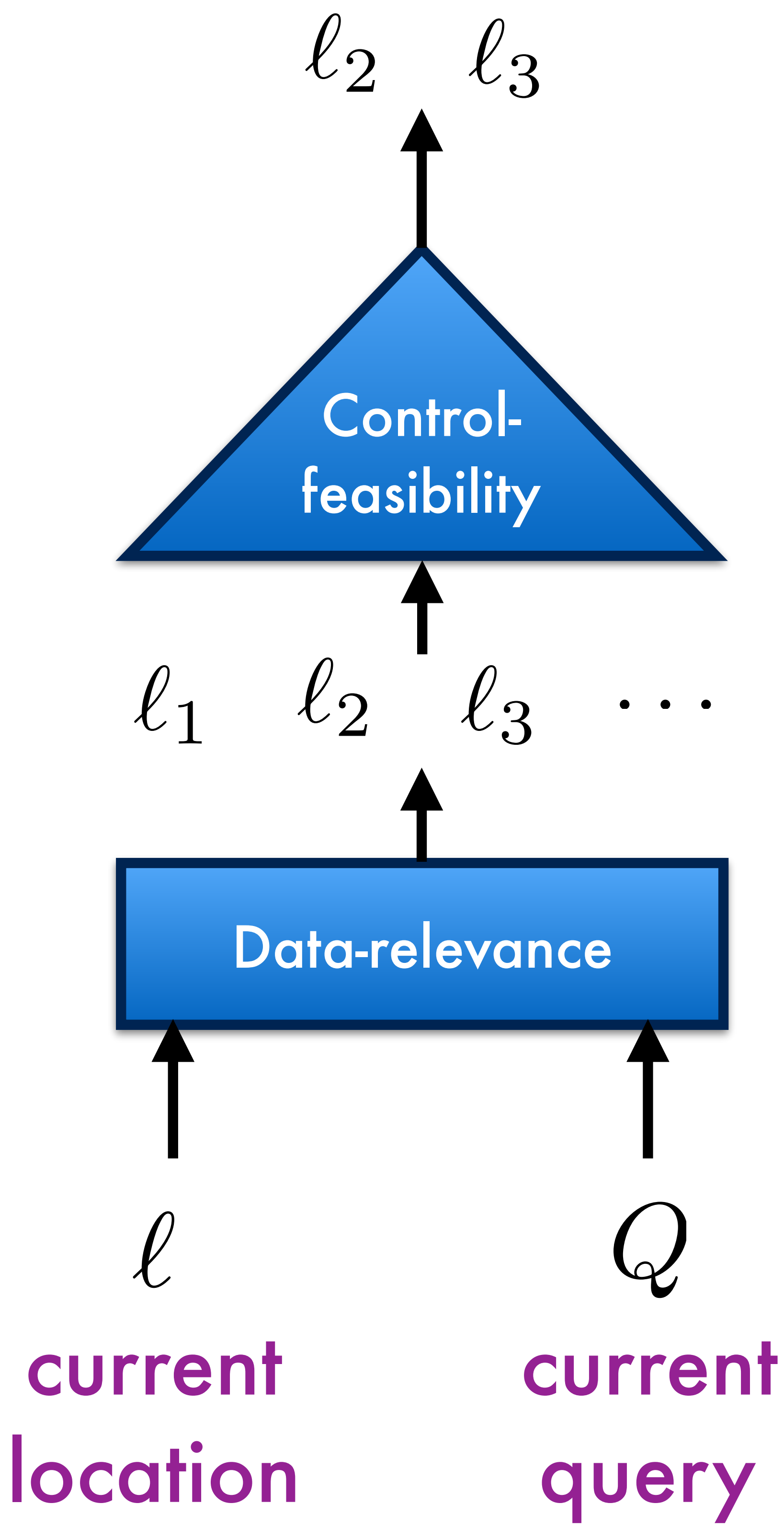
$\langle Q, \ell \rangle \rightsquigarrow T$





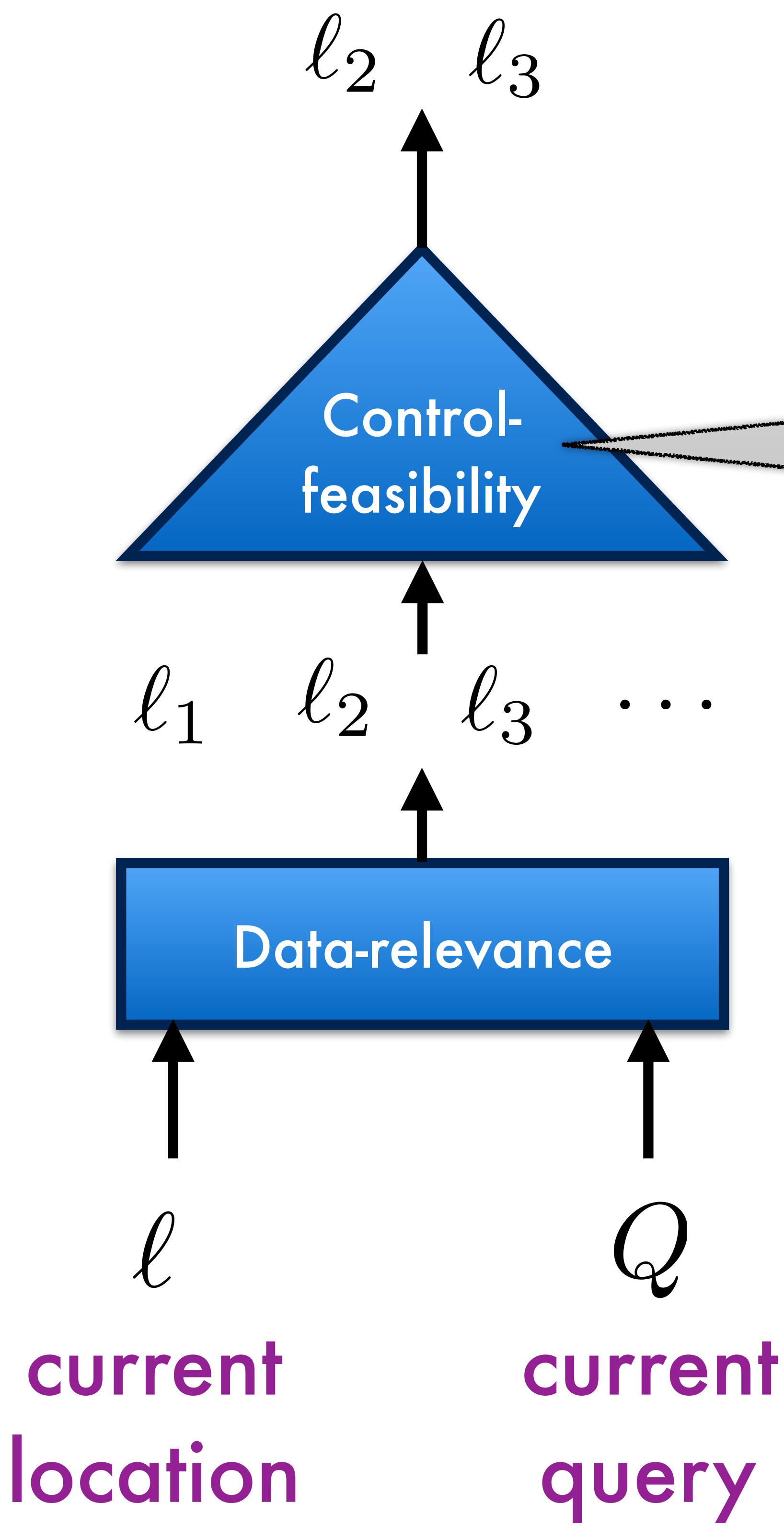
Insensitive versus sensitive

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



Insensitive versus sensitive

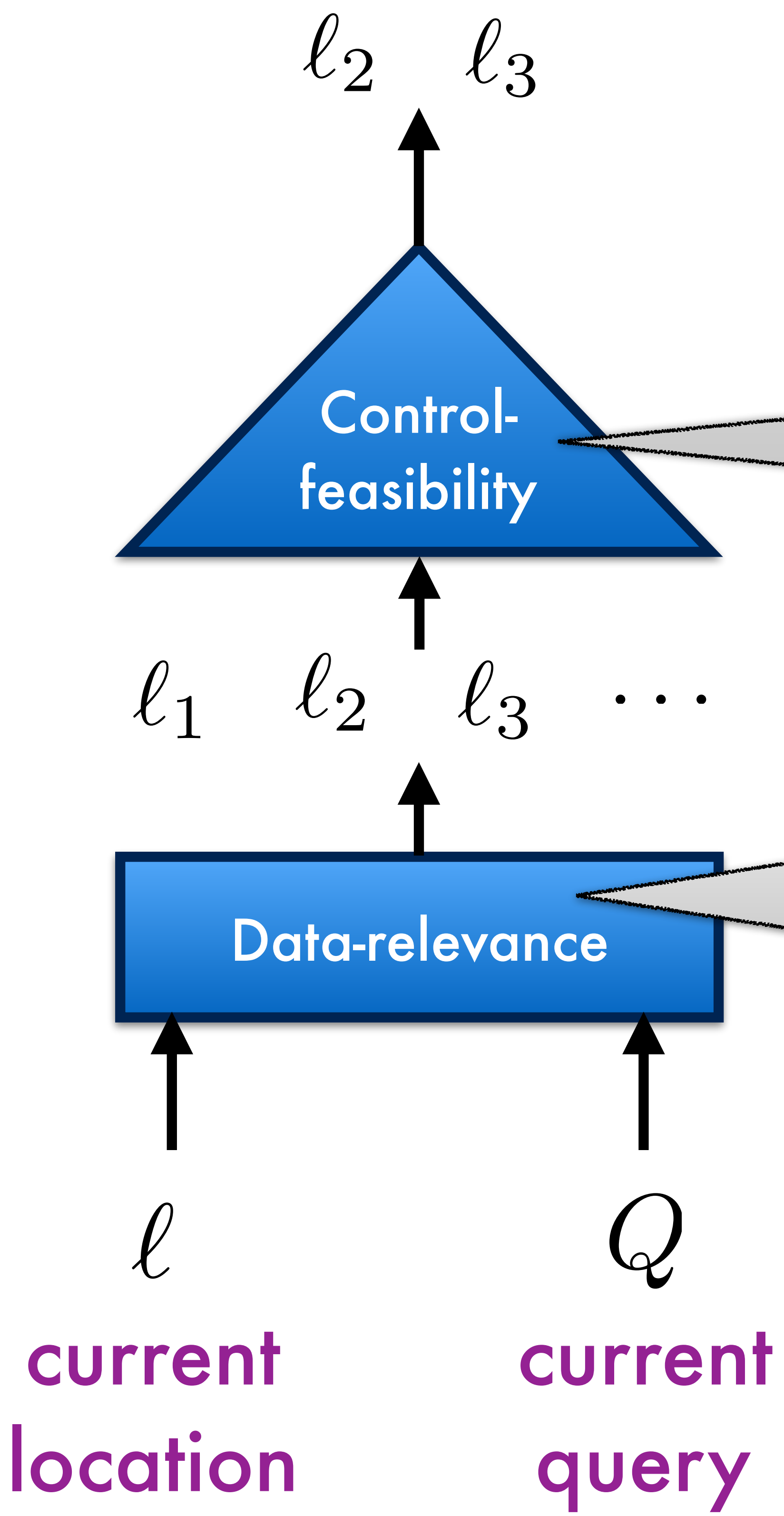
1 $\langle Q, \ell \rangle \rightsquigarrow T$



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

Insensitive versus sensitive

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

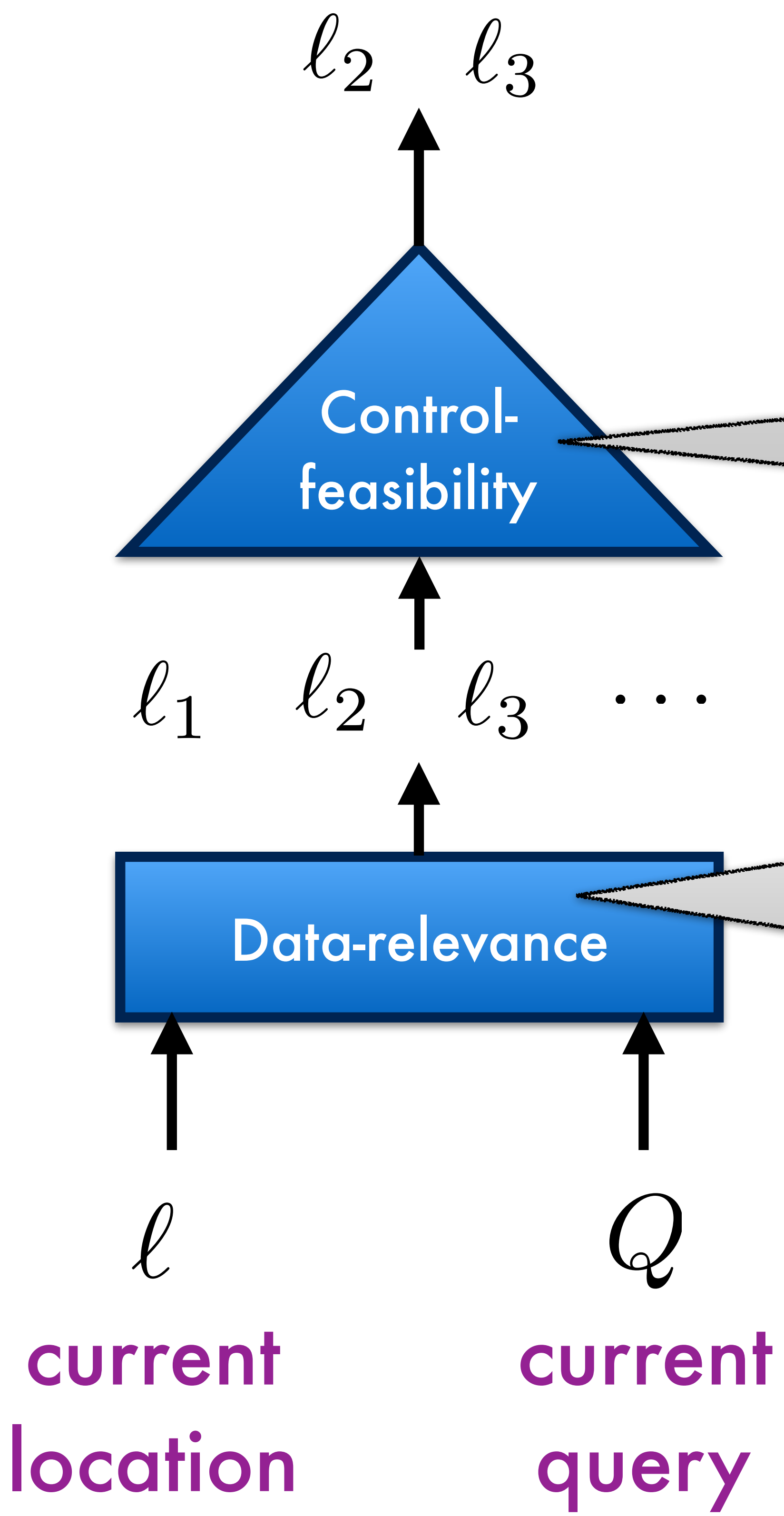


Different choices yield classical variants of flow, path, and context sensitivity/insensitivity

Make sparse

Insensitive versus sensitive

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

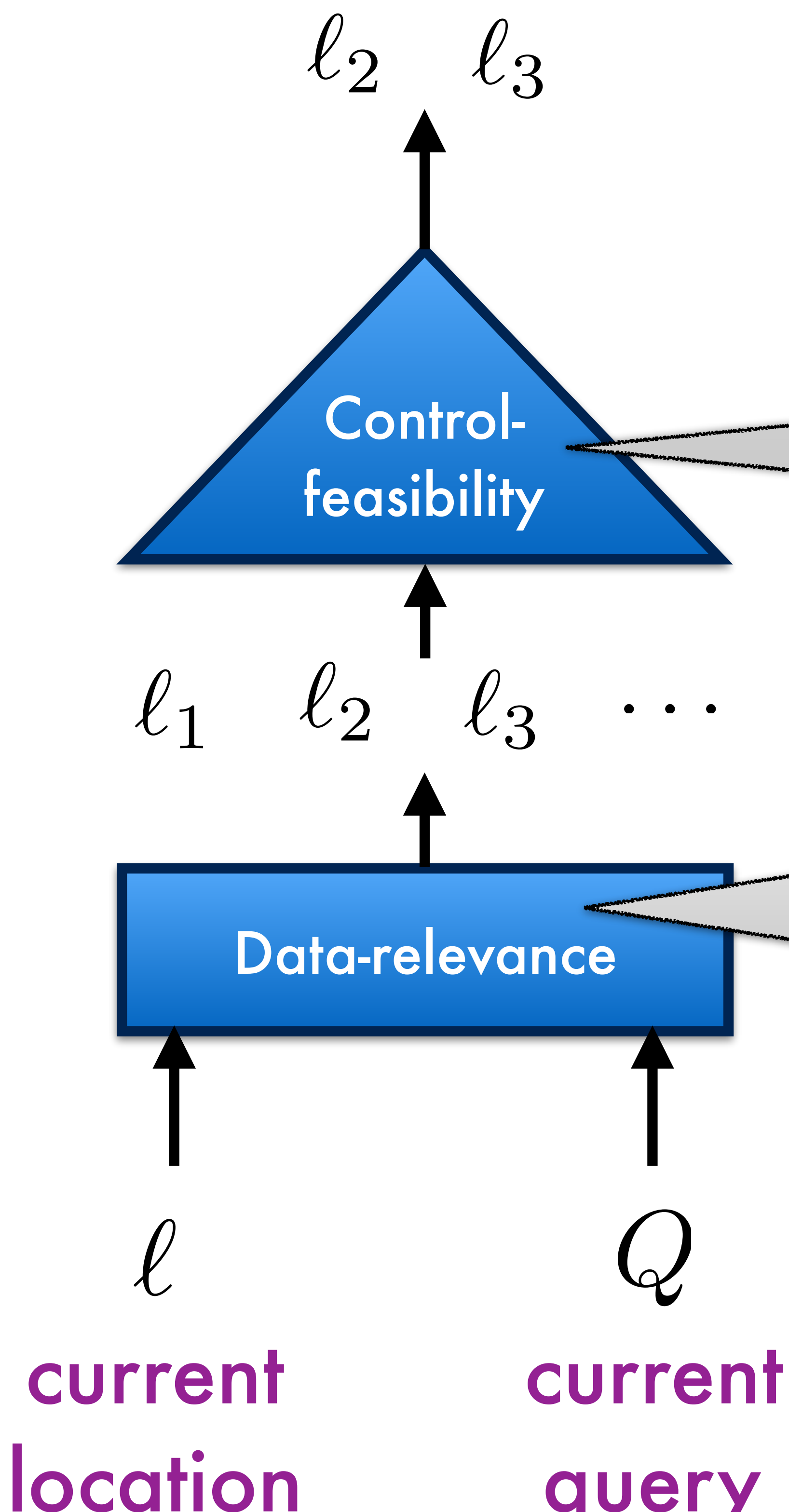


Be sparse by using data relevance with desired control-flow abstraction

Make sparse

Insensitive versus sensitive

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

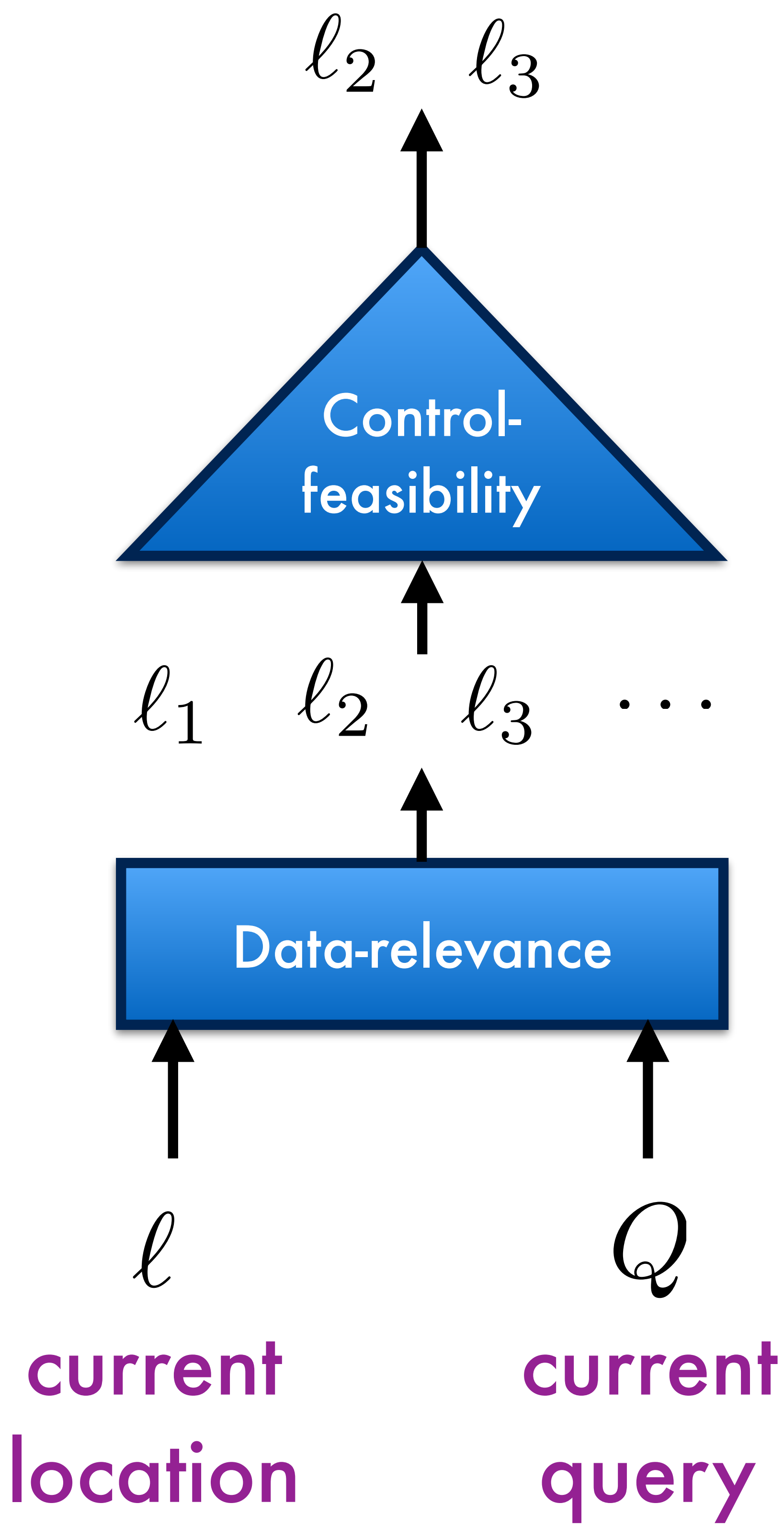


Be sparse by using data relevance with desired control-flow abstraction

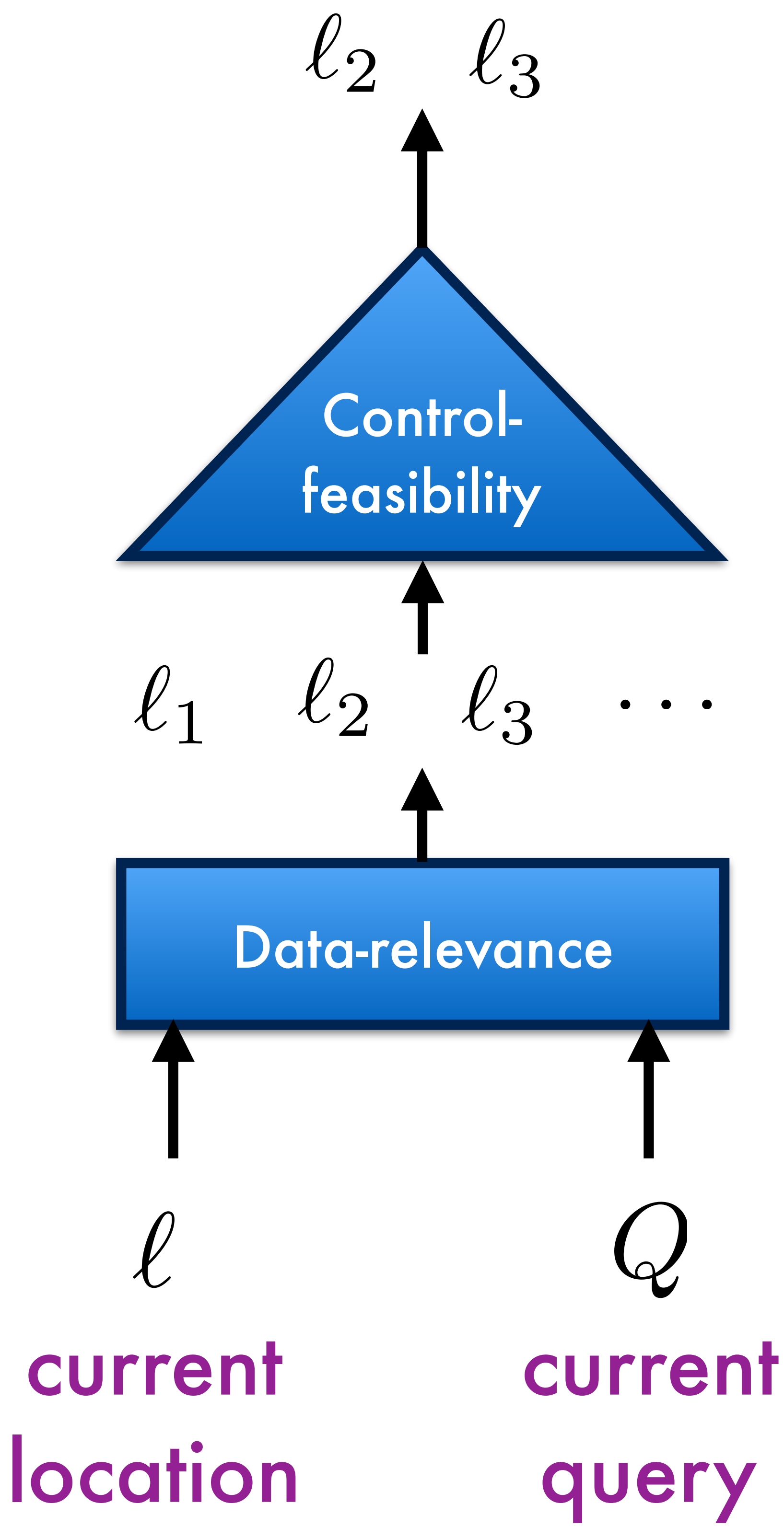
Be selective by varying the relevance relation at each analysis step

Soundness

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

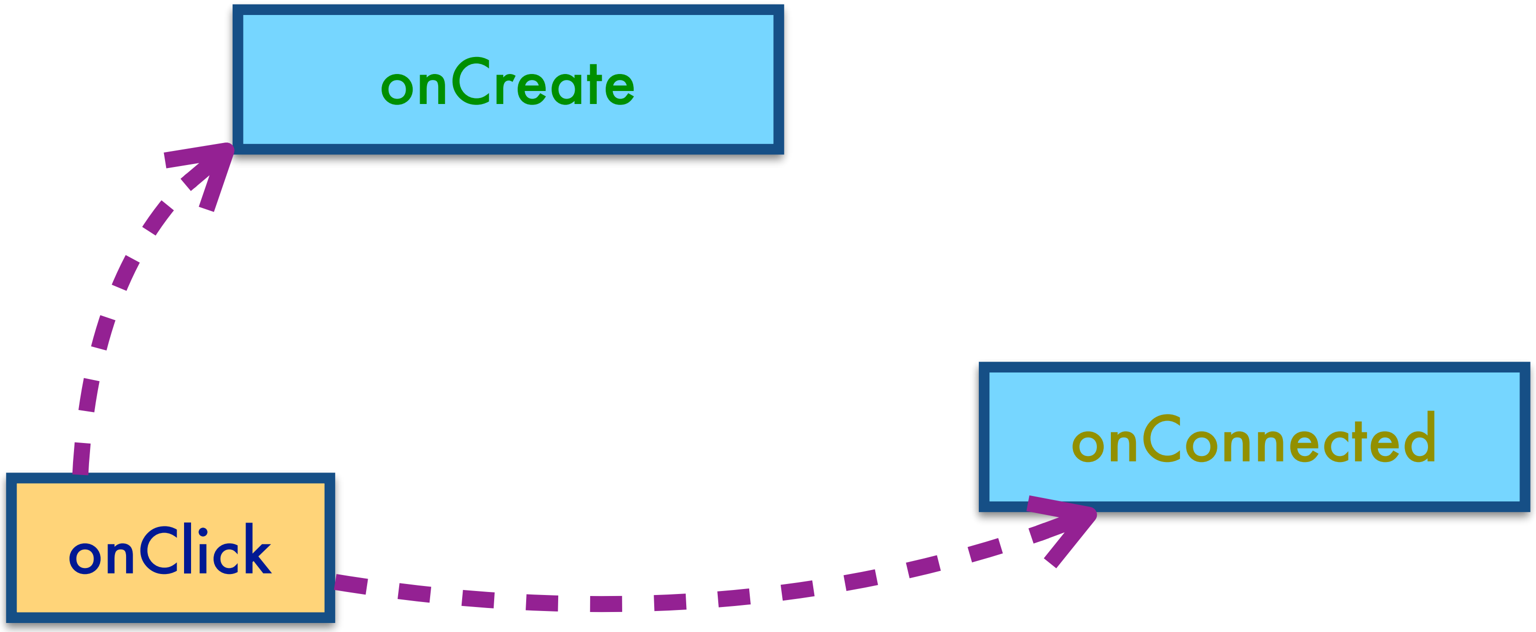


Soundness



Theorem:
If data-relevance and control-feasibility are sound, then no behavior relevant to refuting Q can be missed (i.e., "jumping is sound")

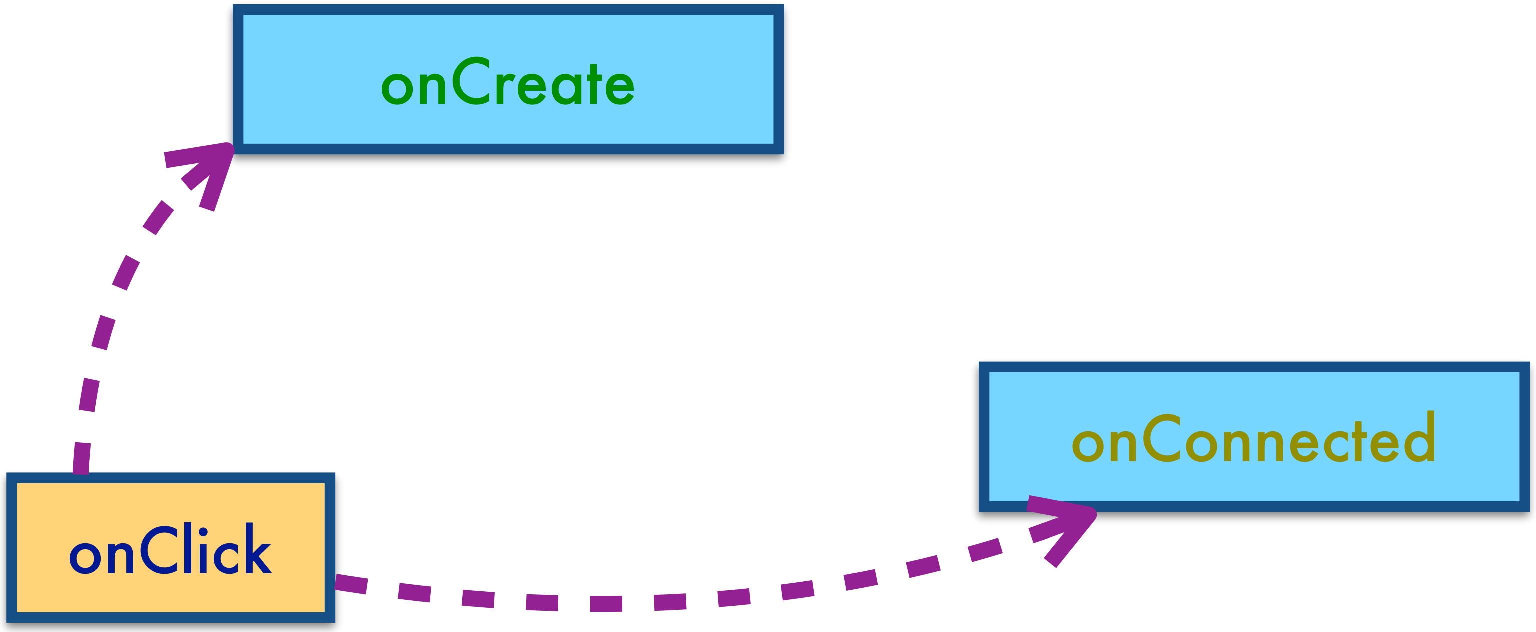
Contributions: Hopper is an analysis that jumps



1 $\langle Q, \ell \rangle \rightsquigarrow T$ Framework for sound jumping analyses

Applied to Android lifecycles

Contributions: Hopper is an analysis that jumps

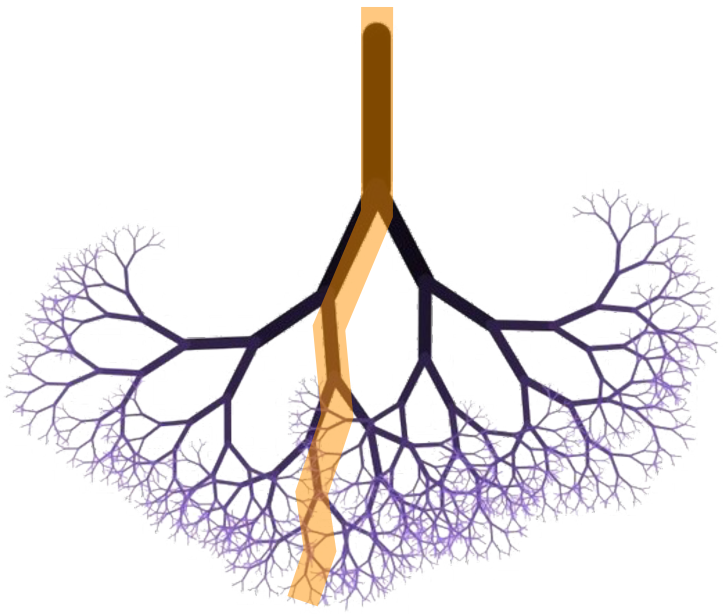


1 $\langle Q, \ell \rangle \rightsquigarrow T$ Framework for sound jumping analyses

2  Applied to Android lifecycles

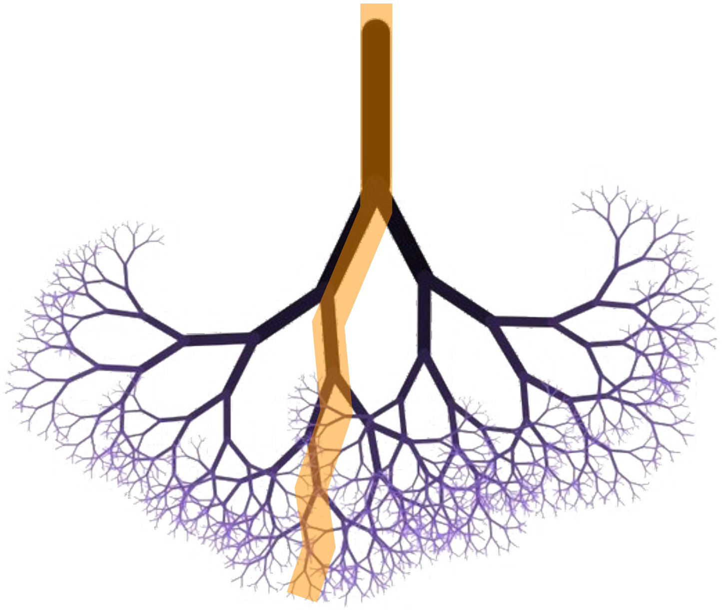
An effective jumping policy for inter-event Android analysis

An effective jumping policy for inter-event Android analysis



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

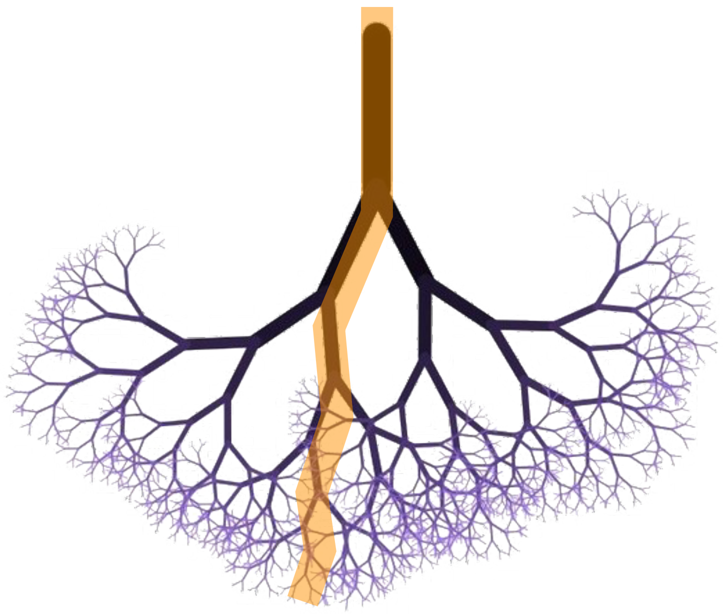
An effective jumping policy for inter-event Android analysis



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

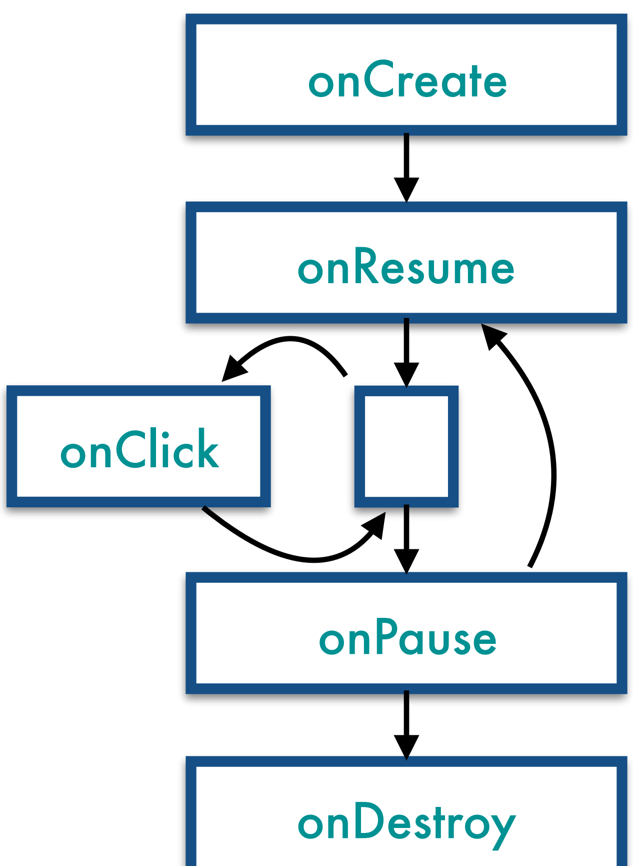
still feasible to be as precise as possible within callbacks

An effective jumping policy for inter-event Android analysis



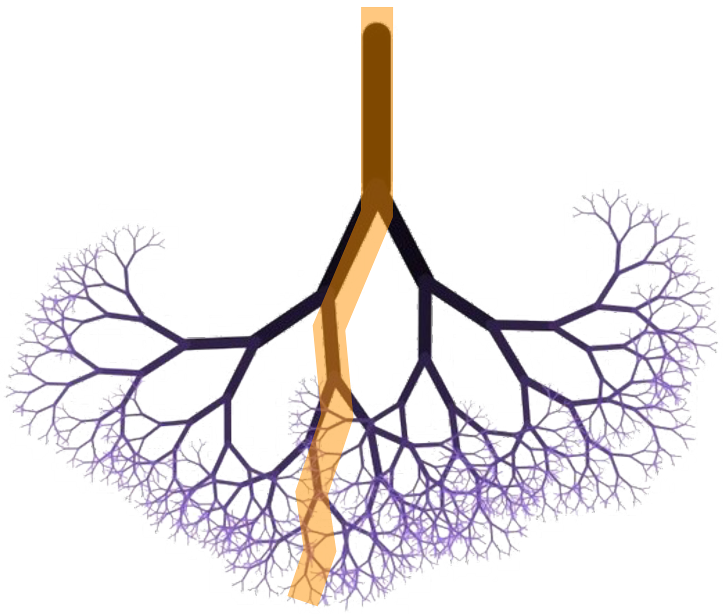
Within an event-callback (intra-event), follow predecessor transitions

still feasible to be as precise as possible within callbacks



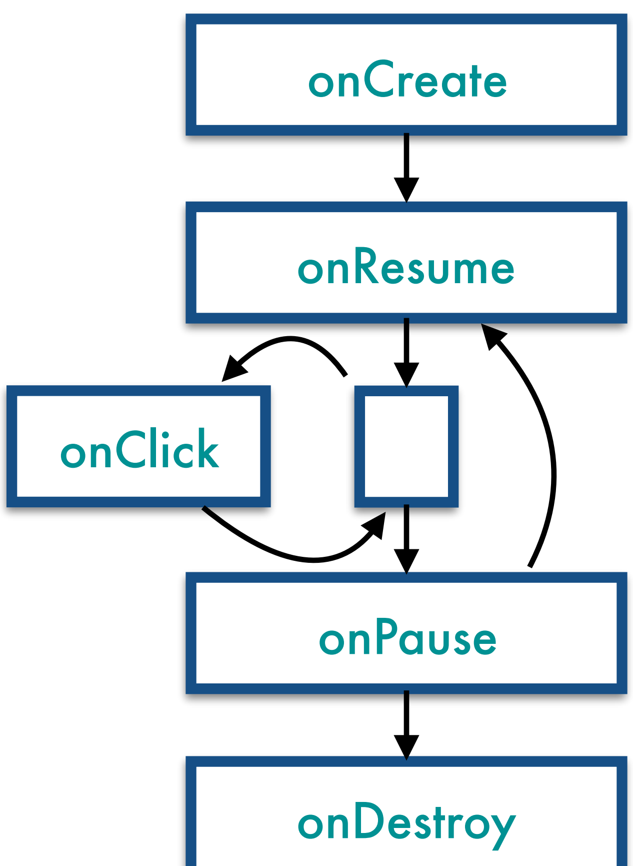
Between event-callbacks (inter-event), jump using lifecycle graphs for control-feasibility filtering

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

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

Compared 3 analyses

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

Is jumping effective?

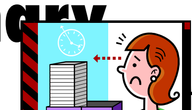
	KLOC
drupaleditor	3
npr	5
duckduckgo	11
lastfm	13
github	19
seriesguide	32
connectbot	33
textsecure	38
k-9	55
wordpress	57
Summary	**266**

Is jumping effective?

	KLOC	Deref
drupaleditor	3	928
npr	5	829
duckduckgo	11	1969
lastfm	13	4840
github	19	3603
seriesguide	32	8184
connectbot	33	2190
textsecure	38	5921
k-9	55	19032
wordpress	57	15066
Summary	**266**	**62562**

Is jumping effective?

	KLOC	Deref
drupaleditor	3	928
npr	5	829
duckduckgo	11	1969
lastfm	13	4840
github	19	3603
seriesguide	32	8184
connectbot	33	2190
textsecure	38	5921
k-9	55	19032
wordpress	57	15066
Summary	**266**	**62562**



Huge number of dereferences

Is jumping effective?

unproven derefs

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

Is jumping effect

type-based

unproven derefs

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

Is jumping effect

type-based

no jumping

unproven derefs

	KLOC	Deref	Nit	Thr
drupaleditor	3	928	679	179
npr	5	829	617	181
duckduckgo	11	1969	1341	518
lastfm	13	4840	3528	954
github	19	3603	2520	601
seriesguide	32	8184	5438	986
connectbot	33	2190	1562	316
textsecure	38	5921	3643	698
k-9	55	19032	11968	3104
wordpress	57	15066	9775	2431
Summary	**266**	**62562**	**41071**	**9968**

Is jumping effect

type-based

no jumping

unproven derefs

	KLOC	Deref	Nit	Thr
drupaleditor	3	928	679	179
npr	5	829	617	181
duckduckgo	11	1969	1341	518
lastfm	13	4840	3528	954
github	19	3603	2520	601
seriesguide	32	8184	5438	986
connectbot	33	2190	1562	316
textsecure	38	5921	3643	698
k-9	55	19032	11968	3104
wordpress	57	15066	9775	2431
Summary	**266**	**62562**	**41071**	**9968**

Find callback interleavings

Is jumping effect

type-based

no jumping

jumping

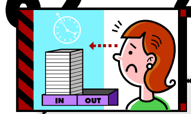
	KLOC	Deref	Nit	Thr	Hop
			unproven derefs		
drupaleditor	3	928	679	179	72
npr	5	829	617	181	51
duckduckgo	11	1969	1341	518	143
lastfm	13	4840	3528	954	477
github	19	3603	2520	601	290
seriesguide	32	8184	5438	986	625
connectbot	33	2190	1562	316	74
textsecure	38	5921	3643	698	330
k-9	55	19032	11968	3104	1988
wordpress	57	15066	9775	2431	1362
Summary	**266**	**62562**	**41071**	**9968**	**5412**

Find callback interleavings

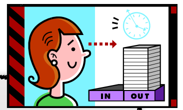
Is jumping effect

type-based | no jumping | jumping

	KLOC	Deref	unproven derefs Nit	Thr	Hop
drupaleditor	3	928	679	179	72
npr	5	829	617	181	51
duckduckgo	11	1969	1341	518	143
lastfm	13	4840	3528	954	477
github	19	3603	2520	601	290
seriesguide	32	8184	5438	986	625
connectbot	33	2190	1562	316	74
textsecure	38	5921	3643	698	330
k-9	55	19032	11968	3104	1988
wordpress	57	15066	9775	2431	1362
Summary	**266**	**62562**	**41071**	**9968**	**5412**



Find callback interleavings



Are given callback interleavings feasible?

Is jumping effect

type-based

no jumping

jumping

			unproven derefs			jumping effectiveness
	KLOC	Deref	Nit	Thr	Hop	% Improve
drupaleditor	3	928	679	179	72	60
npr	5	829	617	181	51	72
duckduckgo	11	1969	1341	518	143	72
lastfm	13	4840	3528	954	477	50
github	19	3603	2520	601	290	52
seriesguide	32	8184	5438	986	625	37
connectbot	33	2190	1562	316	74	77
textsecure	38	5921	3643	698	330	53
k-9	55	19032	11968	3104	1988	36
wordpress	57	15066	9775	2431	1362	44
Summary	**266**	**62562**	**41071**	**9968**	**5412**	**54**

Is jumping effect

type-based

no jumping

jumping

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

Is jumping effect

type-based | no jumping | jumping

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

Is jumping effect

			type-based	no jumping	jumping		
			unproven derefs			jumping effectiveness	
	KLOC	Deref	Nit	Thr	Hop	% Improve	% Proven
drupaleditor	3	928	679	179	72	60	92
npr	5	829	617	181	51	72	94
duckduckgo	11	1969	1341	518	143	72	93
lastfm	13	4840	3528	954	477	50	90
github	19	3603	2520	601	290	52	92
seriesguide	32	8184	5438	986	625	37	92
connectbot	33	2190	1562	316	74	77	97
textsecure	38	5921	3643	698	330	53	94
k-9	55	19032	11968	3104	1988	36	90
wordpress	57	15066	9775	2431	1362	44	91
Summary	**266**	**62562**	**41071**	**9968**	**5412**	**54**	**92**

Is jumping effect

type-based

no jumping

jumping

			unproven derefs			jumping effectiveness	
	KLOC	Deref	Nit	Thr	Hop	% Improve	% Proven
drupaleditor	3	928	679	179	72	60	92
npr	5	829	617	181	51	72	94
duckduckgo	11	1969	1341	518	143	72	93
lastfm	13	4840	3528	954	477	50	90
github	[illegible]	[illegible]	[illegible]	[illegible]	290	52	92
seriesguide	[illegible]	[illegible]	[illegible]	[illegible]	625	37	92
connectbot	[illegible]	[illegible]	[illegible]	[illegible]	74	77	97
textsecure	[illegible]	[illegible]	[illegible]	[illegible]	330	53	94
k-9	[illegible]	[illegible]	[illegible]	[illegible]	1988	36	90
wordpress	[illegible]	[illegible]	[illegible]	[illegible]	1362	44	91
Summary	**266**	**62562**	**41071**	**9968**	**5412**	**54**	**92**

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

Is jumping effect

type-based

no jumping

jumping

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

Is jumping effect

type-based

no jumping

jumping

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

< 1 day

~1 sec per deref

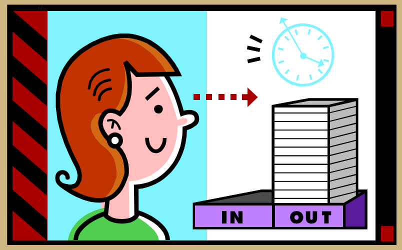
Is jumping effect

	type-based	no jumping	jumping			

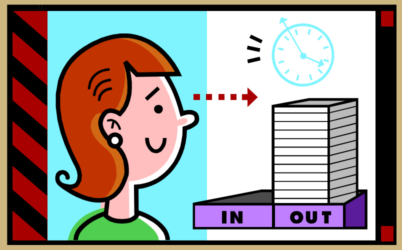
		unproven derefs			jumping effectiveness		
	KLOC	Deref	Nit	Thr	Hop	% Improve	% Proven	Time (min)
drupaleditor	3	928	679	179	72	60	92	12.0
npr	5	829	617	181	51	72	94	11.5
duckduckgo	11	1969	1341	518	143	72	93	98.7
lastfm	13	4840	3528	954	477	50	90	29.2
github	19	3603	2520	601	290	52	92	62.5
seriesguide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

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

Triaging alarms to find bugs

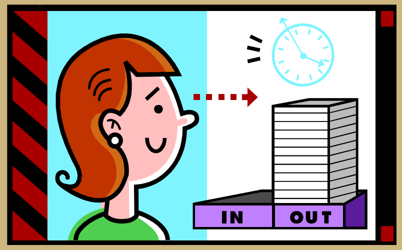


Triaging alarms to find bugs



Triaged 200 alarms (from Hopper), 189 false

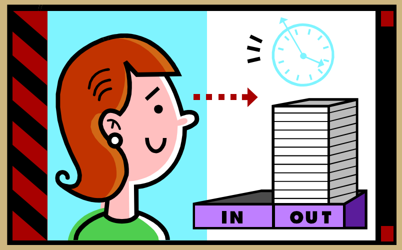
Triaging alarms to find bugs



Triaged 200 alarms (from Hopper), 189 false

- Reasons: insufficient Android modeling, imprecise container and string domains

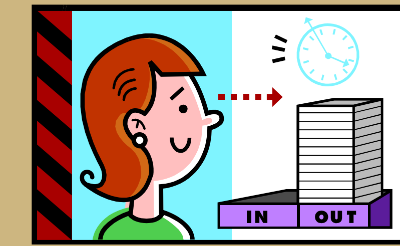
Triaging alarms to find bugs



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

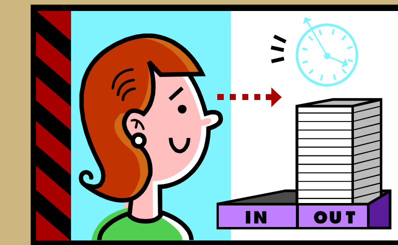
- Reasons: insufficient Android modeling, imprecise container and string domains
- Only 17 false alarms due to timeouts

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



Tria… se

Re… e

containe… and

Only 17 false…

Fixing possible NPE in OverviewActivity.launchSearch #449

Merged

Fixing possible NPEs when CheckInDialogFragment.newInstance returns null #450

Merged

Fix possible NPE in HostListActivity.onCreateContextMenu #60

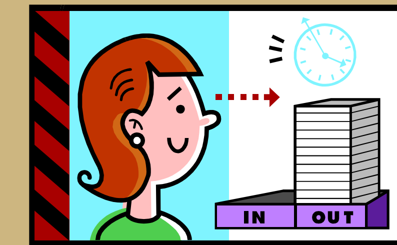
Merged kruton merged 1 commit into `connectbot:master` from `sblackshear:service_npe_fix` on Mar 28

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



Tria … se

Re … e

containe … and

Only 17 false

Fixing possible NPE in OverviewActivity.launchSearch #449

Merged

Fixing possible NPEs when CheckInDialogFragment.newInstance returns null #450

Merged

Fix possible NPE in HostListActivity.onCreateContextMenu #60

Merged kruton merged 1 commit into `connectbot:master` from `sblackshear:service_npe_fix` on Mar 28

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

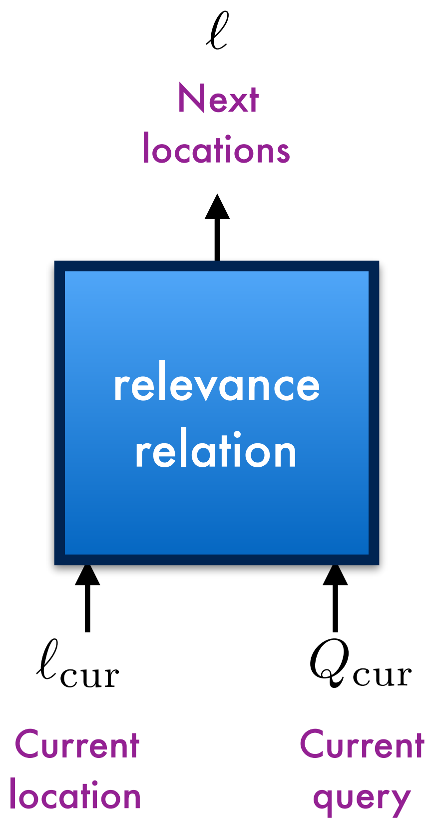
5 bugs due to bad ordering assumptions

10/11 patches accepted

one not accepted in a seemingly inactive project

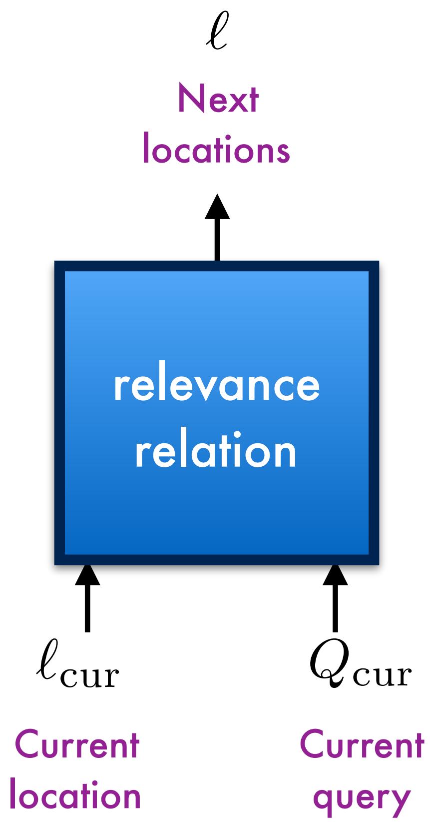
Summary: Hopper is an analysis that jumps

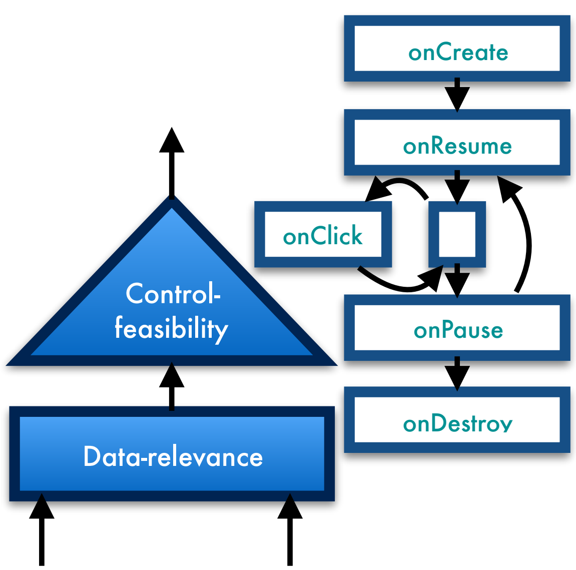
Summary: Hopper is an analysis that jumps



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

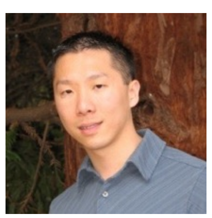
Prove and triage
applications (assuming protocol specifications)

Hopper

Mine artifacts for protocol specifications to subsequently "transfer" bug fixes

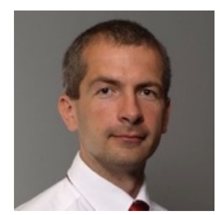
Fixr: Mining and Understanding Bug Fixes for Event-Driven Protocols

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



Bor-Yuh Evan Chang

Kenneth M. Anderson



Pavol Černý

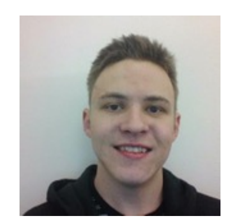
Sriram Sankaranarayanan

Tom Yeh

Edmund S.L. Lam

Sergio Mover

Shawn Meier

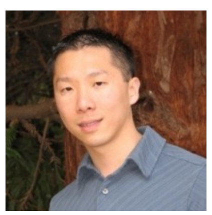


Rhys Braginton Pettee Olsen

Maxwell Russek

University of Colorado Boulder

Fixr: Mining and Understanding Bug Fixes for Event-Driven



Bor-Yuh Evan Chan

Edmund S.L. Lam

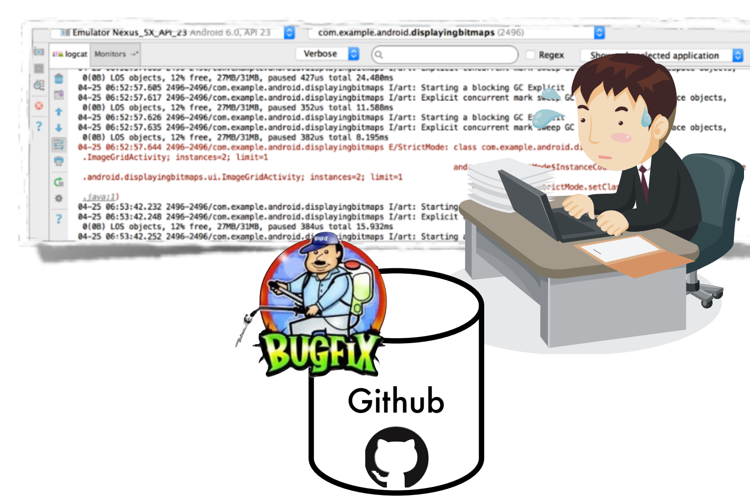
n

ssek

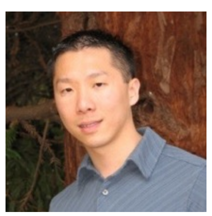
I am not alone

I don't know how that field became **null**.

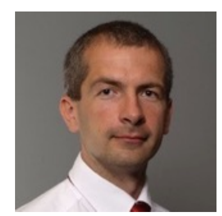
"Transfer" the bug fix with program analysis and synthesis



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


Bor-Yuh Evan Chang


Kenneth M. Anderson


Pavol Černý

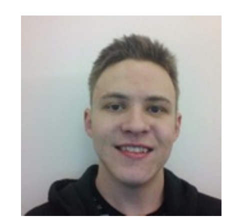

Sriram Sankaranarayanan


Tom Yeh


Edmund S.L. Lam


Sergio Mover

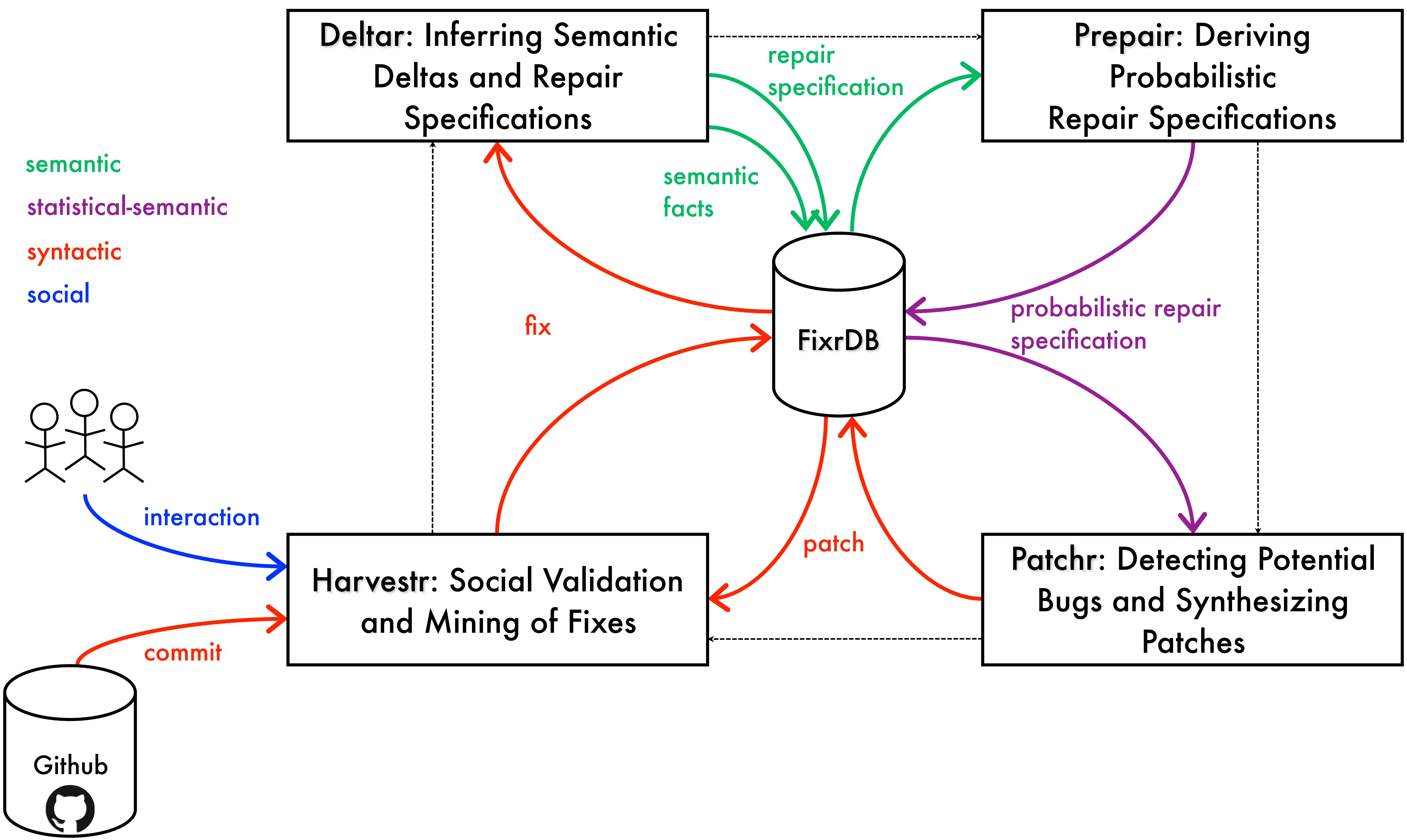

Shawn Meier


Rhys Braginton Pettee Olsen

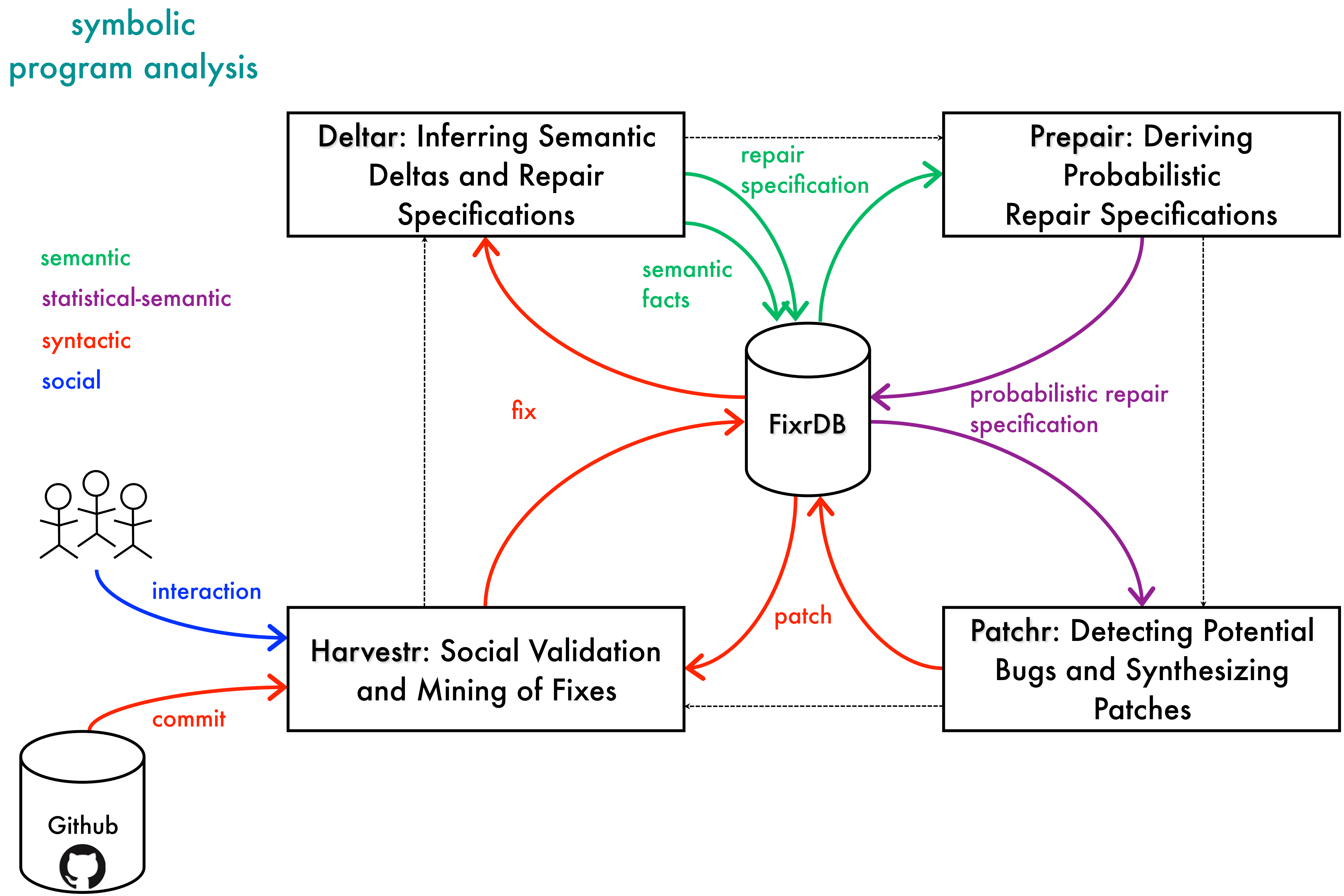

Maxwell Russek

University of Colorado Boulder

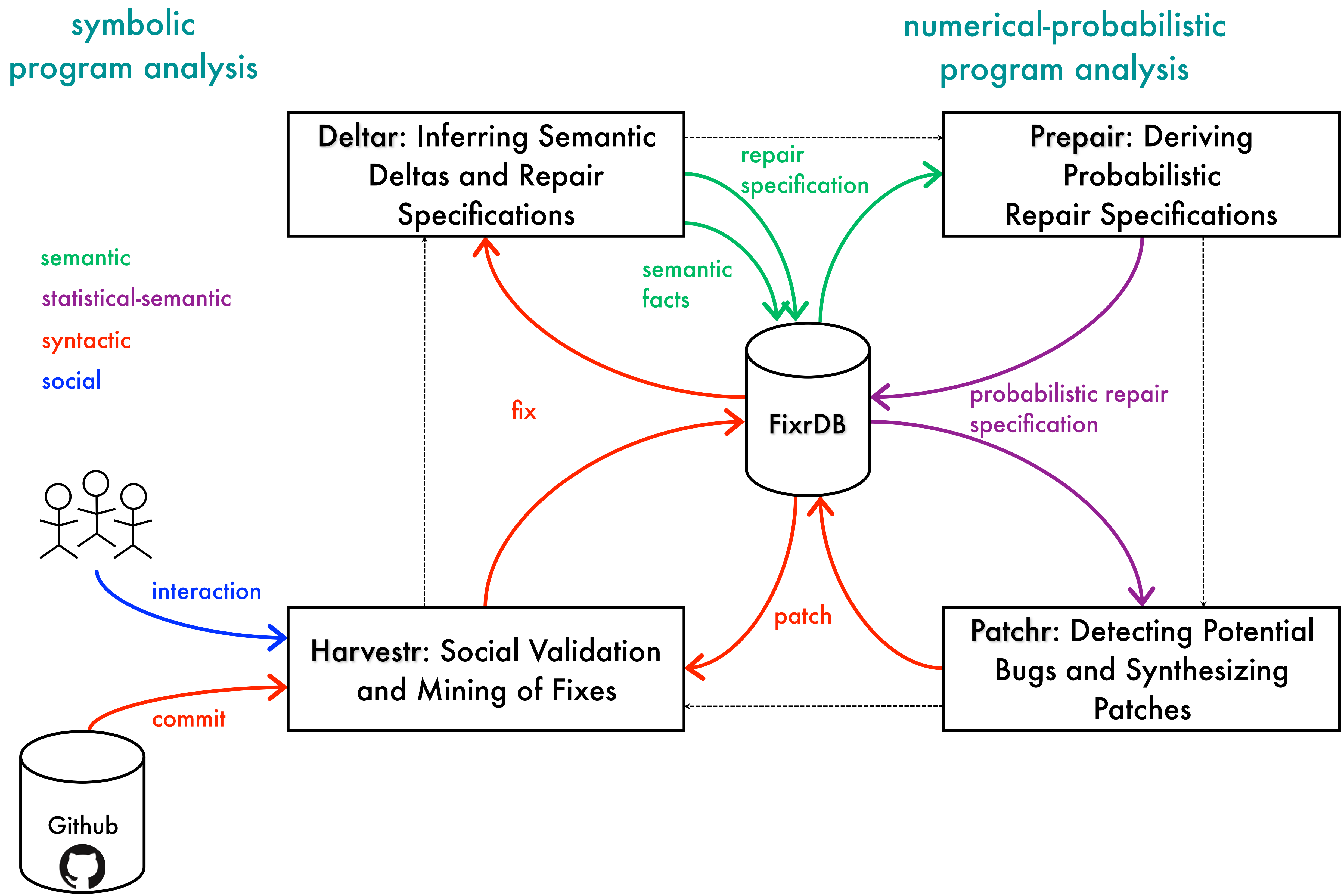
The **Fixr** Project



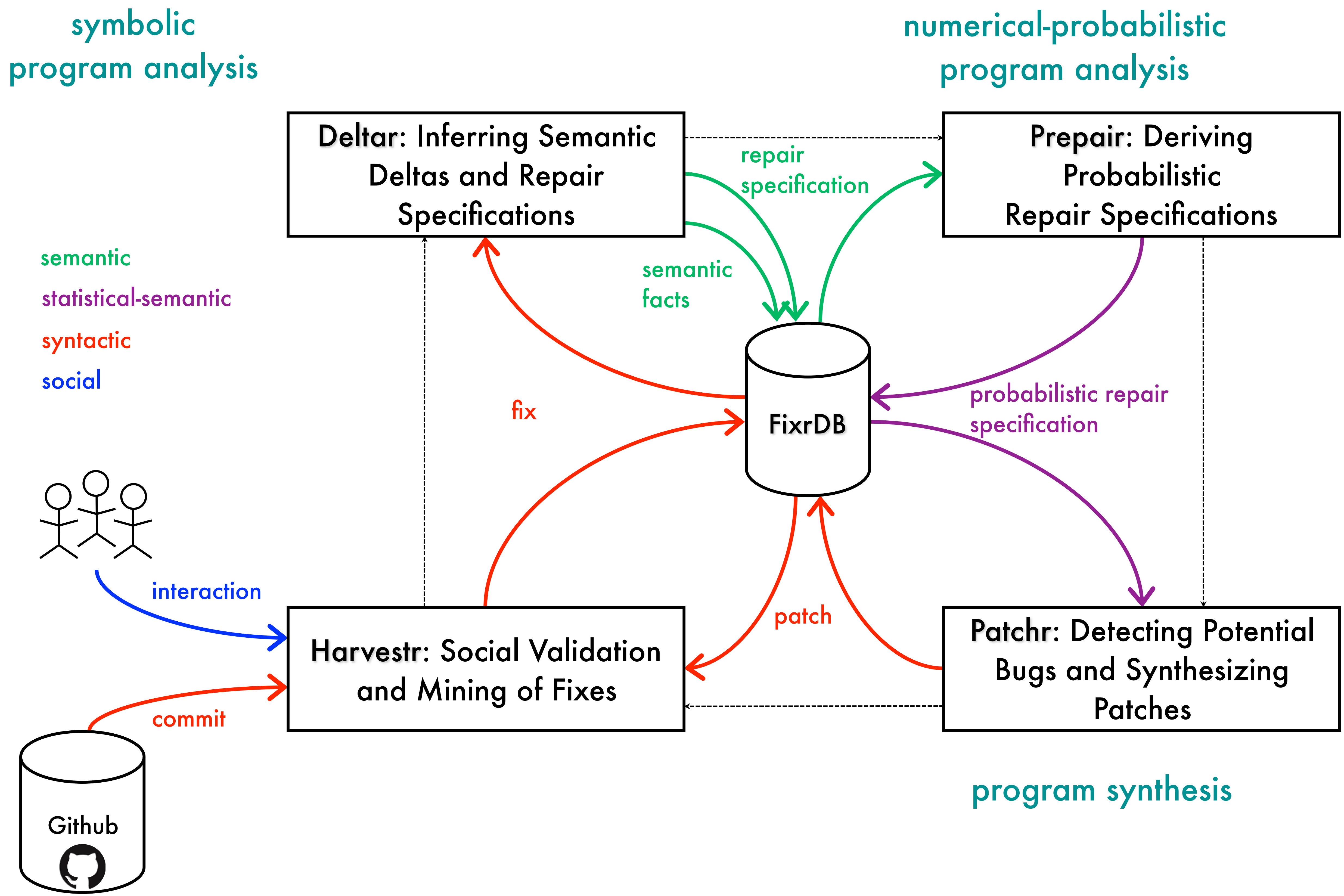
The **Fixr** Project



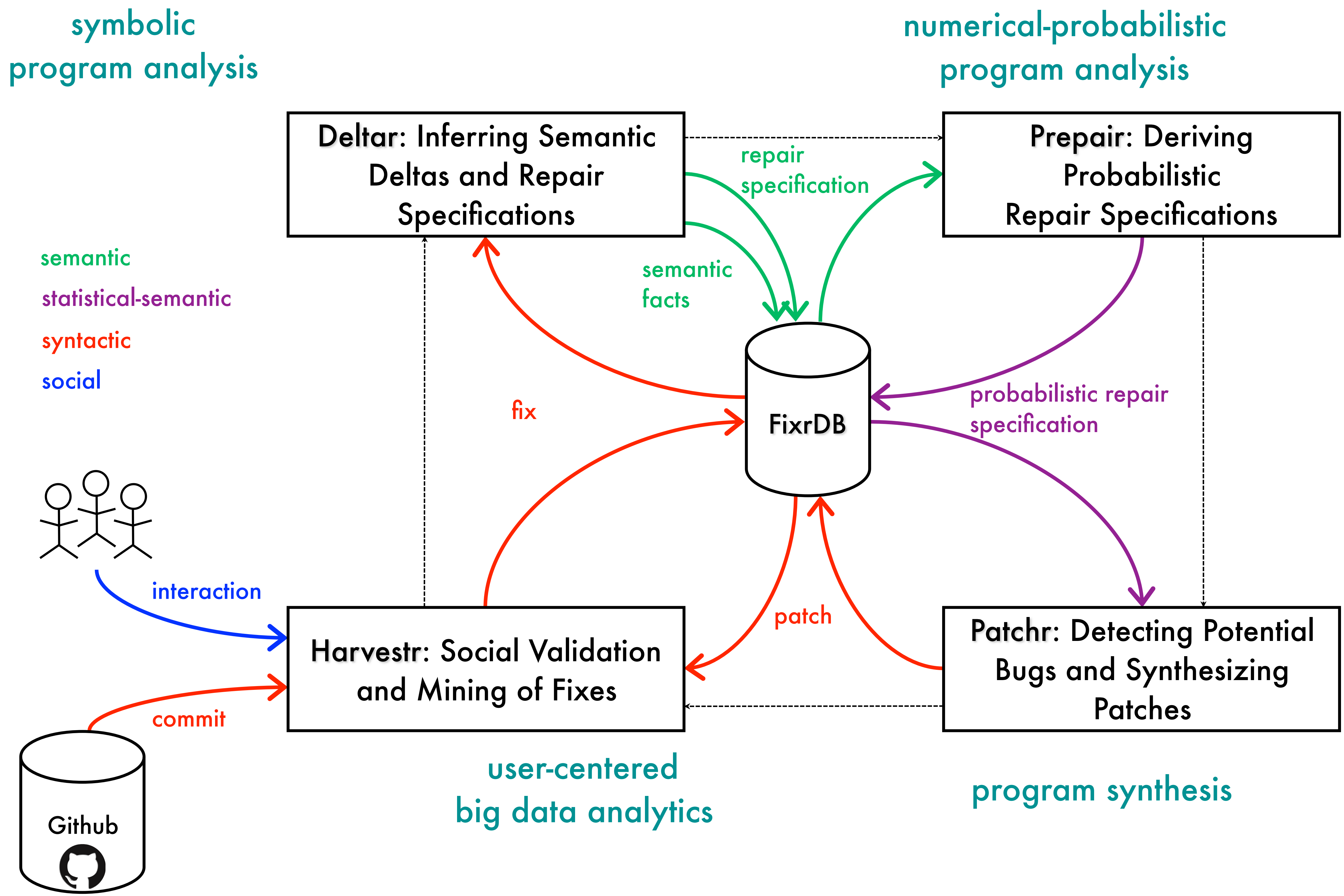
The **Fixr** Project



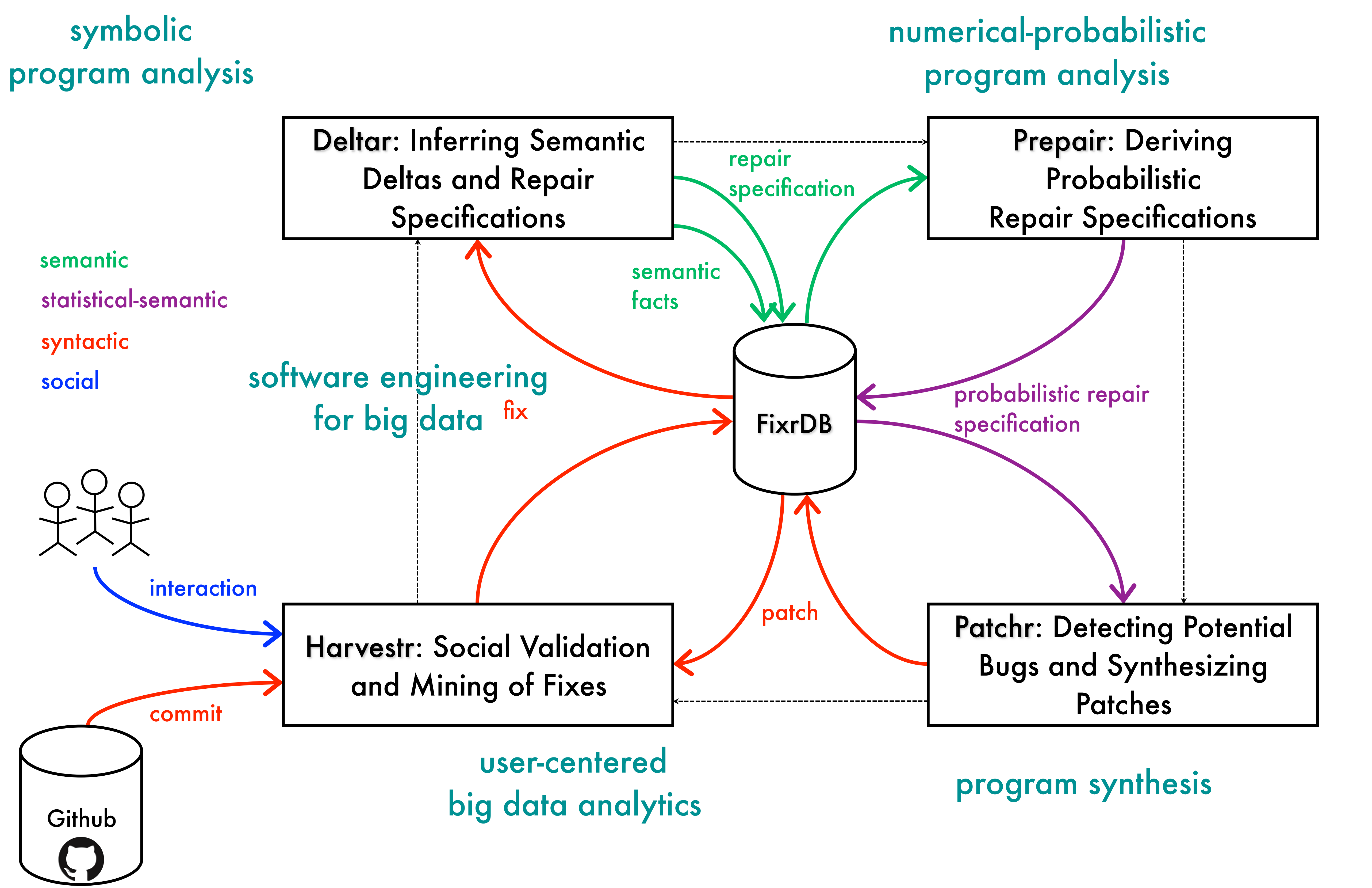
The **Fixr** Project



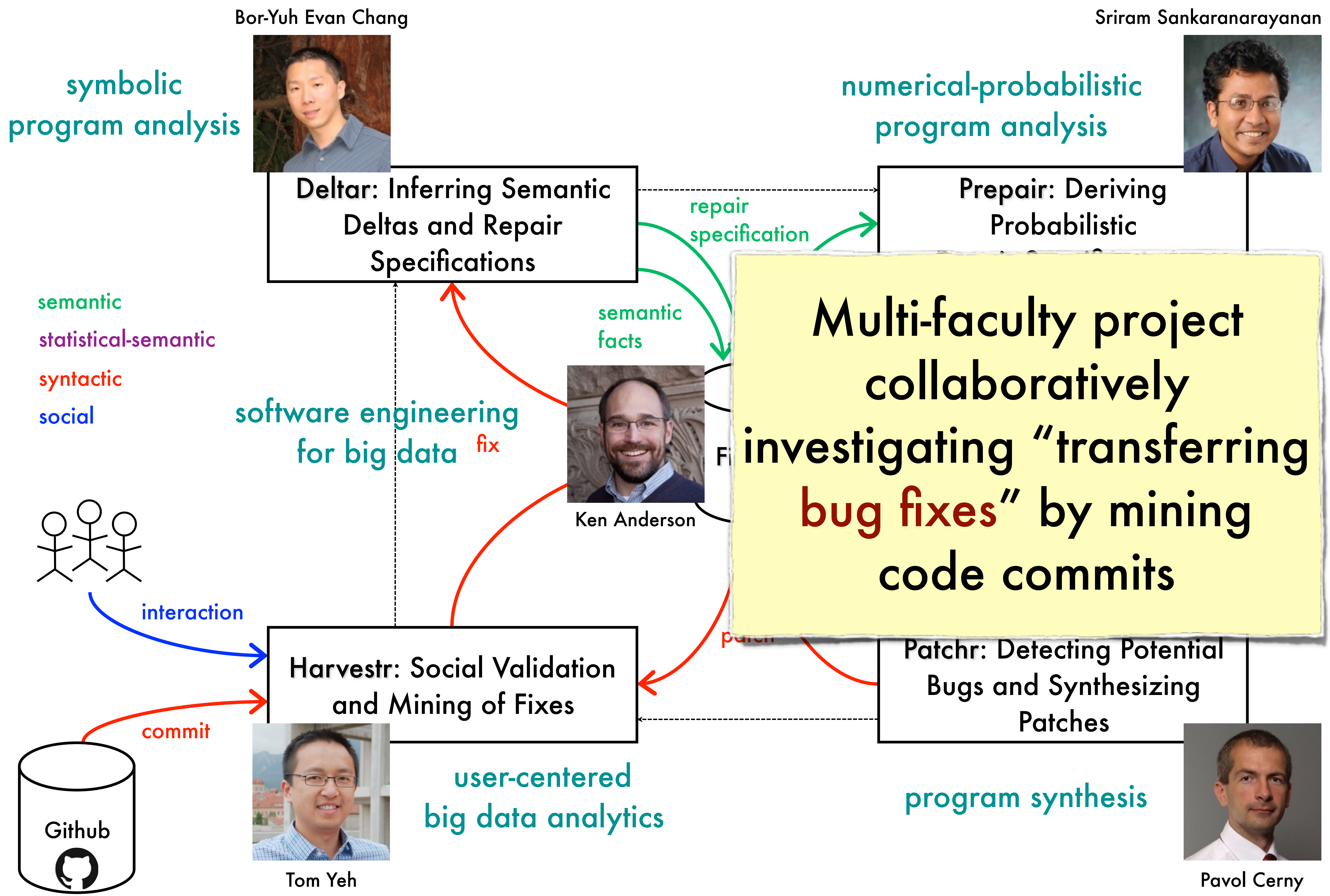
The **Fixr** Project



The **Fixr** Project



The **Fixr** Project



Multi-faculty project collaboratively investigating "transferring **bug fixes**" by mining code commits

The **Fixr** Project

Bor-Yuh Evan Chang

symbolic
program analysis

Sriram Sankaranarayanan

numerical-probabilistic
program analysis

Deltar: Inferring Semantic

Prepair: Deriving

roject
ely
nsferring
nining
its

otential
sizing

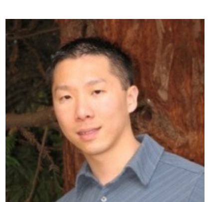
s

Pavol Cerny

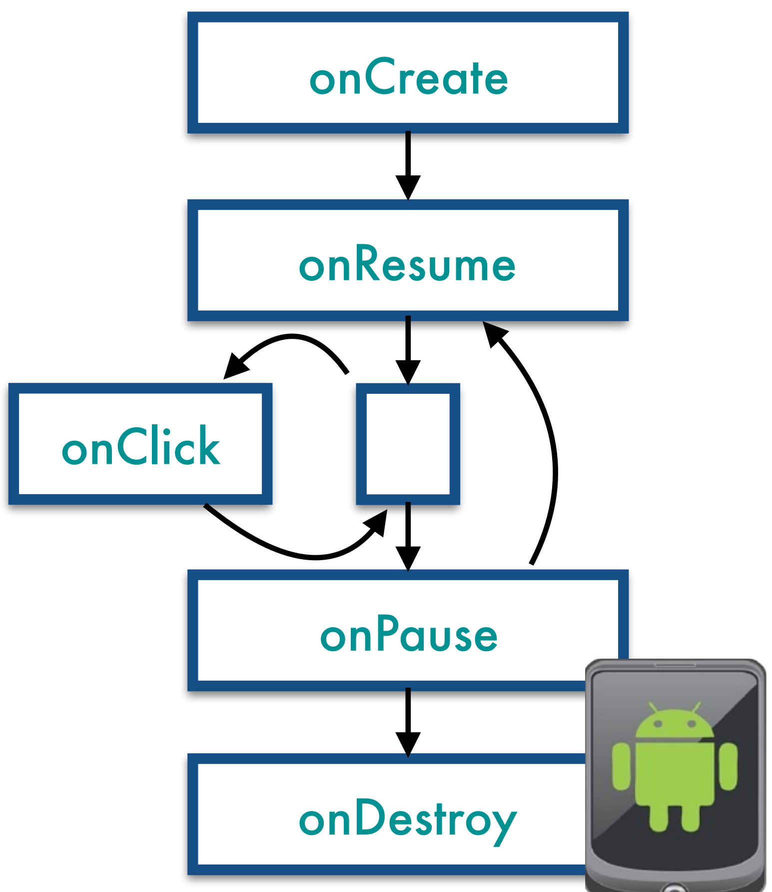
Abstracting Event-Driven Systems with Lifestate Specifications

Shawn Meier

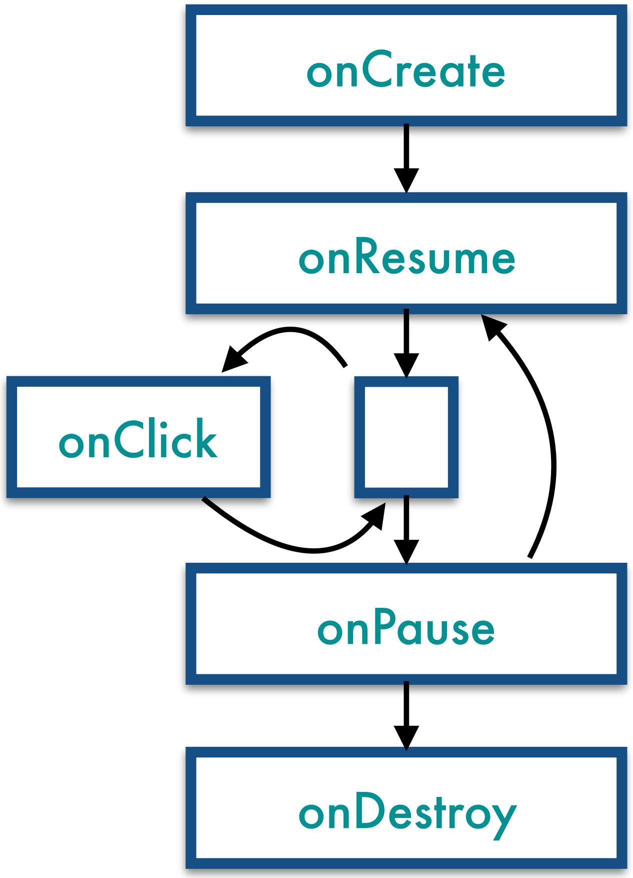
Sergio Mover



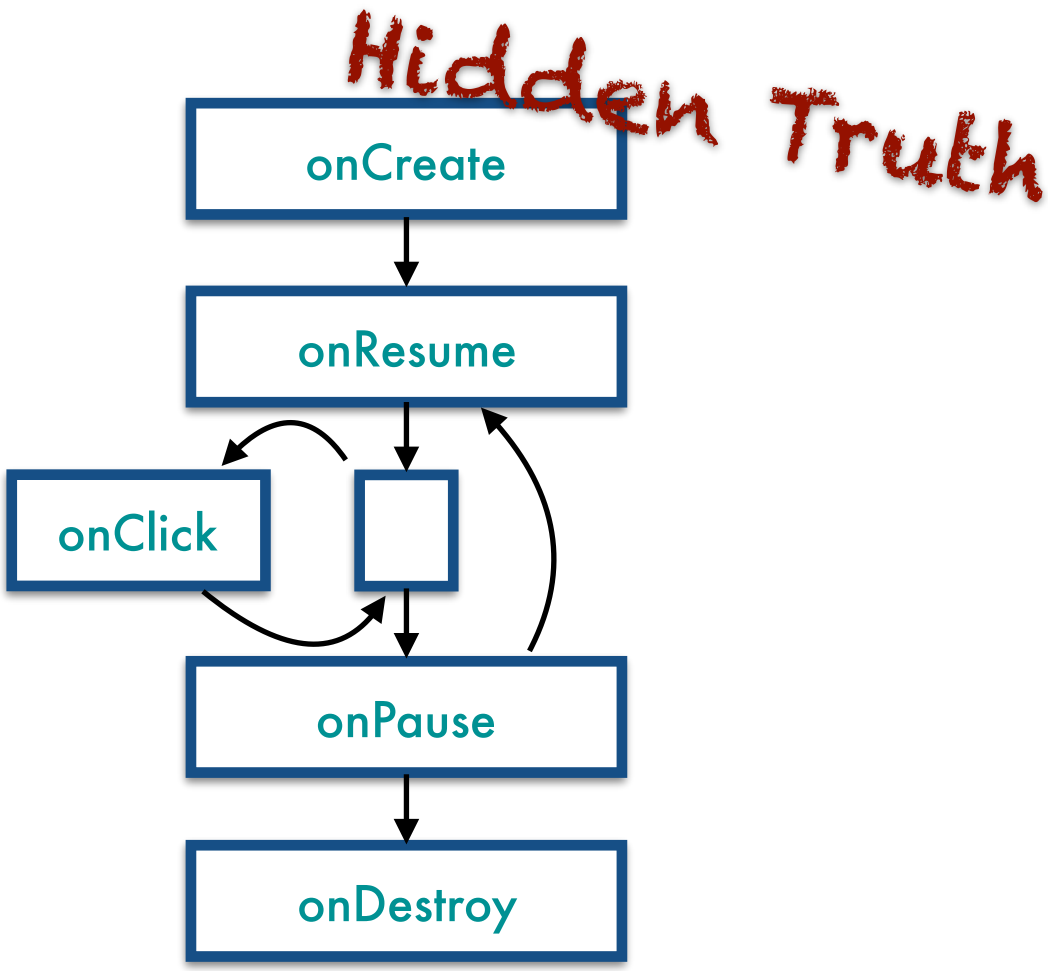
Bor-Yuh Evan Chang



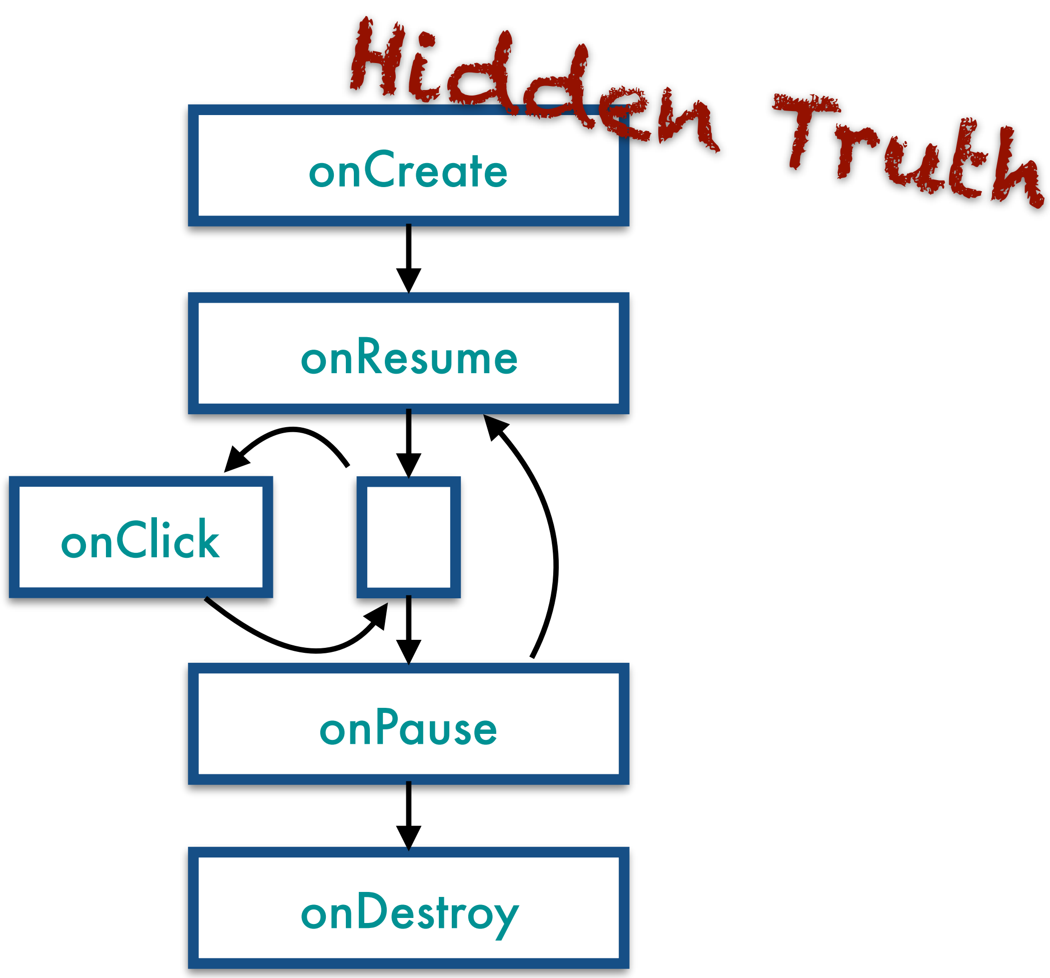
Android is an event-driven system



Android is an event-driven system



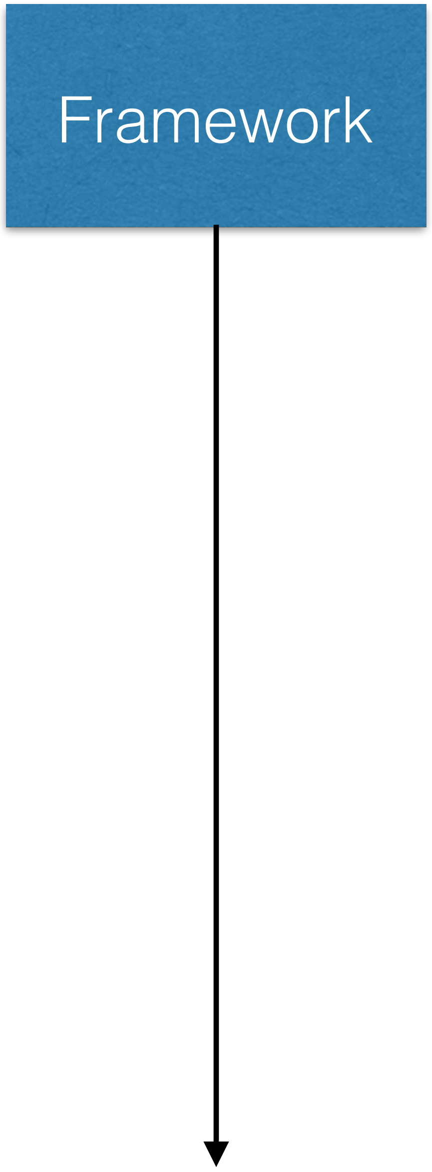
Android is an event-driven system



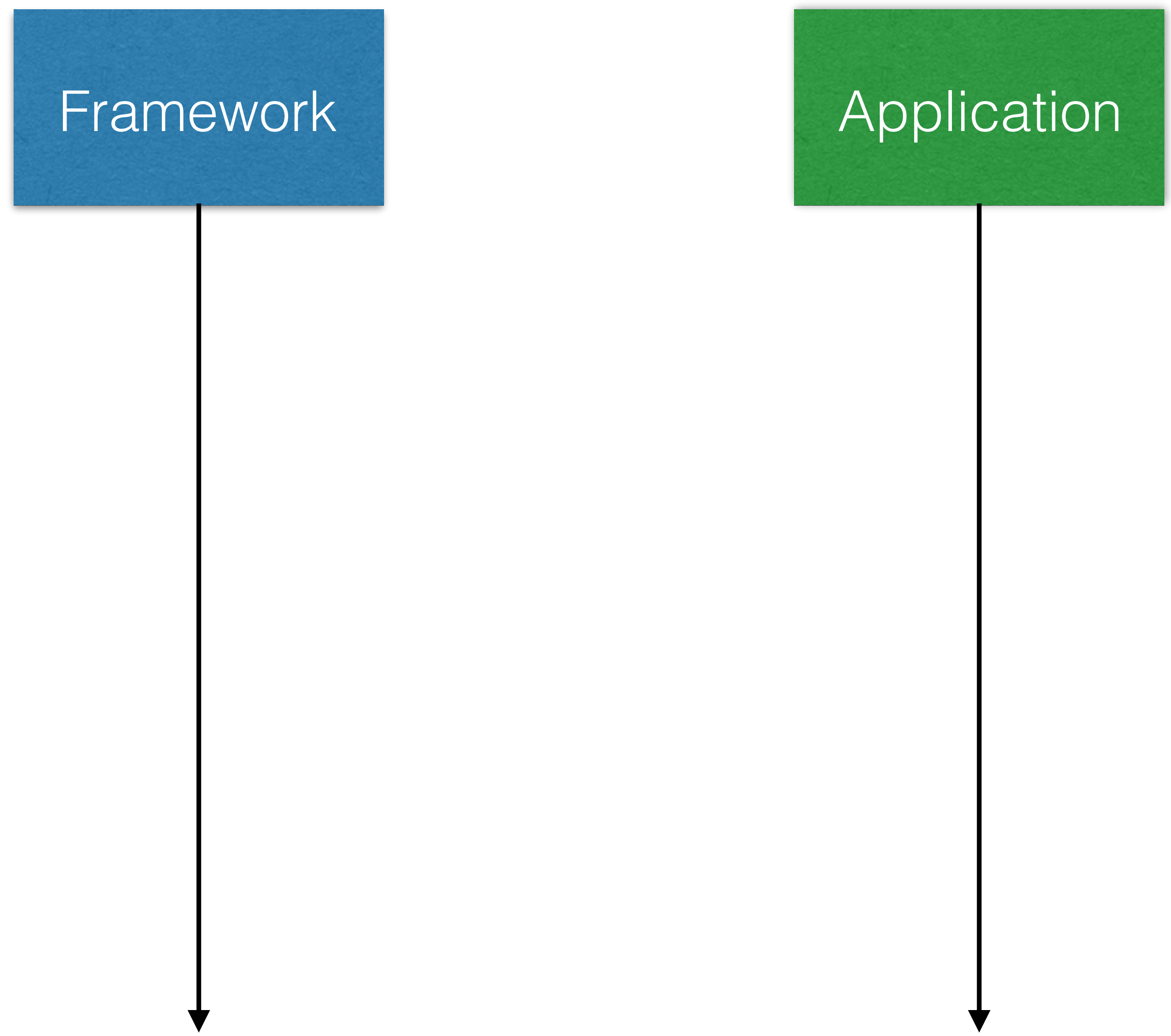
Callback ordering constraints are not static

Android is an event-driven system

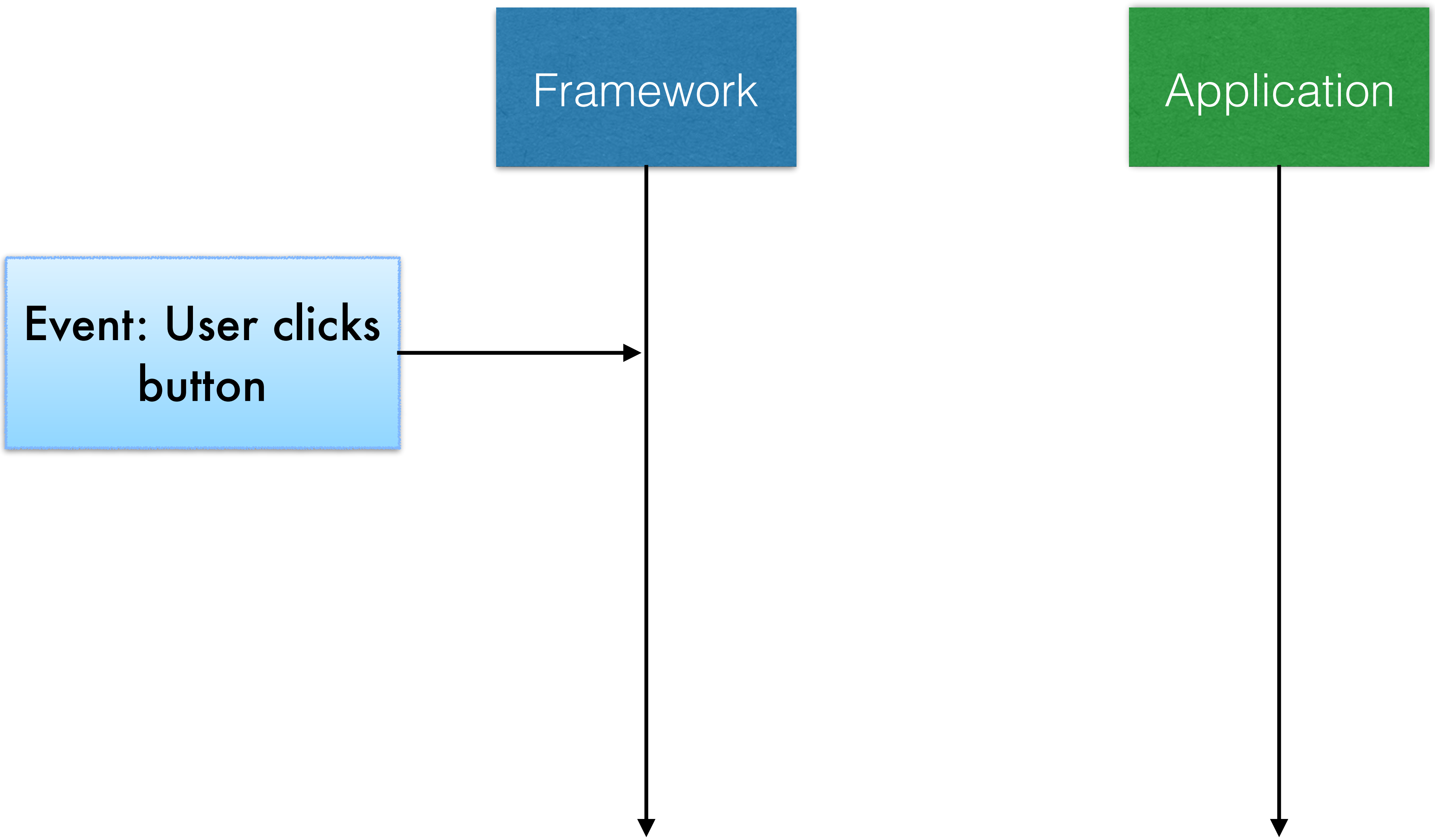
Android is an event-driven system



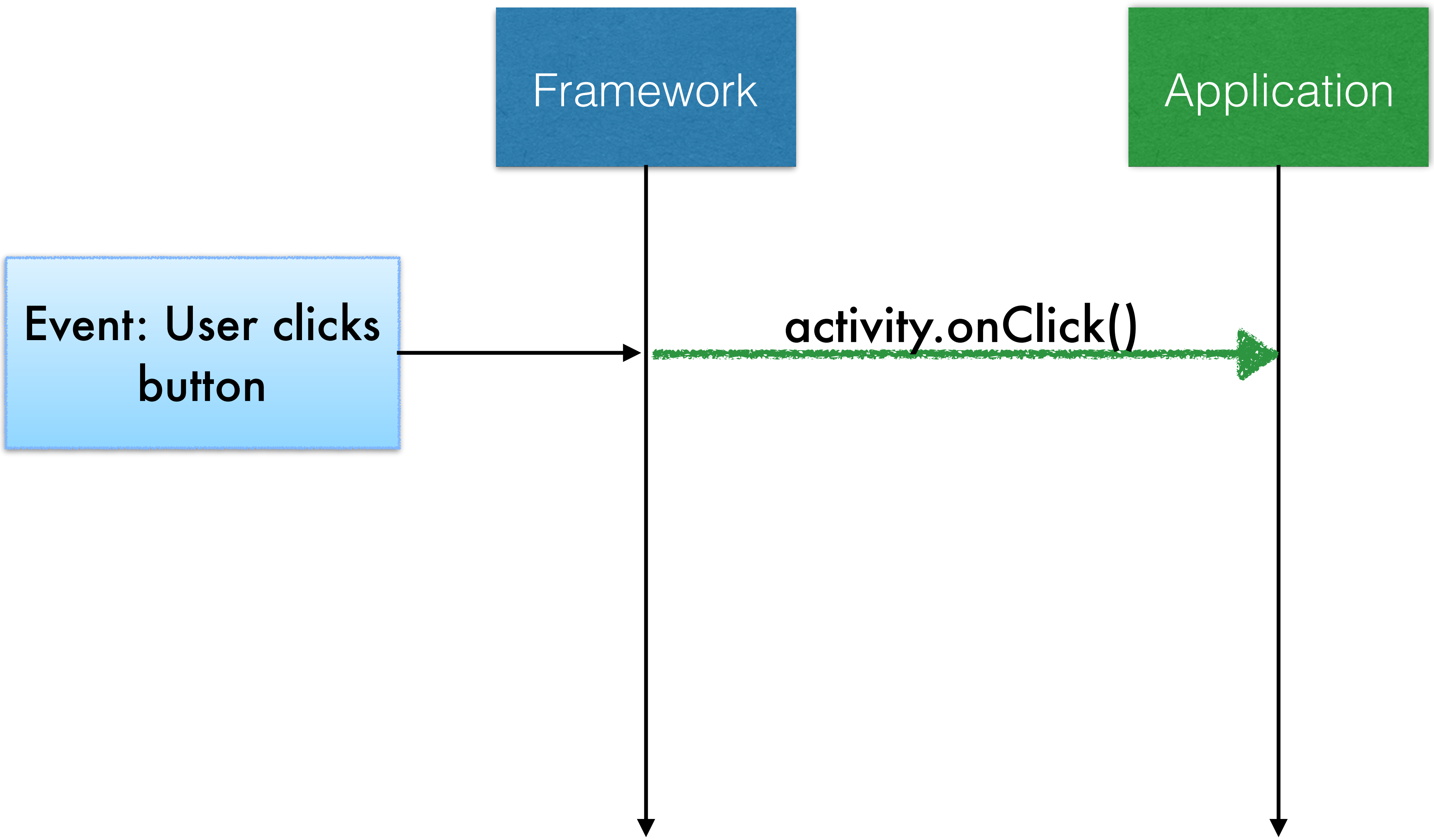
Android is an event-driven system



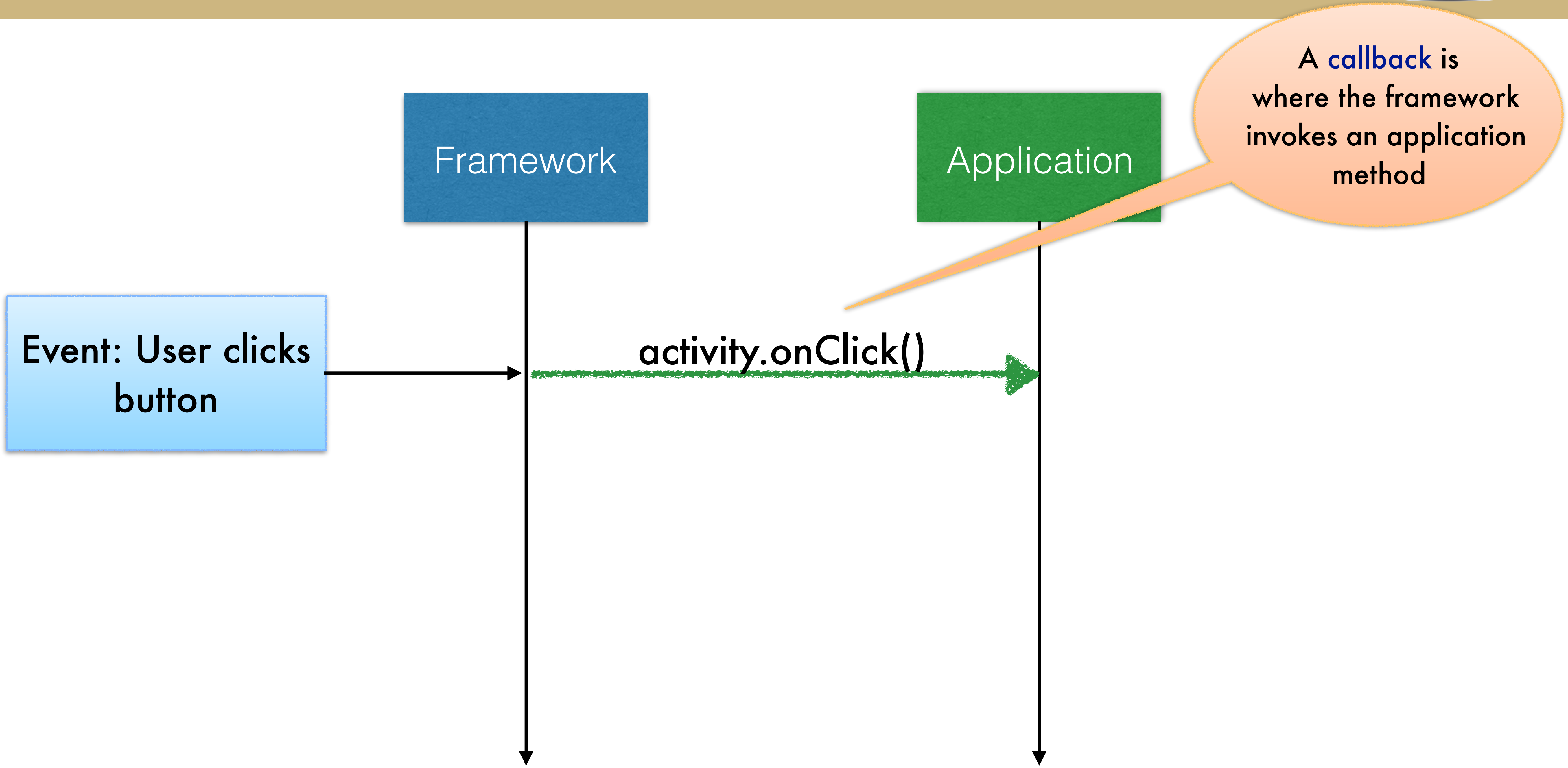
Android is an event-driven system



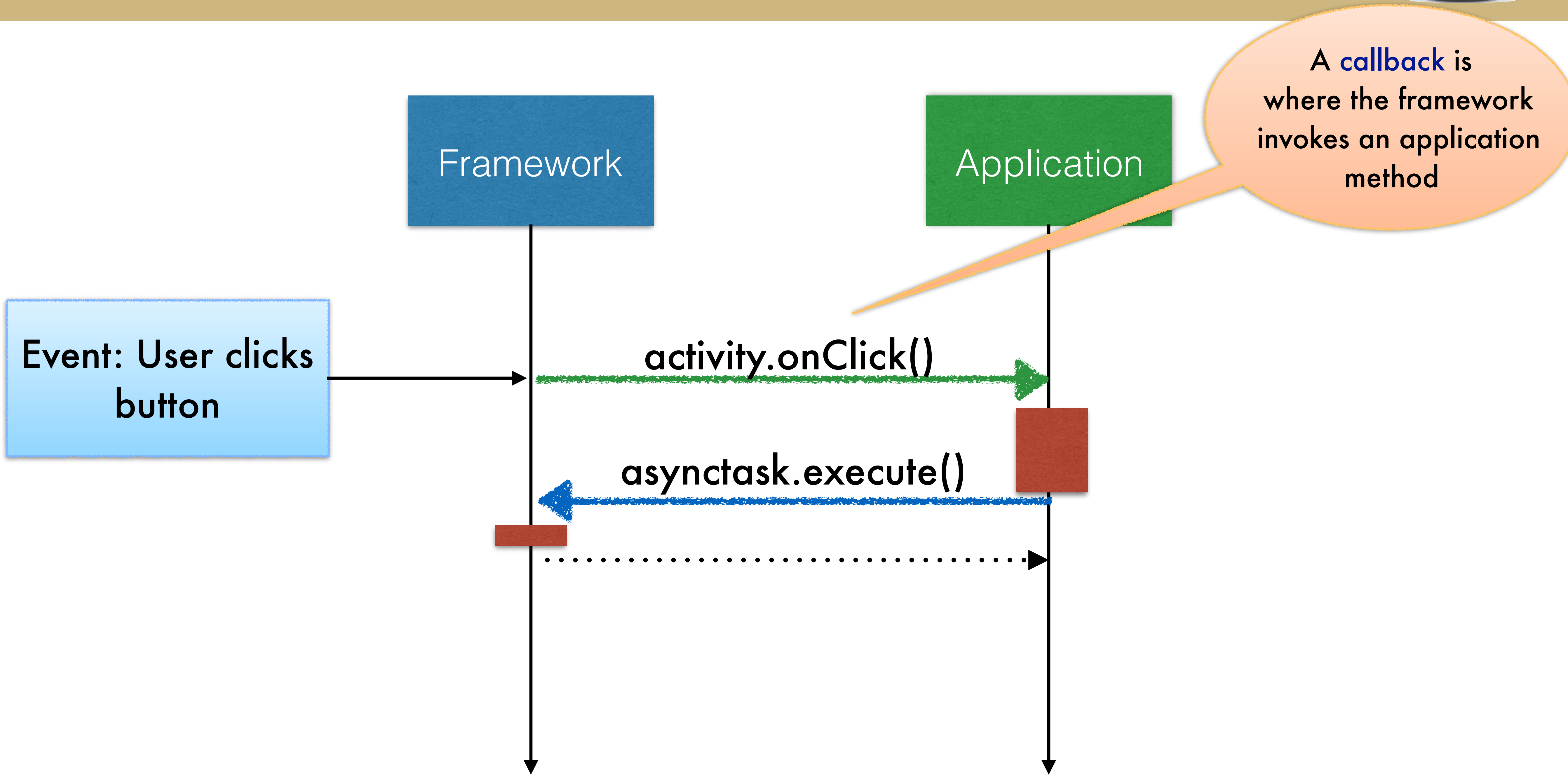
Android is an event-driven system



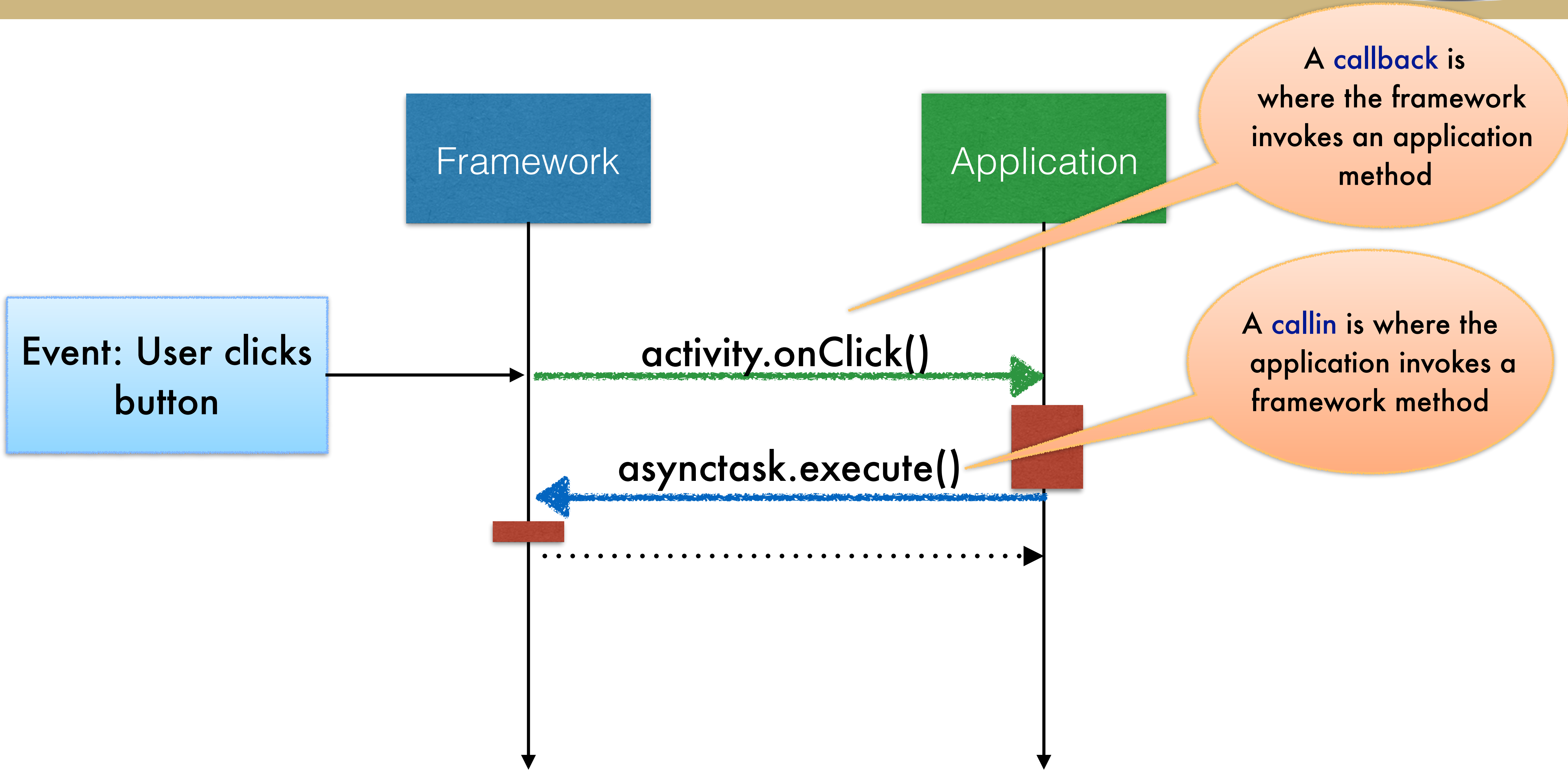
Android is an event-driven system



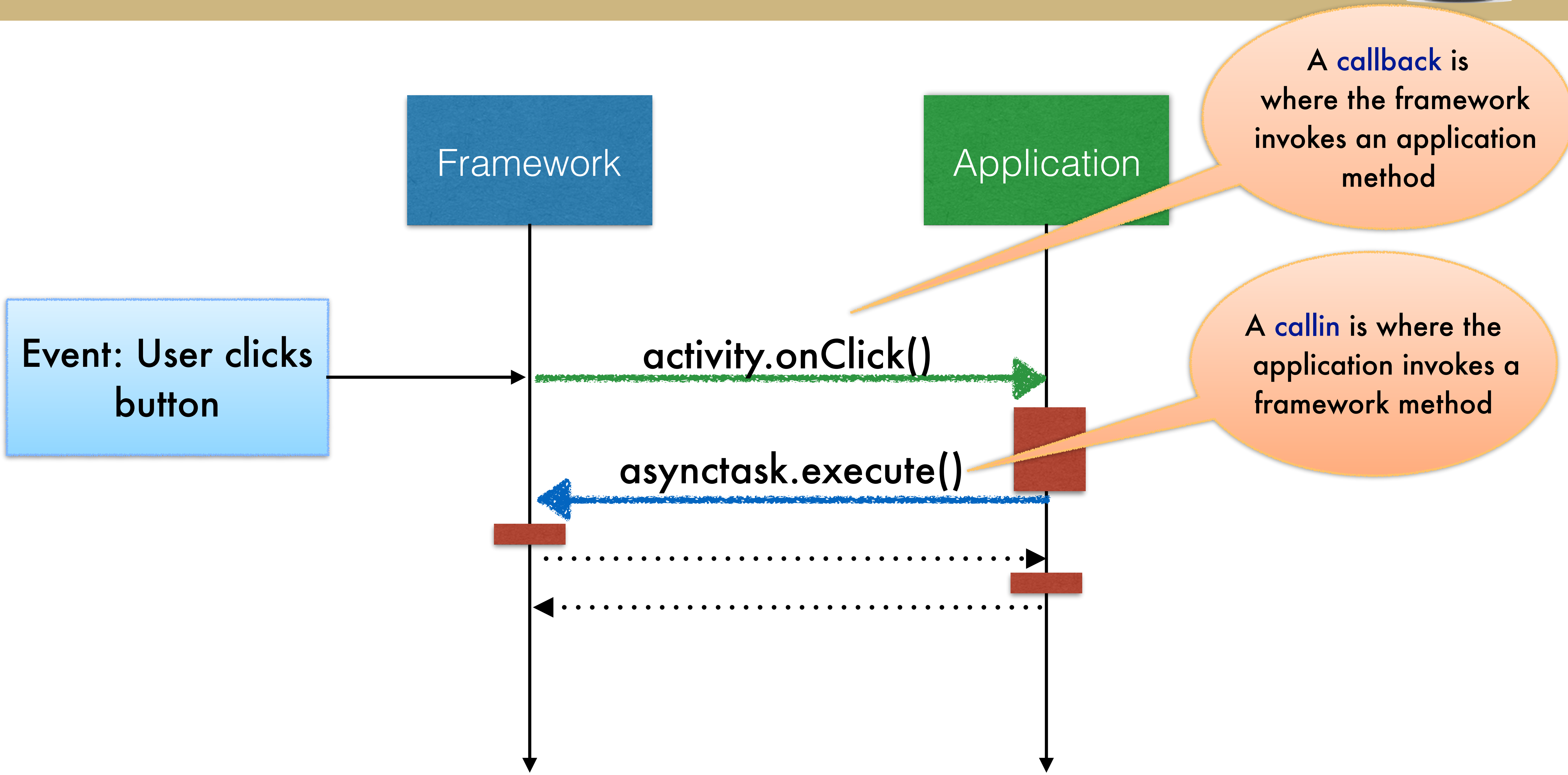
Android is an event-driven system



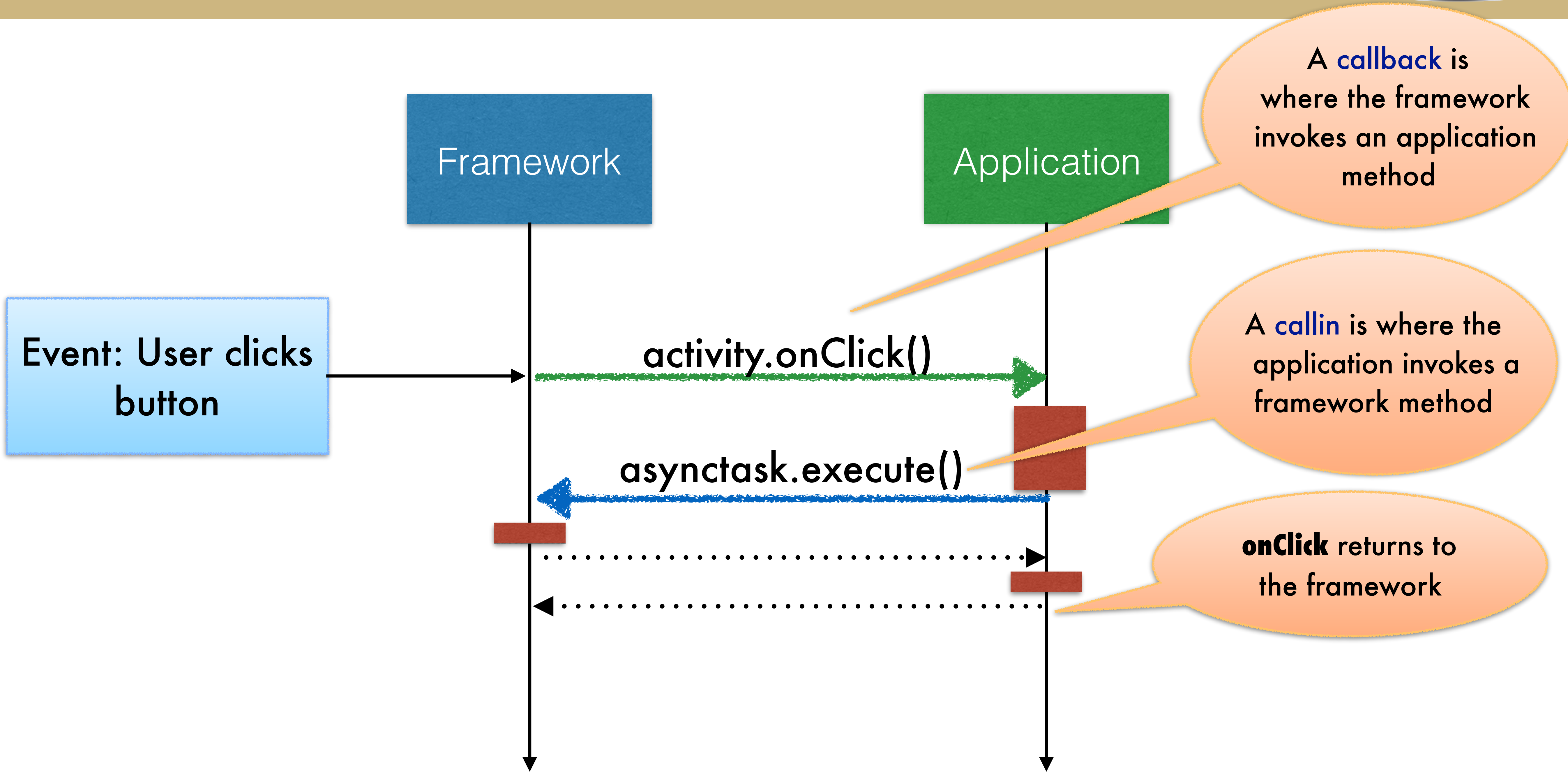
Android is an event-driven system



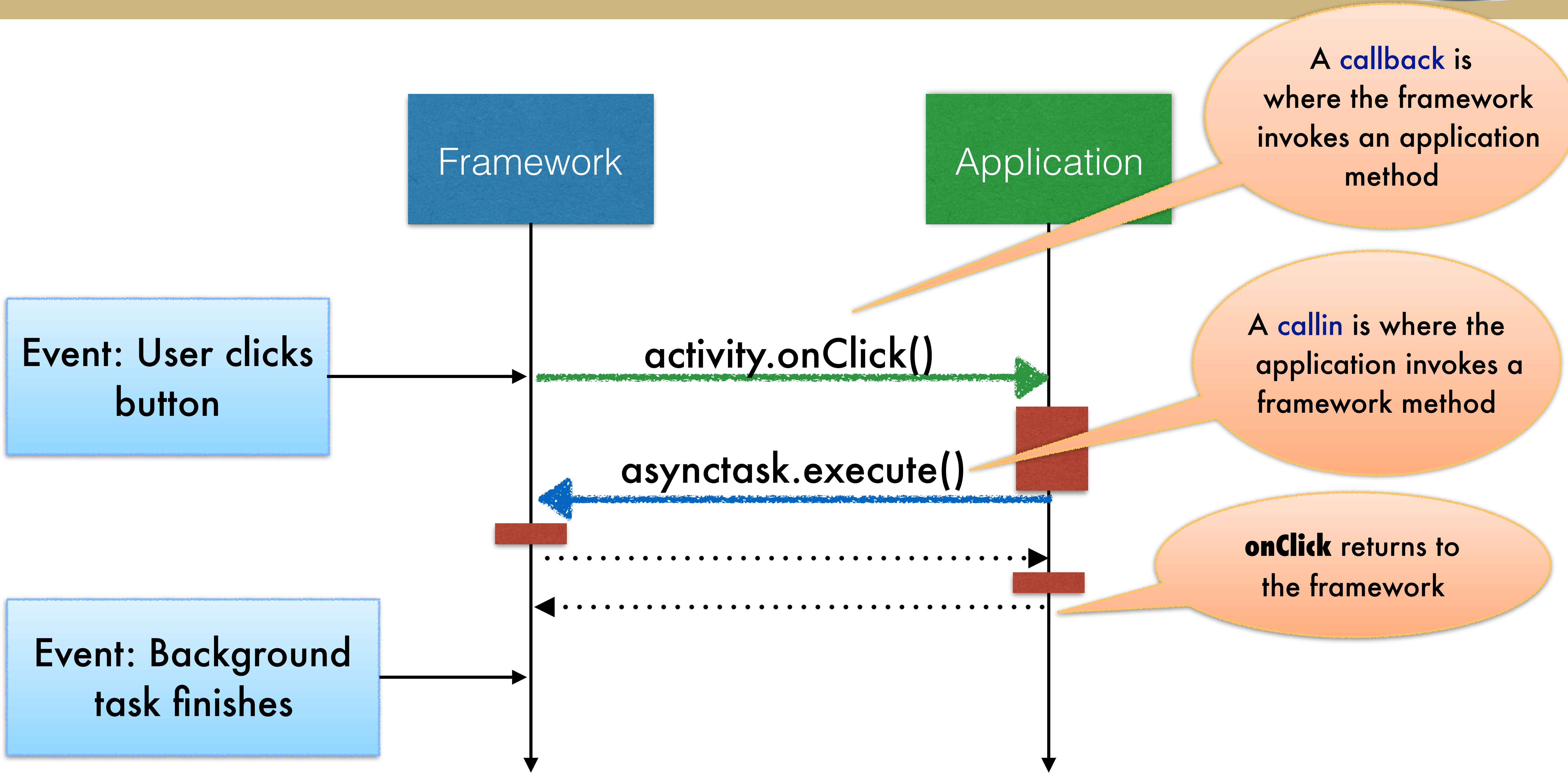
Android is an event-driven system



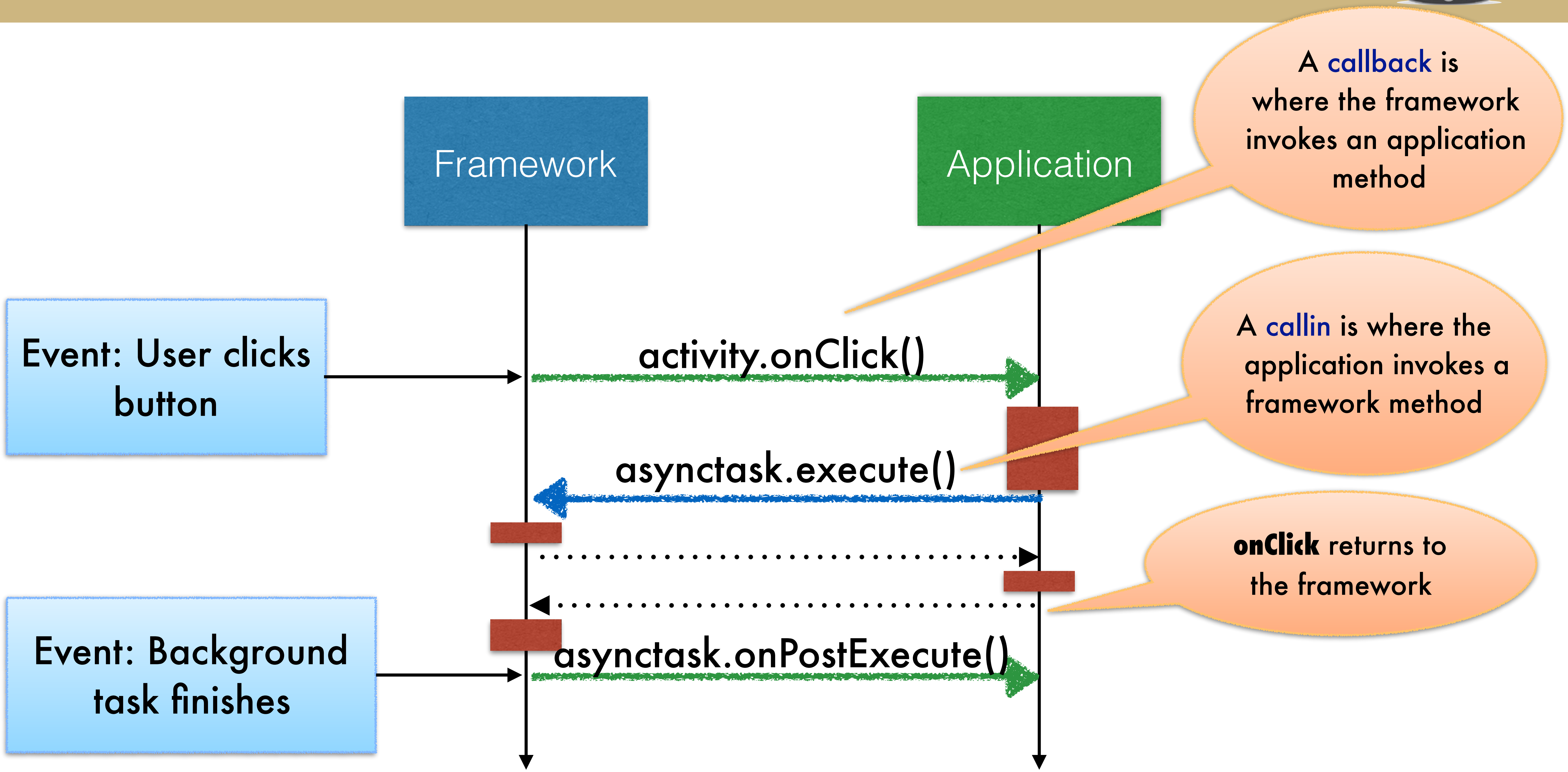
Android is an event-driven system



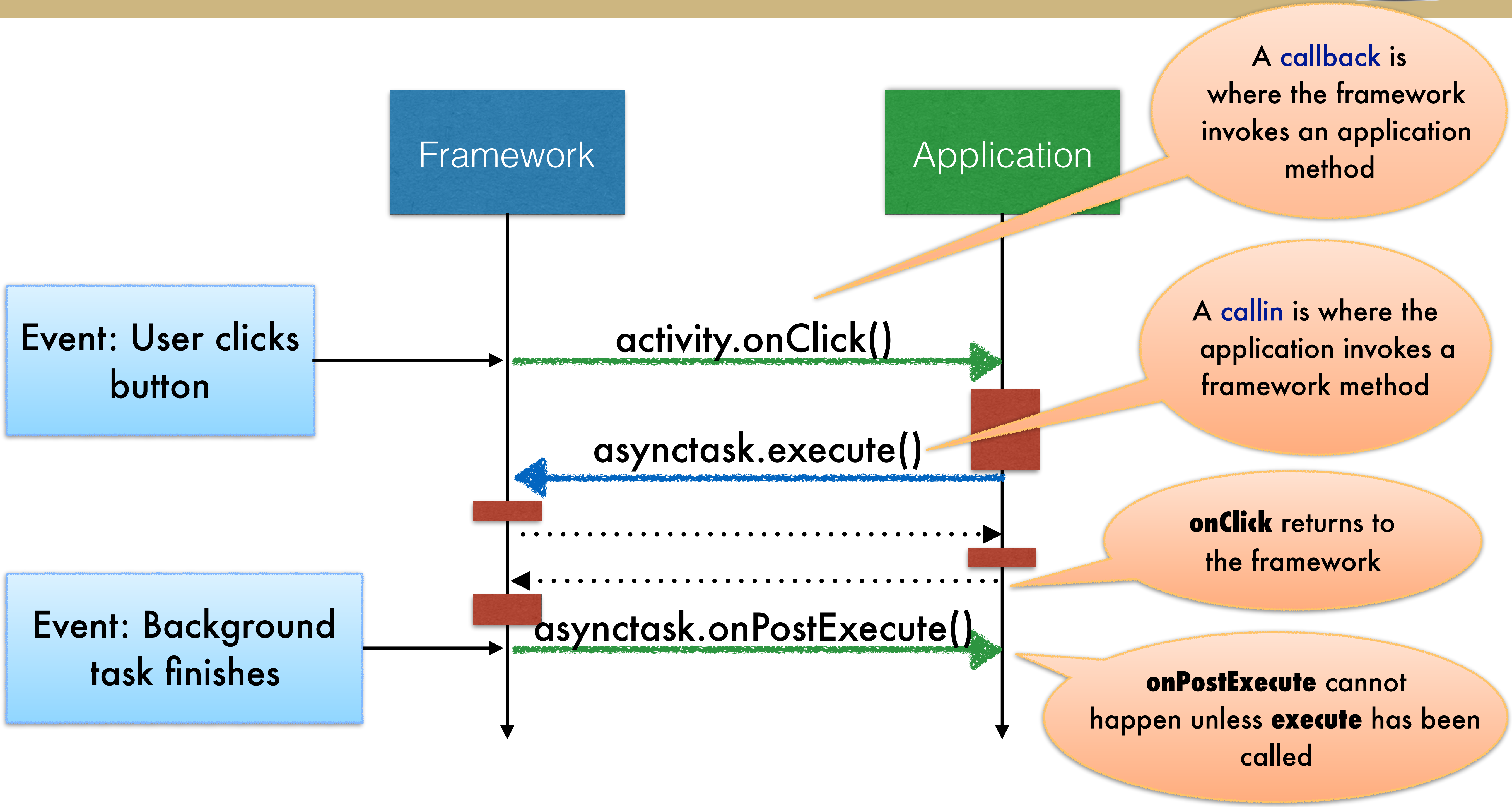
Android is an event-driven system



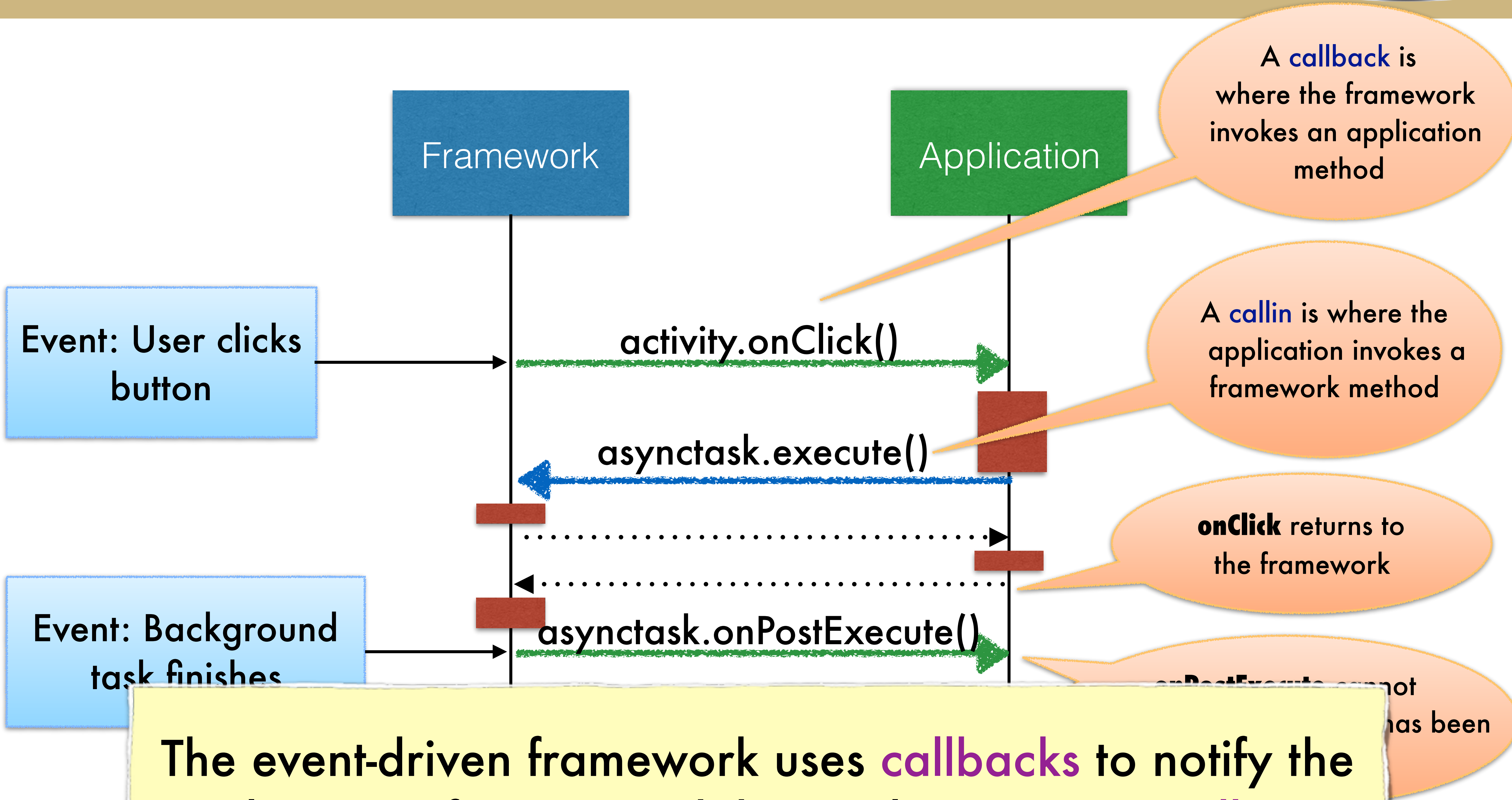
Android is an event-driven system



Android is an event-driven system

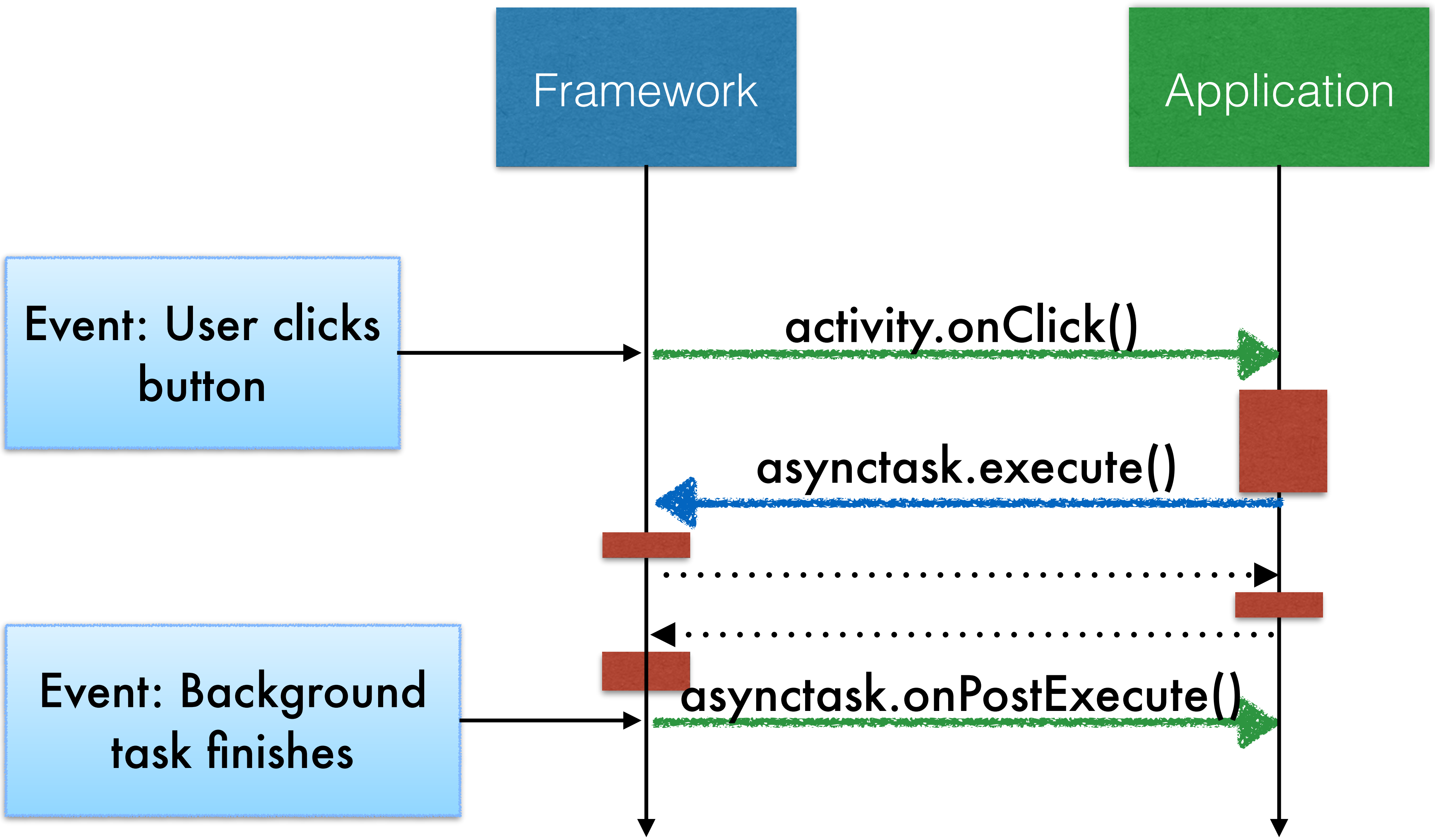


Android is an event-driven system

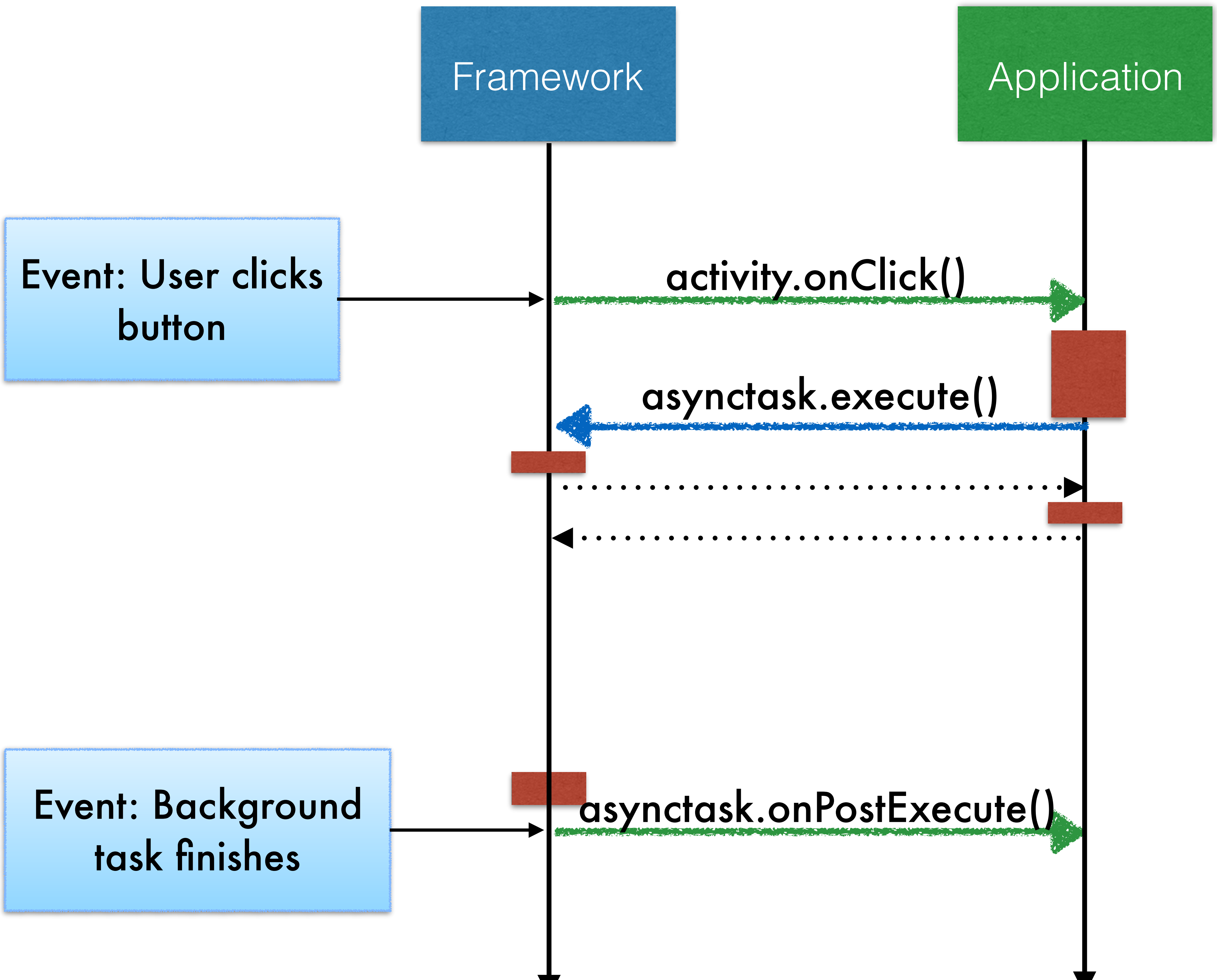


The event-driven framework uses callbacks to notify the application of events and the application uses callins to affect how the framework invokes future callbacks.

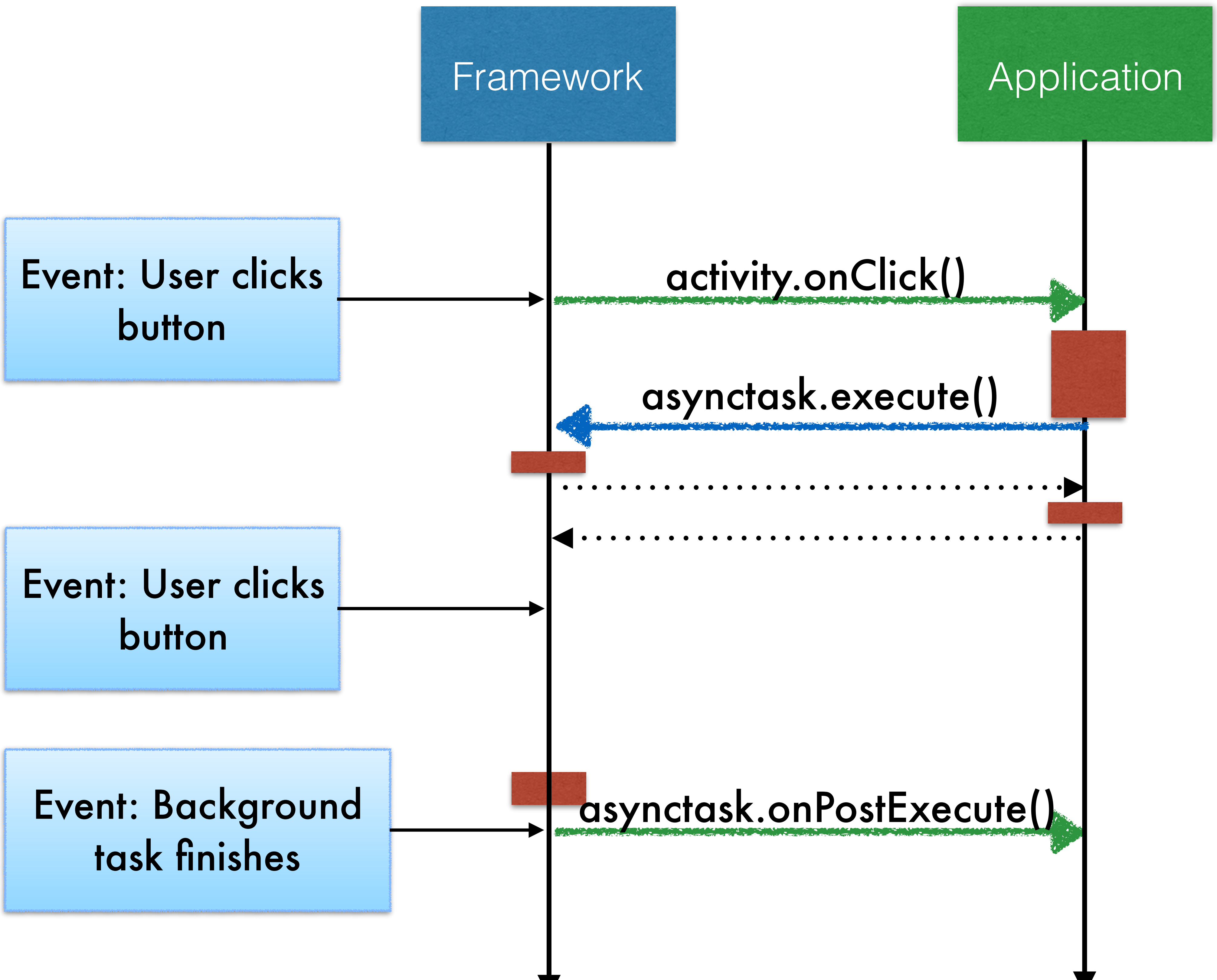
Android is an event-driven system



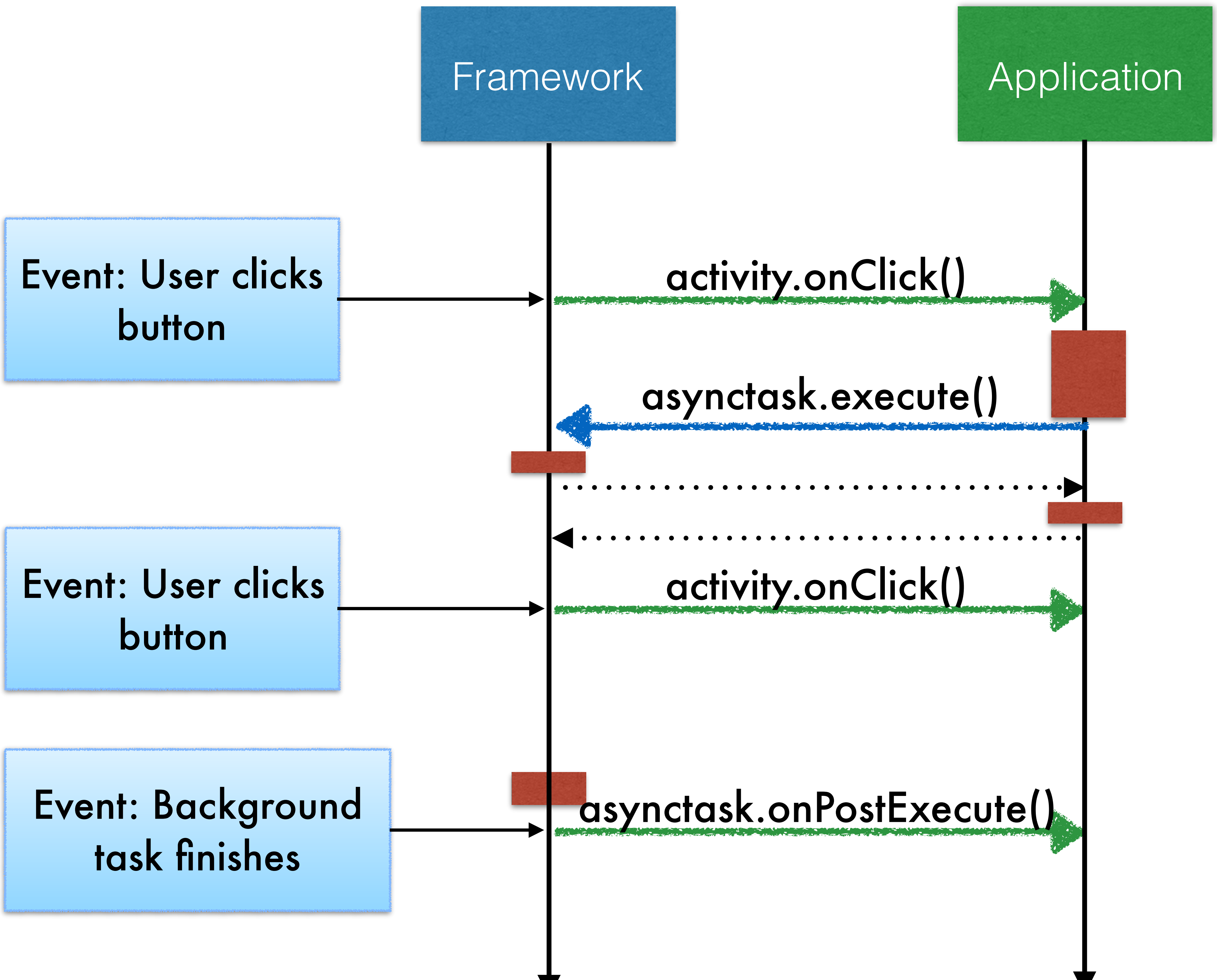
Android is an event-driven system



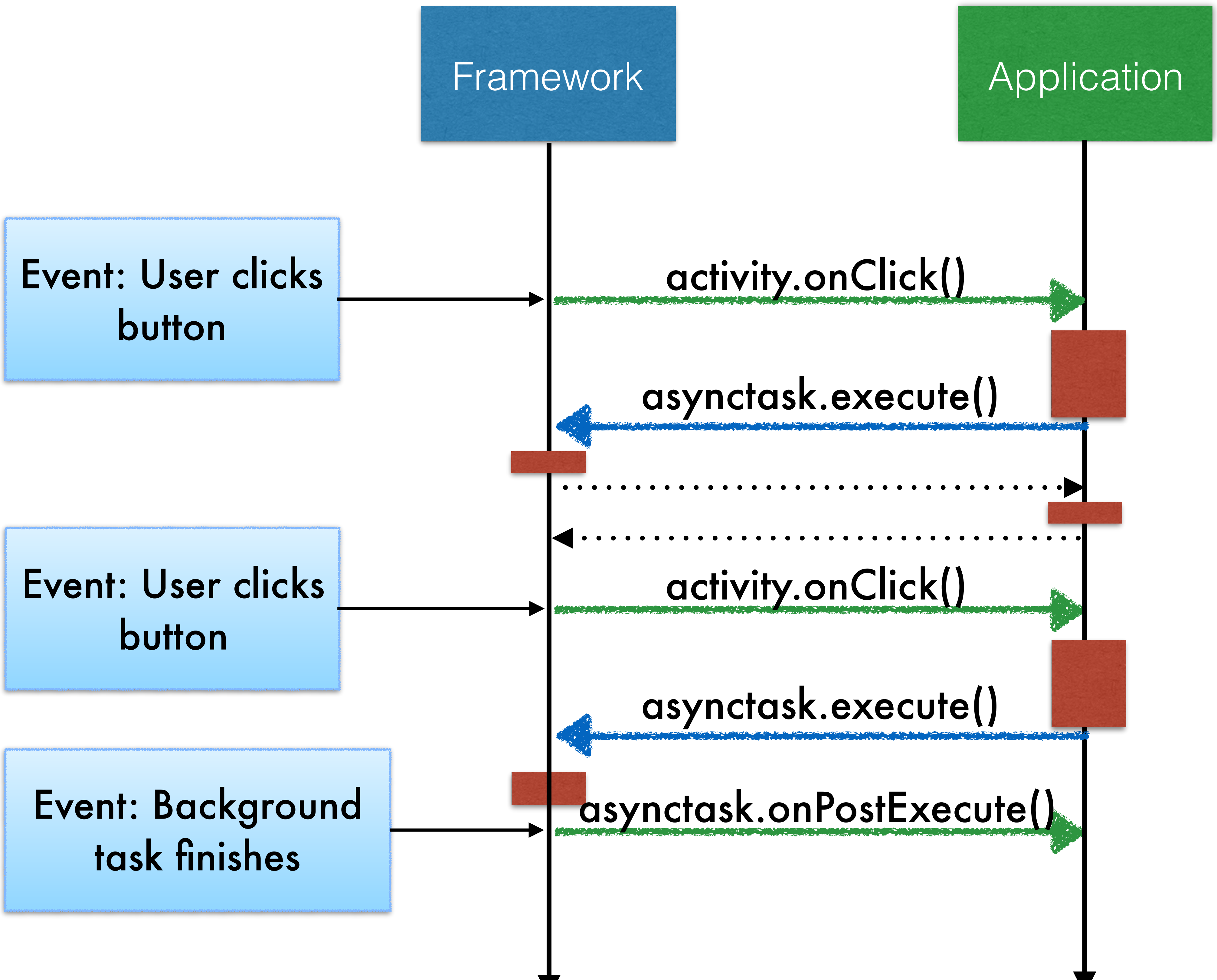
Android is an event-driven system



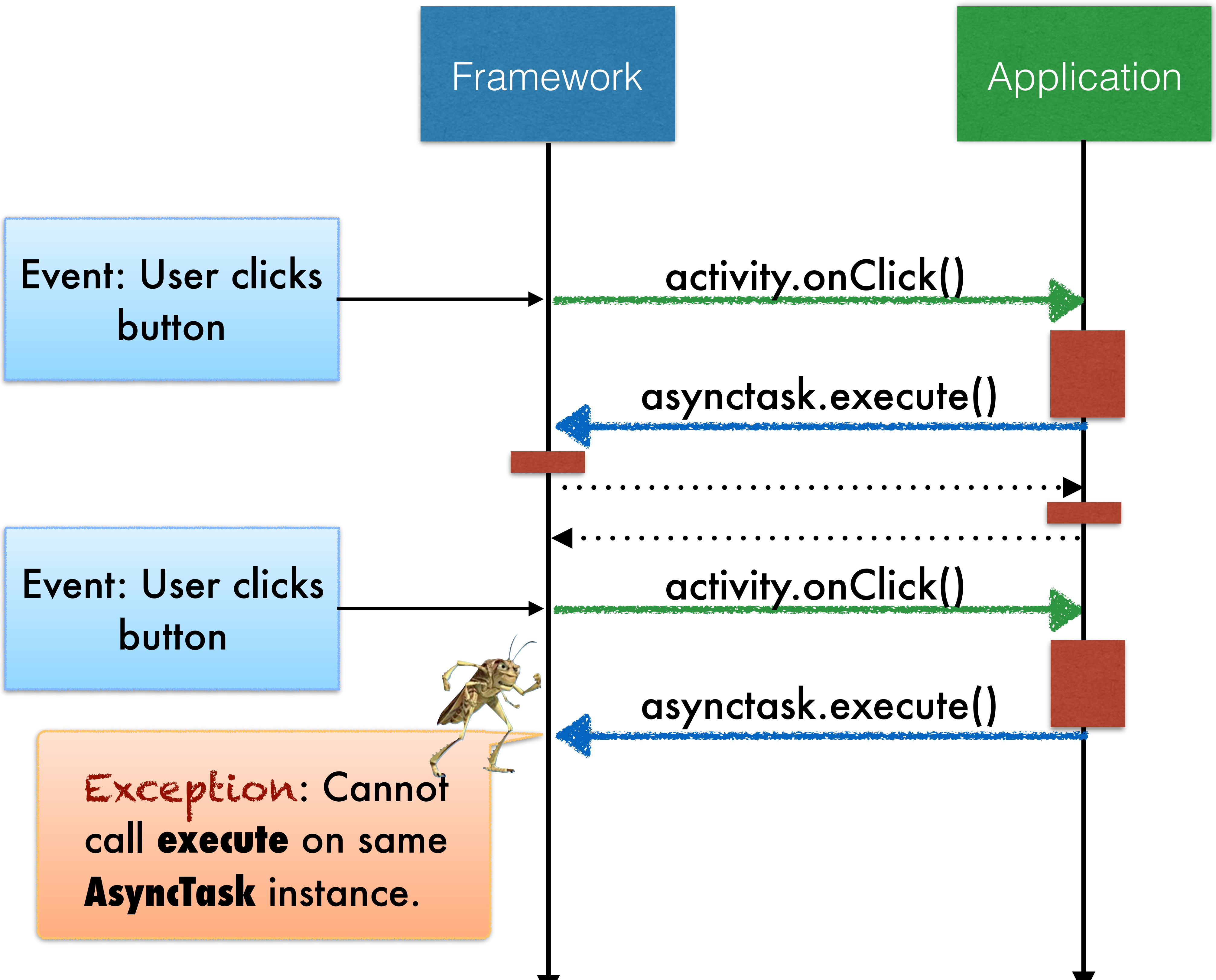
Android is an event-driven system



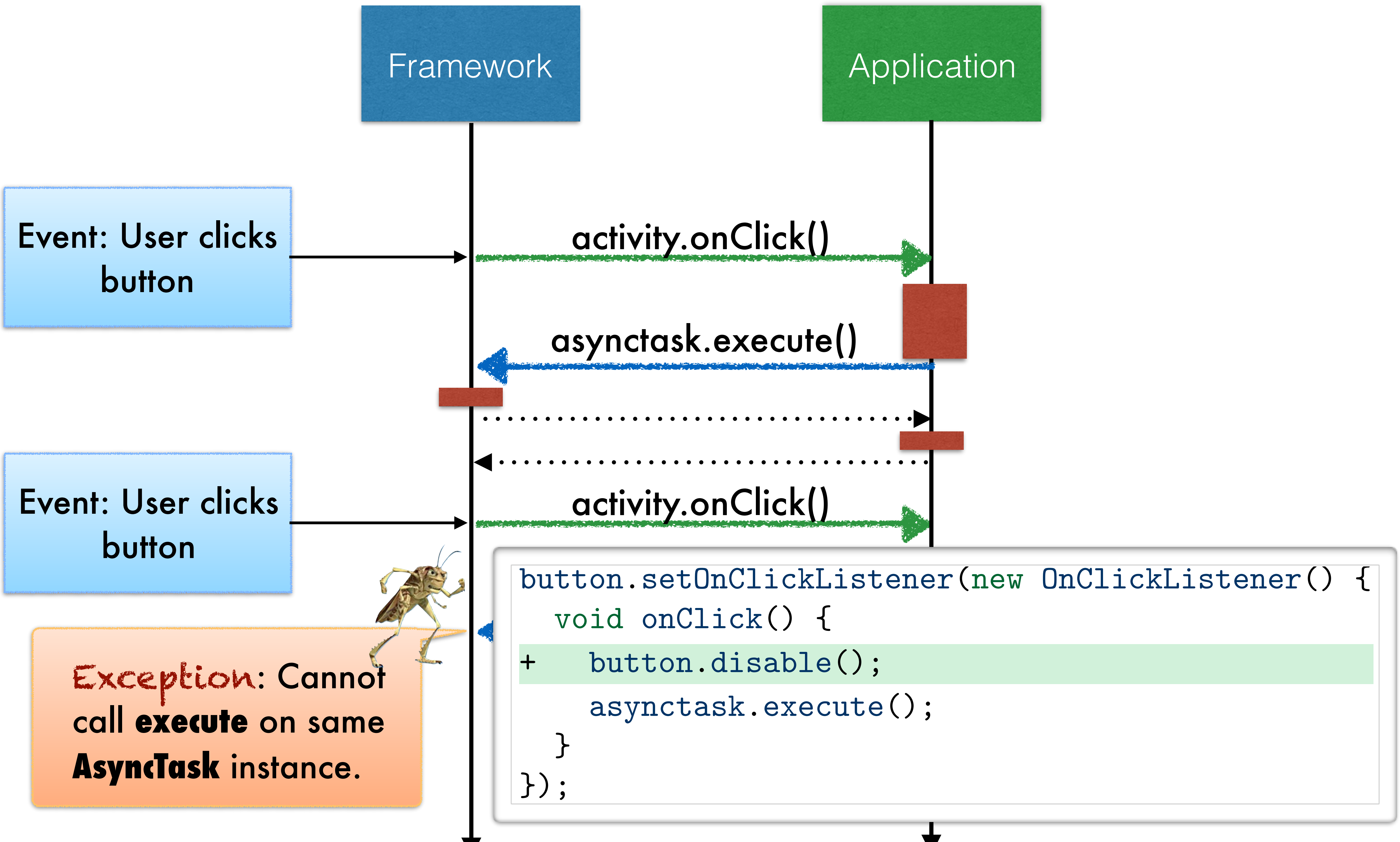
Android is an event-driven system



Android is an event-driven system

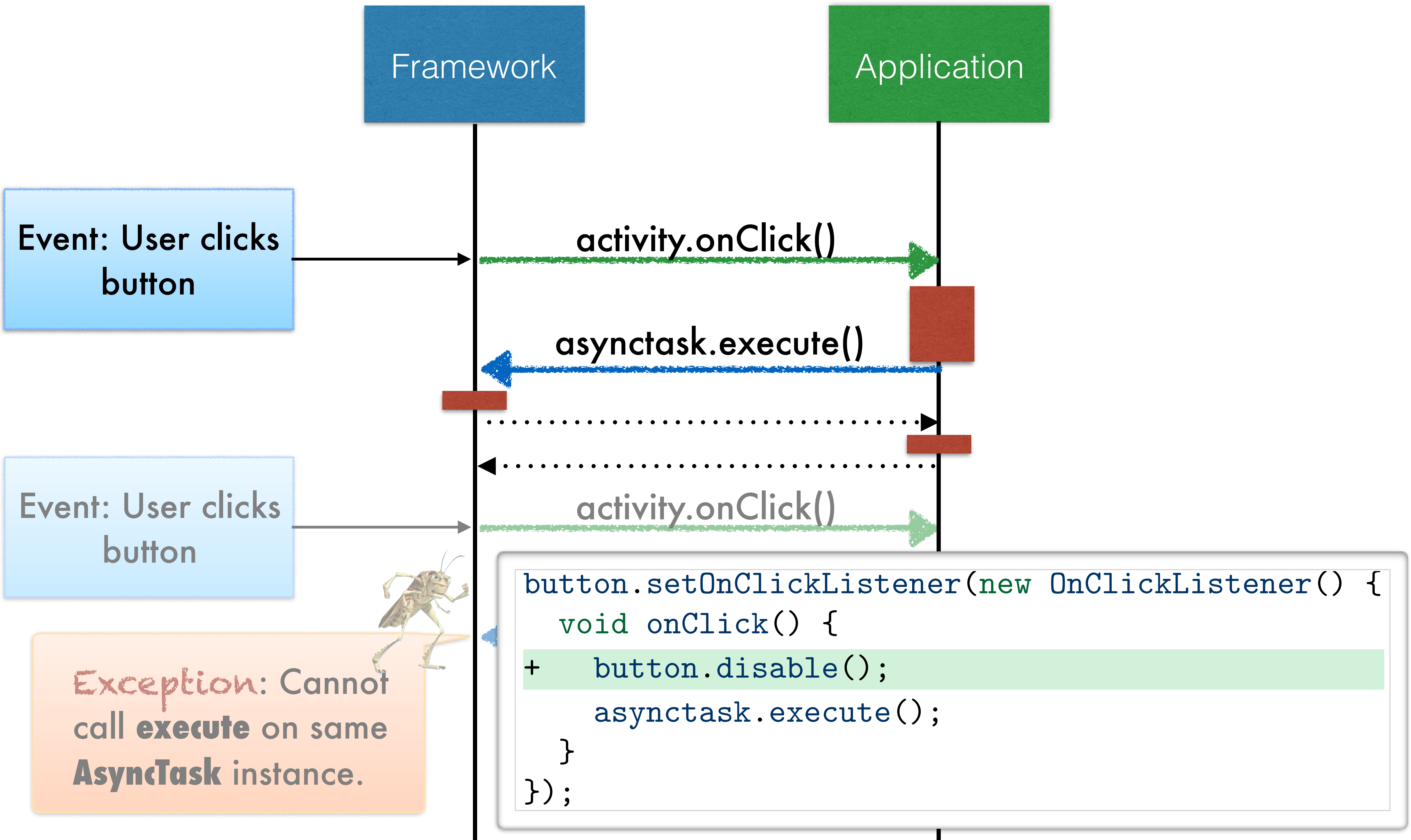


Android is an event-driven system




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

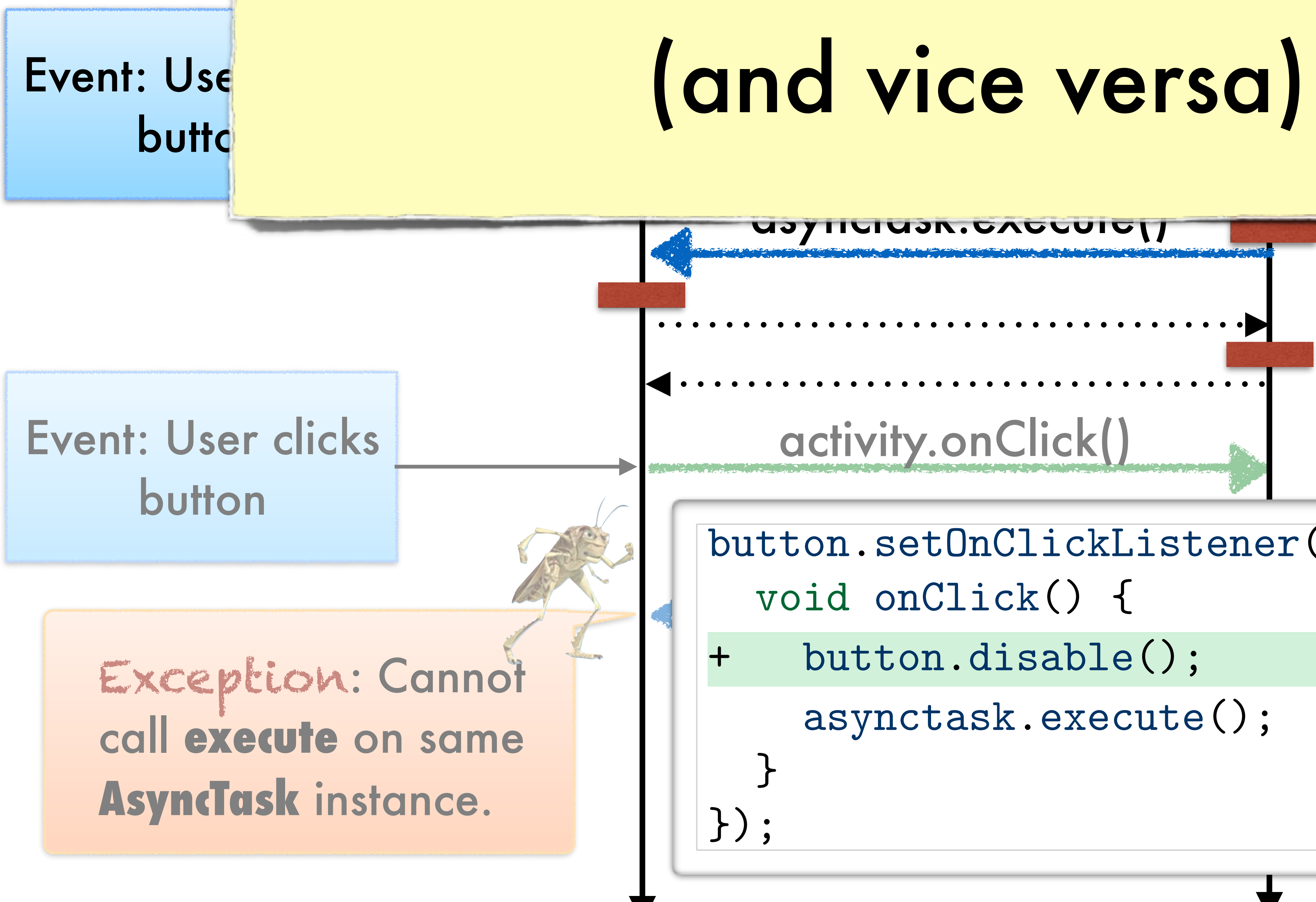
Android is an event-driven system




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

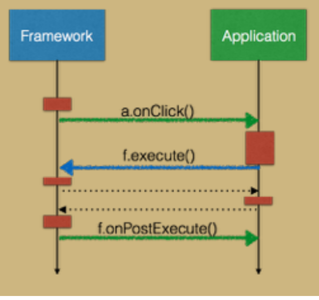
Android is an event-driven system

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

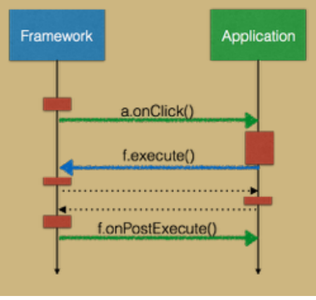



```
button.setOnClickListener(new OnClickListener() {
  void onClick() {
+   button.disable();
    asynctask.execute();
  }
});
```

Contributions: Lifestate

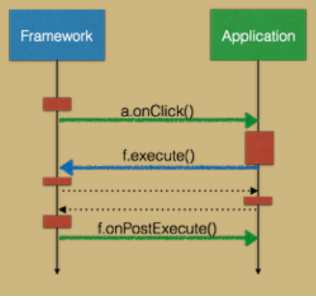


Contributions: Lifestate



λlife: A (concrete) model of event-driven systems capturing how callins and callbacks affect each other

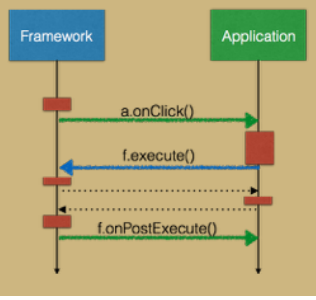
Contributions: Lifestate



λlife: 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

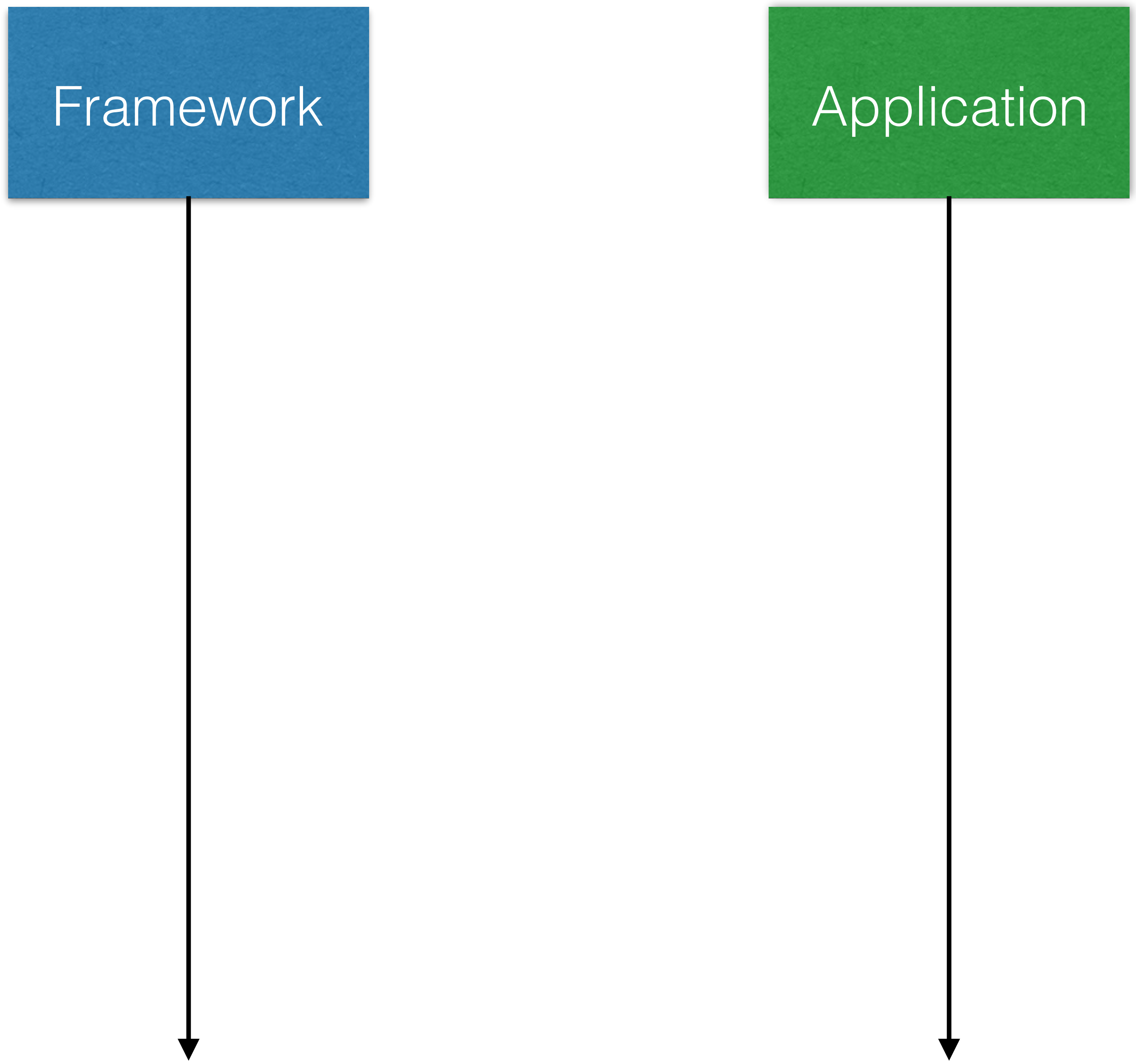


λlife: A (concrete) model of event-driven systems capturing how callins and callbacks affect each other

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

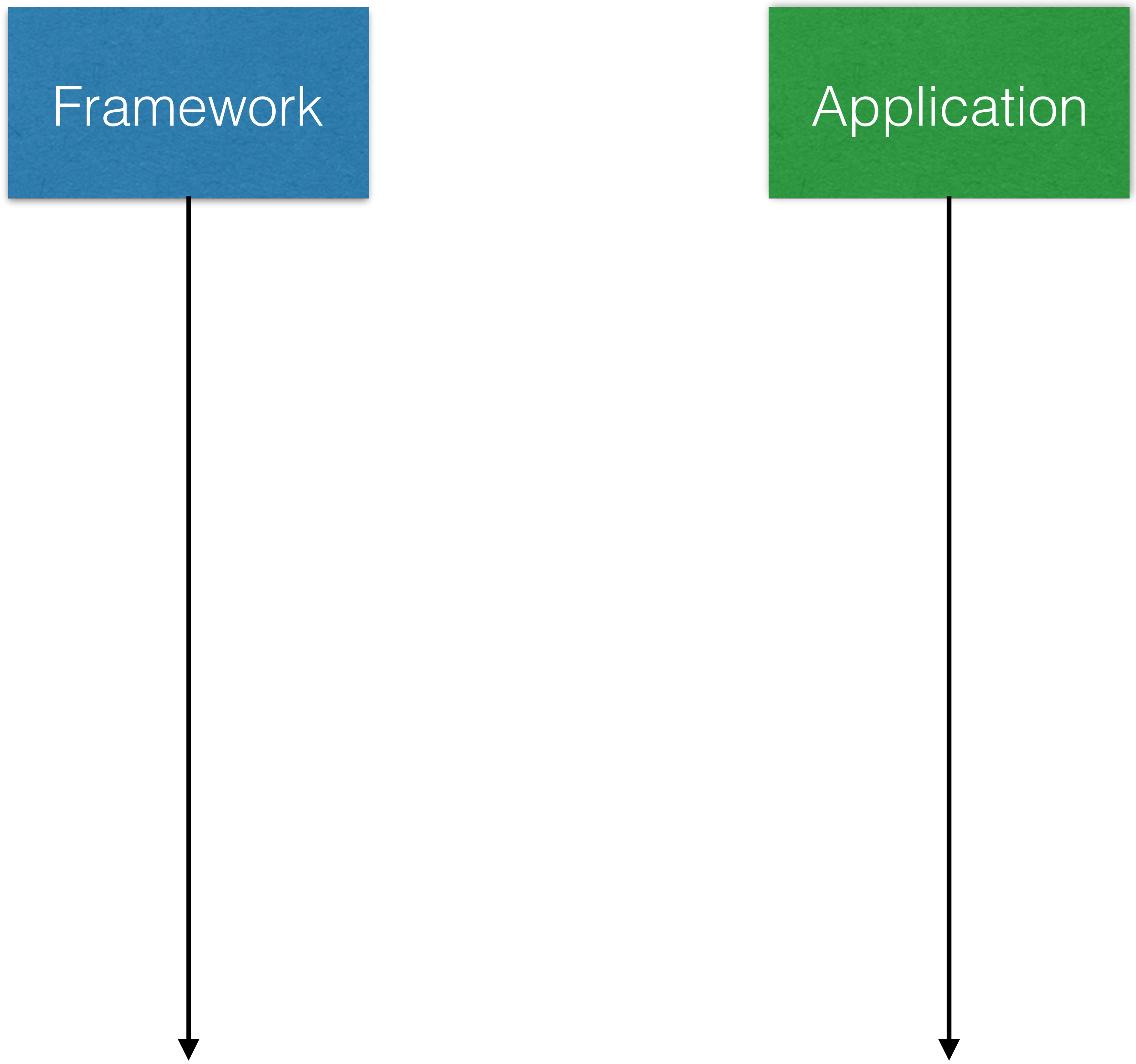
DroidLife: Mining lifestate specifications and verifying the absence of lifestate “races”

Enabled callbacks and allowed callins

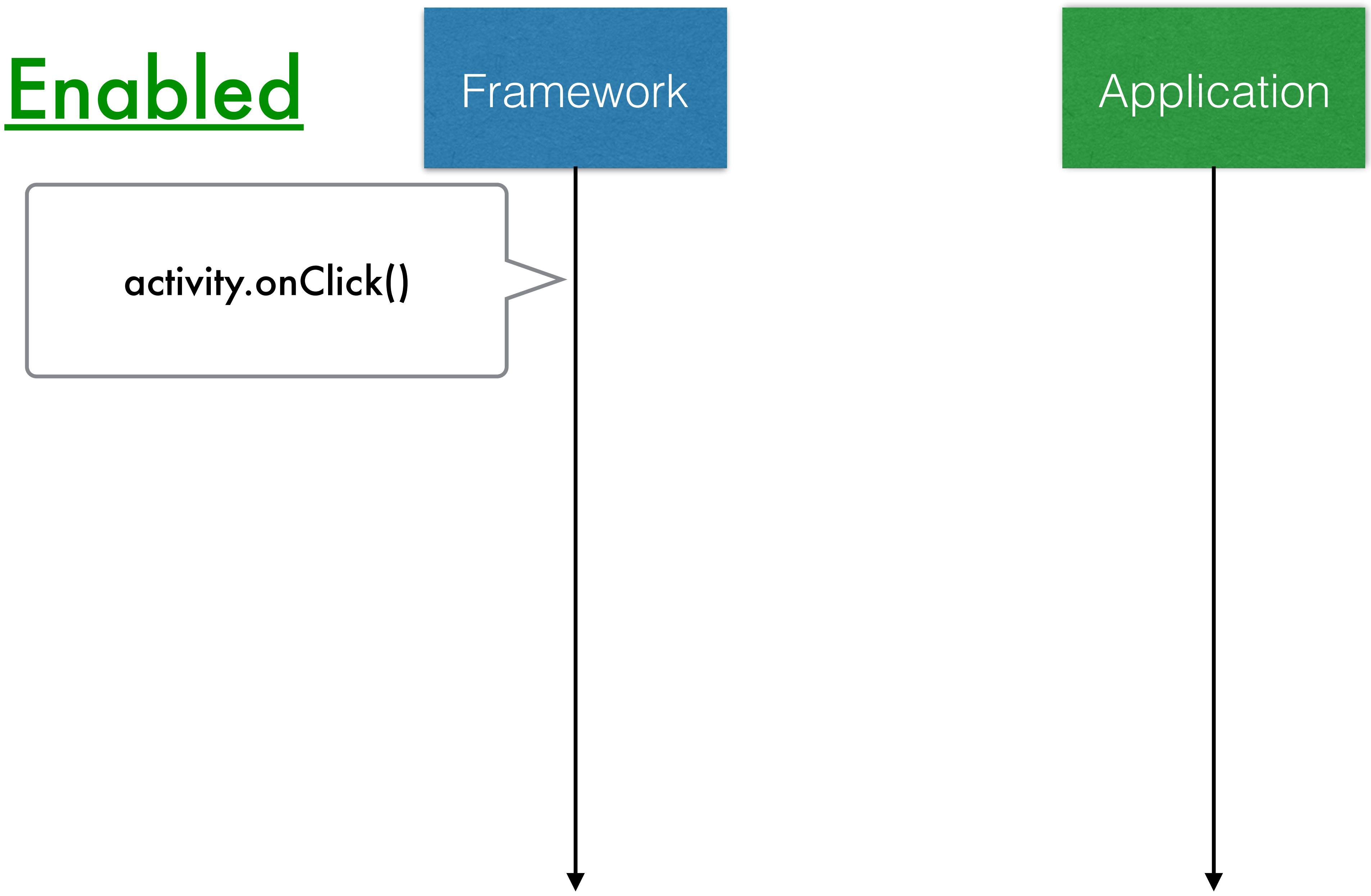


Enabled callbacks and allowed callins

Enabled

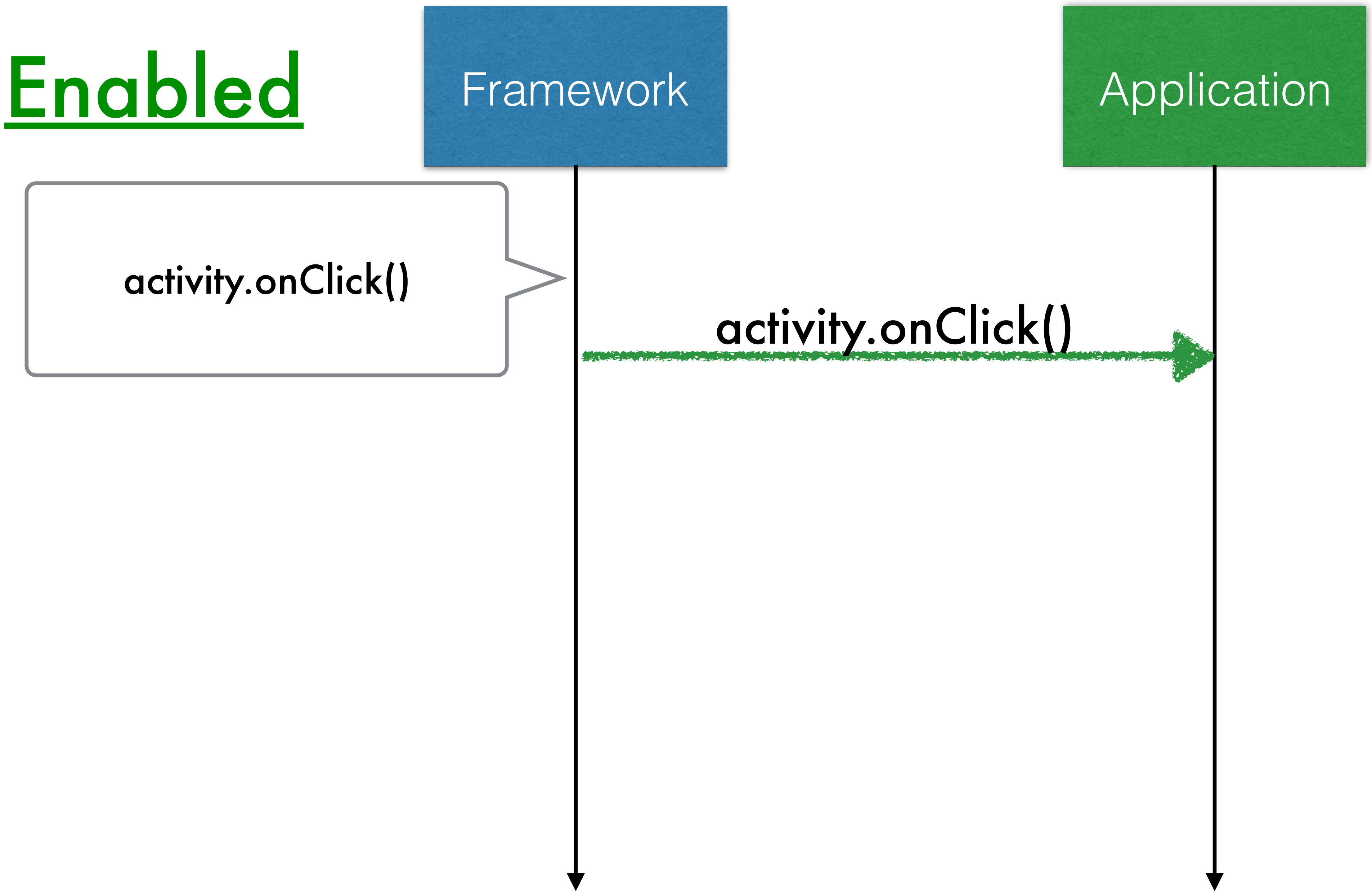


Enabled callbacks and allowed callins

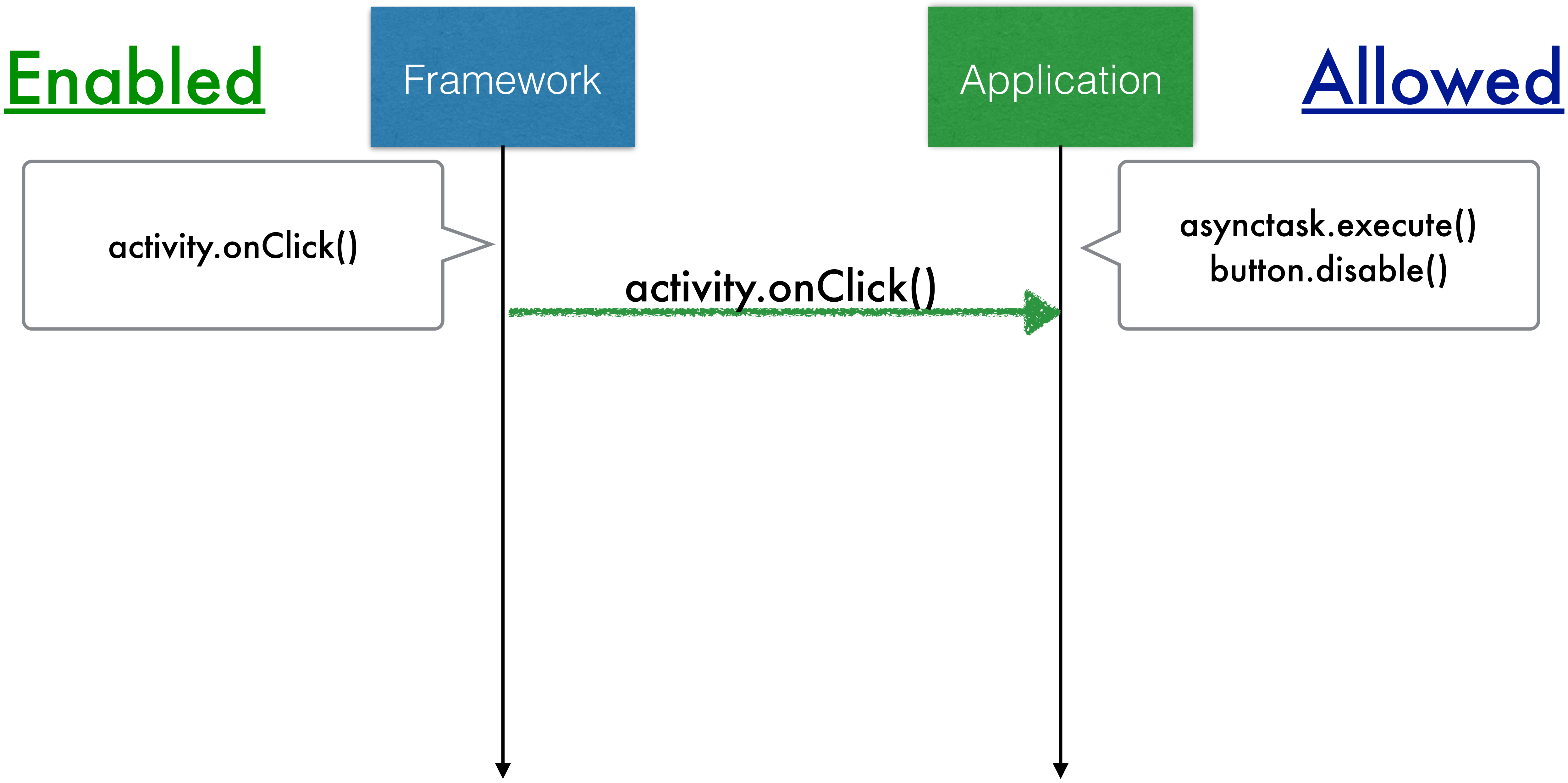


Enabled callbacks and allowed callins

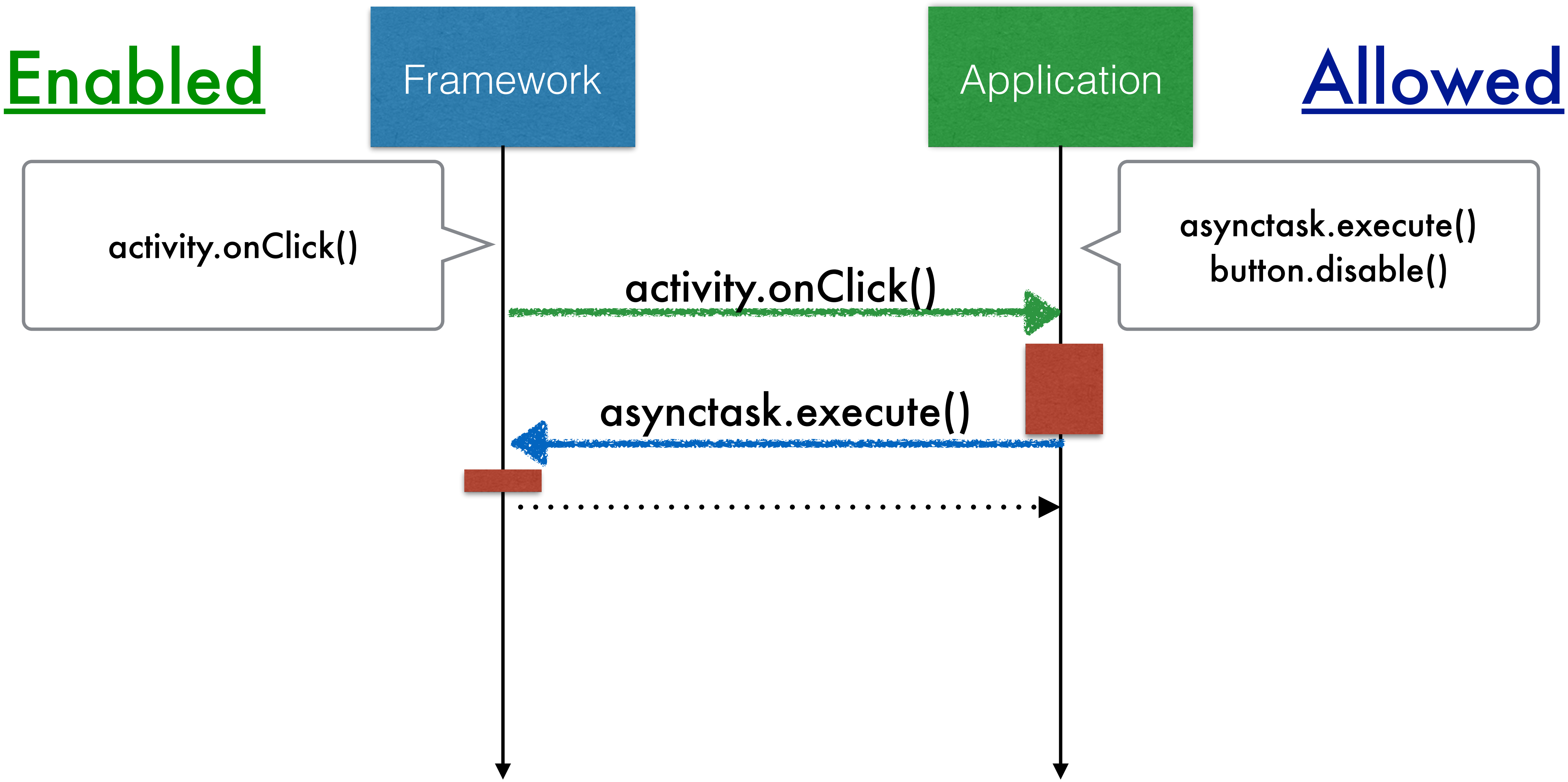
Enabled



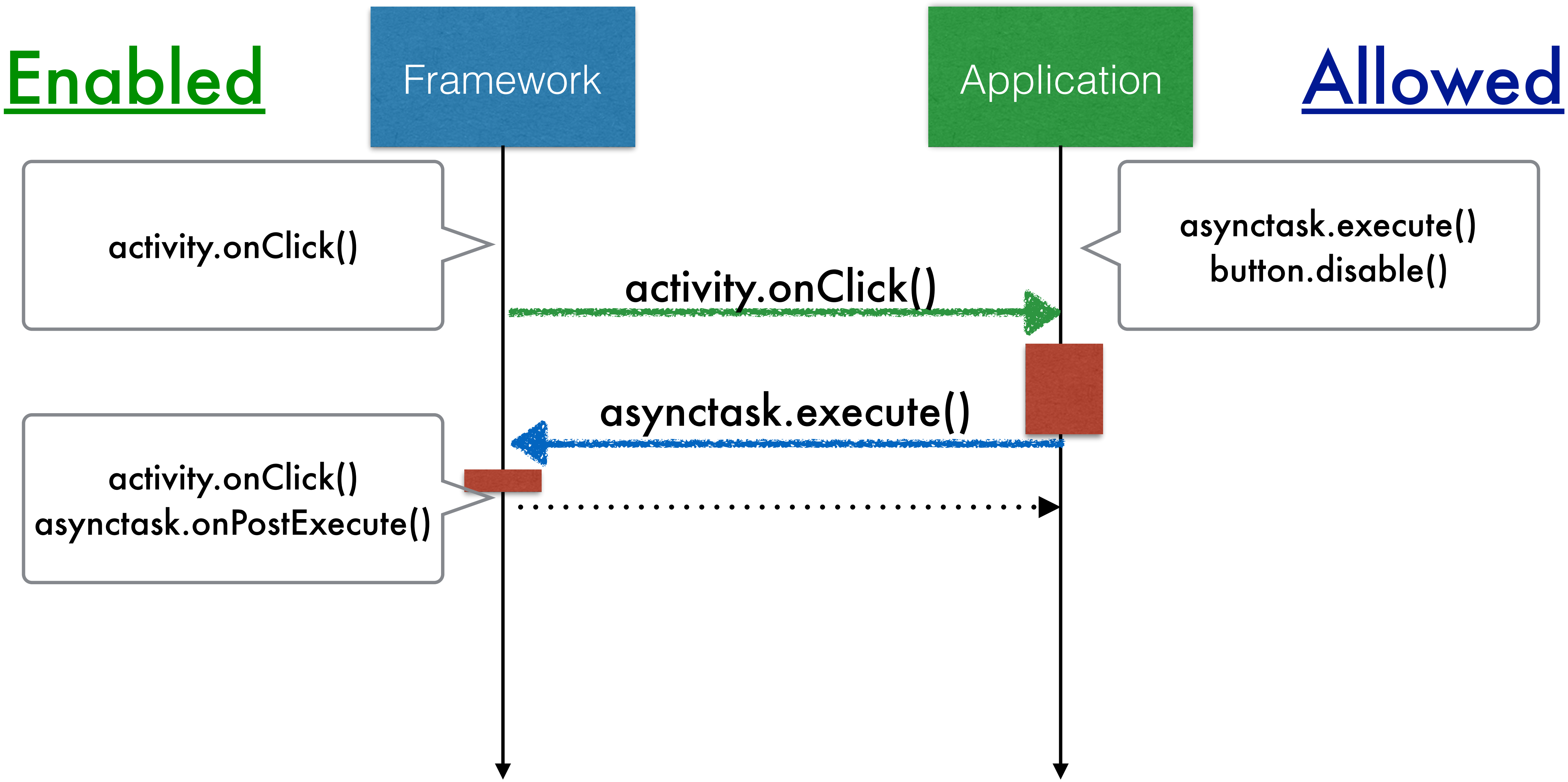
Enabled callbacks and allowed callins



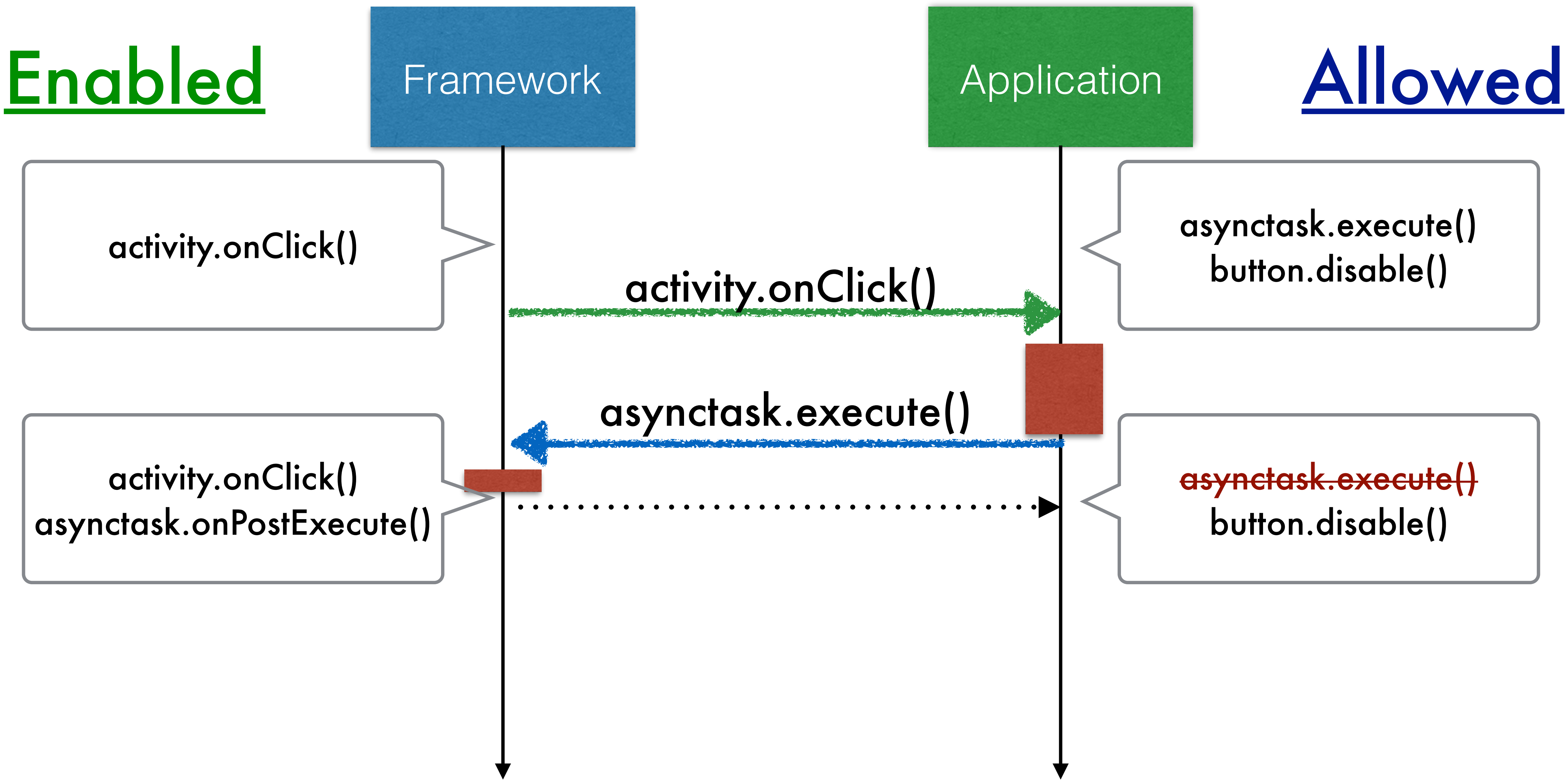
Enabled callbacks and allowed callins



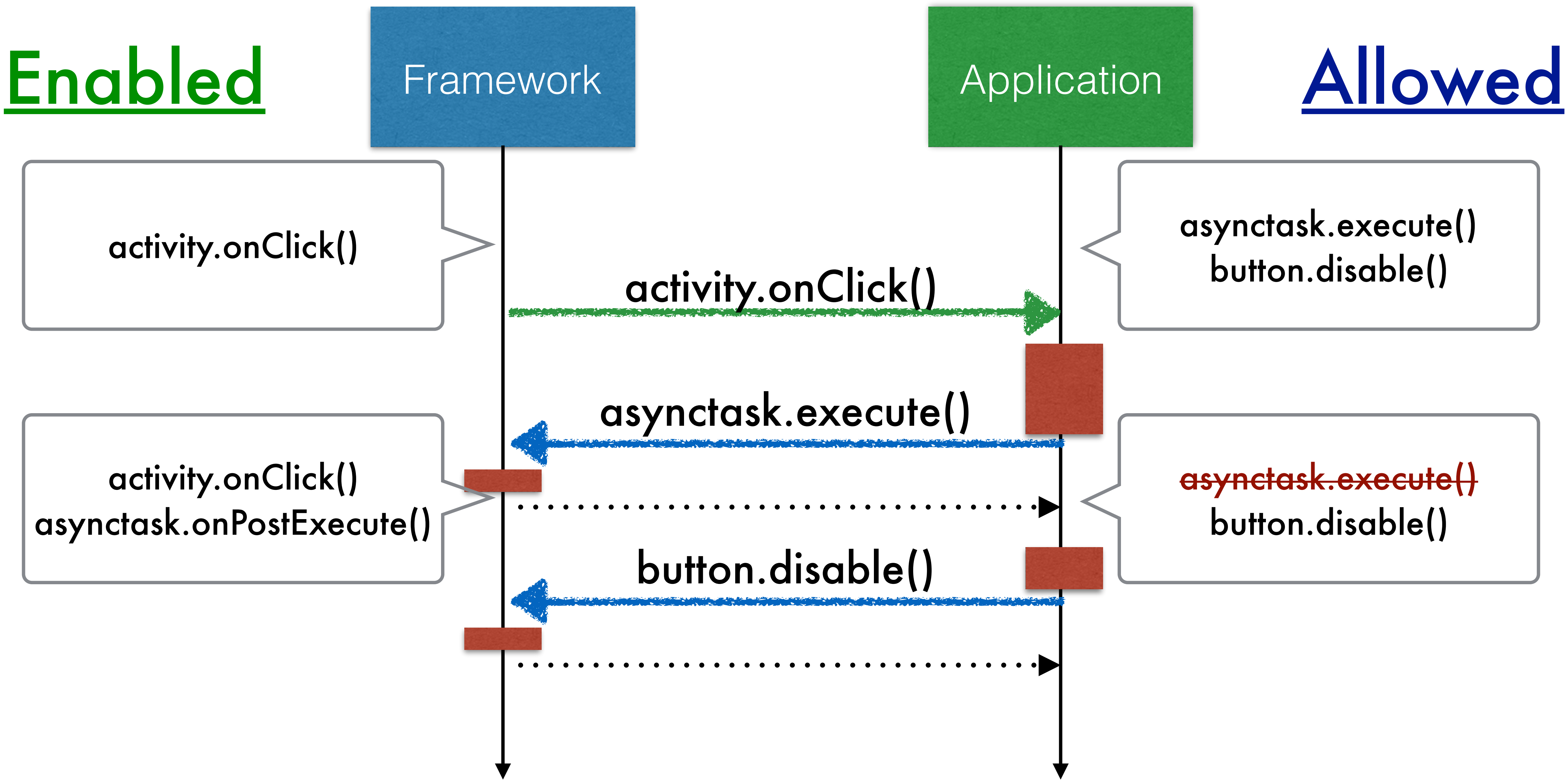
Enabled callbacks and allowed callins



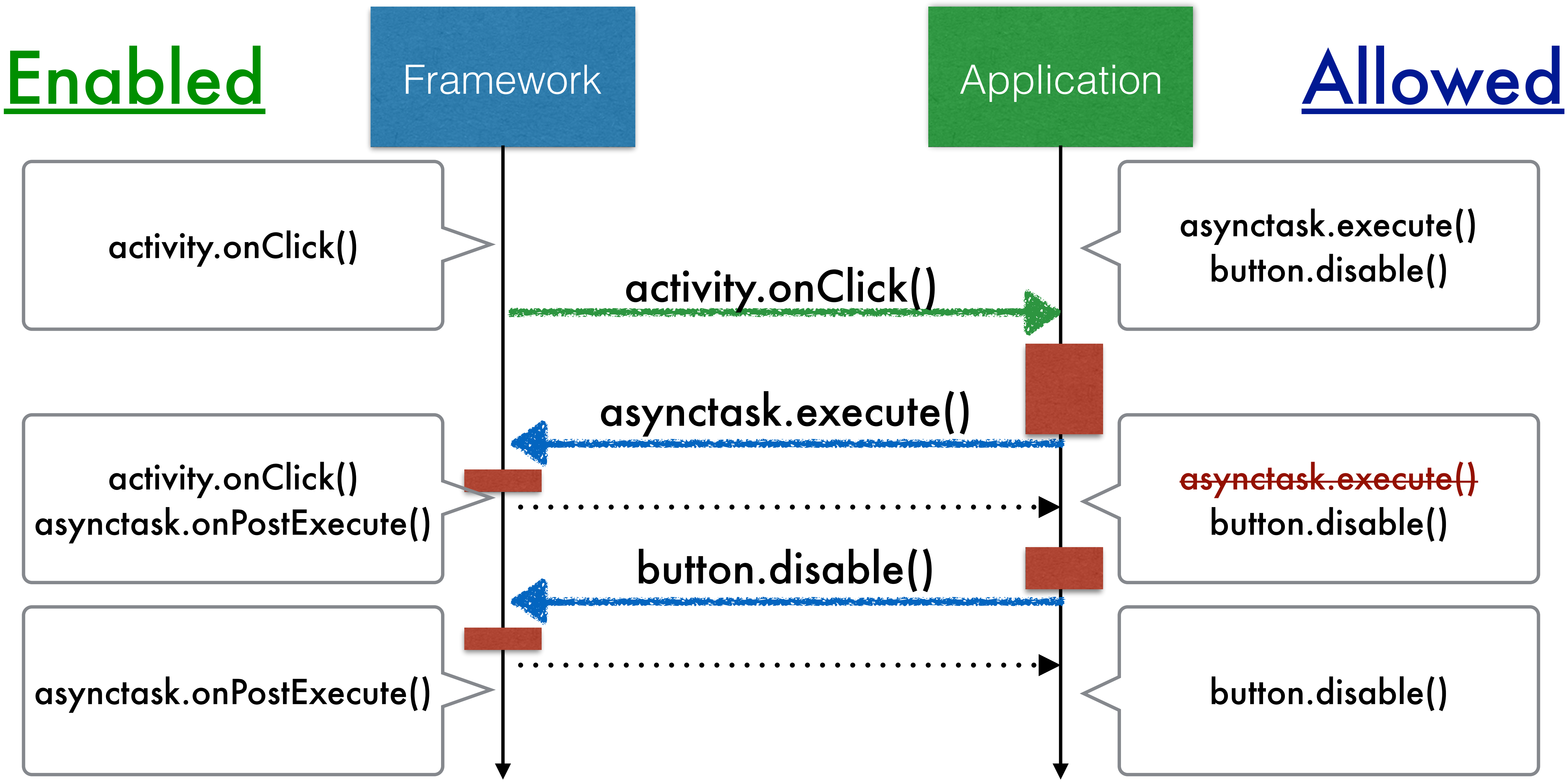
Enabled callbacks and allowed callins



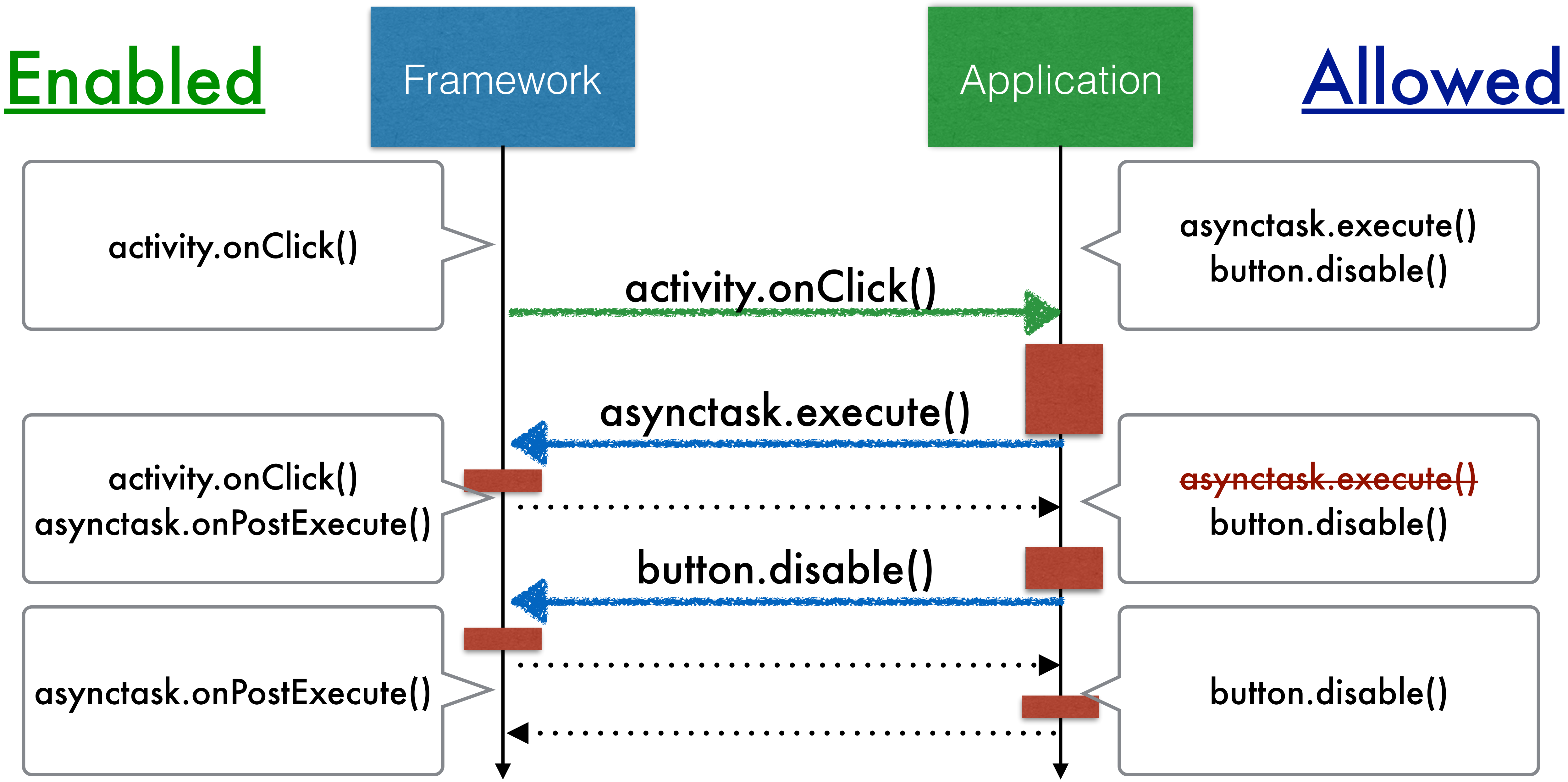
Enabled callbacks and allowed callins



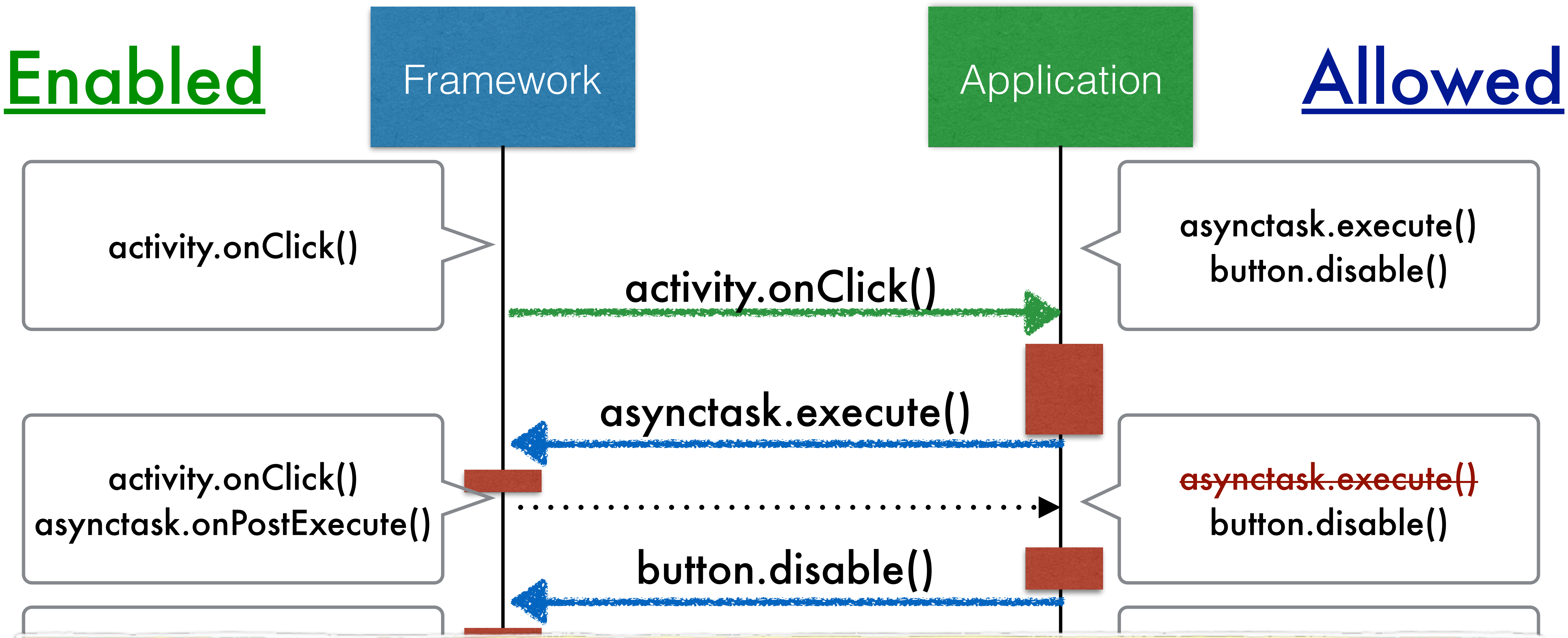
Enabled callbacks and allowed callins



Enabled callbacks and allowed callins



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

enable

$message_1$ **→cb** $message_2$

Lifestate rules specify enabledness and allowedness effects

A suspended callback or callin invokation (thunk)

enable

$message_1$ **→cb** $message_2$

Lifestate rules specify enabledness and allowedness effects

A suspended callback or callin invocation (thunk)

enable

$$message_1 \;\; \boldsymbol{\rightarrow}\mathbf{cb} \;\; message_2$$

Message causes a callback to be added to the enabled set

Lifestate rules specify enabledness and allowedness effects

A suspended callback or callin invokation (thunk)

enable

$message_1$ **→cb** $message_2$

Message causes a callback to be added to the enabled set

disable

$message_1$ **↛cb** $message_2$

Lifestate rules specify enabledness and allowedness effects

A suspended callback or callin invokation (thunk)

enable

$$message_1 \ \rightarrow\mathbf{cb} \ message_2$$

Message causes a callback to be added to the enabled set

disable

$$message_1 \ \nrightarrow\mathbf{cb} \ message_2$$

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

$$message_1 \;\mathbf{\rightarrow cb}\; message_2$$

Message causes a callback to be added to the enabled set

allow

$$message_1 \;\mathbf{\rightarrow ci}\; message_2$$

disable

$$message_1 \;\mathbf{\nrightarrow cb}\; message_2$$

Message causes a callback to be removed to the enabled set

Lifestate rules specify enabledness and allowedness effects

A suspended callback or callin invokation (thunk)

enable

$message_1 \rightarrow_{\mathbf{cb}} message_2$

Message causes a callback to be added to the enabled set

allow

$message_1 \rightarrow_{\mathbf{ci}} message_2$

Message causes a callin to be added to the allowed set

disable

$message_1 \nrightarrow_{\mathbf{cb}} message_2$

Message causes a callback to be removed to the enabled set

Lifestate rules specify enabledness and allowedness effects

A suspended callback or callin invokation (thunk)

enable

$$message_1 \ \mathbf{\rightarrow cb} \ message_2$$

Message causes a callback to be added to the enabled set

allow

$$message_1 \ \mathbf{\rightarrow ci} \ message_2$$

Message causes a callin to be added to the allowed set

disable

$$message_1 \ \mathbf{\nrightarrow cb} \ message_2$$

Message causes a callback to be removed to the enabled set

disallow

$$message_1 \ \mathbf{\nrightarrow ci} \ message_2$$

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

allow

$message_1$ $\rightarrow$**ci** $message_2$

Message causes a callin to be added to the allowed set

disable

$message_1$ $\nrightarrow$**cb** $message_2$

Message causes a callback to be removed to the enabled 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 callba[ck] ... callin invokation (th[...]

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

enable

$message_1$ →`cb` $message_2$

Message causes a callback to be added to the enabled set

allow

$message_1$ →`ci` $message_2$

Message causes a callin to be added to the allowed set

disable

$message_1$ ↛`cb` $message_2$

Message causes a callback to be removed to the enabled set

disallow

$message_1$ ↛`ci` $message_2$

Message causes a callin to be removed from the allowed set

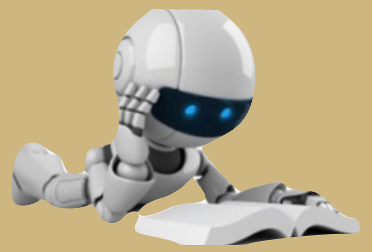
Need: Mining lifestate specifications

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

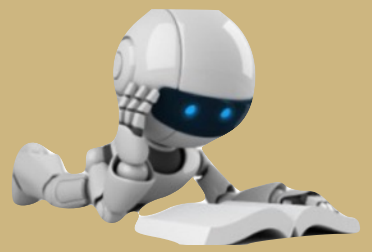


These specifications often don’t exist in the documentation.

> 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 `Params`, `Progress` and `Result`, and 4 steps, called `onPreExecute`, `doInBackground`, `onProgressUpdate` and `onPostExecute`.

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 **AsyncTask** documentation does not fully document the behavior

The Android Framework is Huge

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

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 **AsyncTask** documentation does not fully document the behavior

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

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 **AsyncTask** documentation does not fully document the behavior

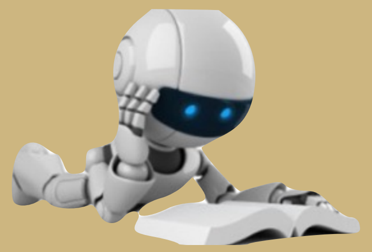
The Android Framework is Huge

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

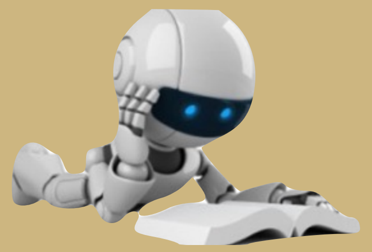
Writing specifications by hand is error prone

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

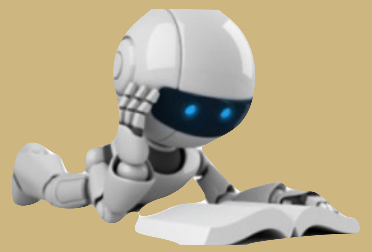


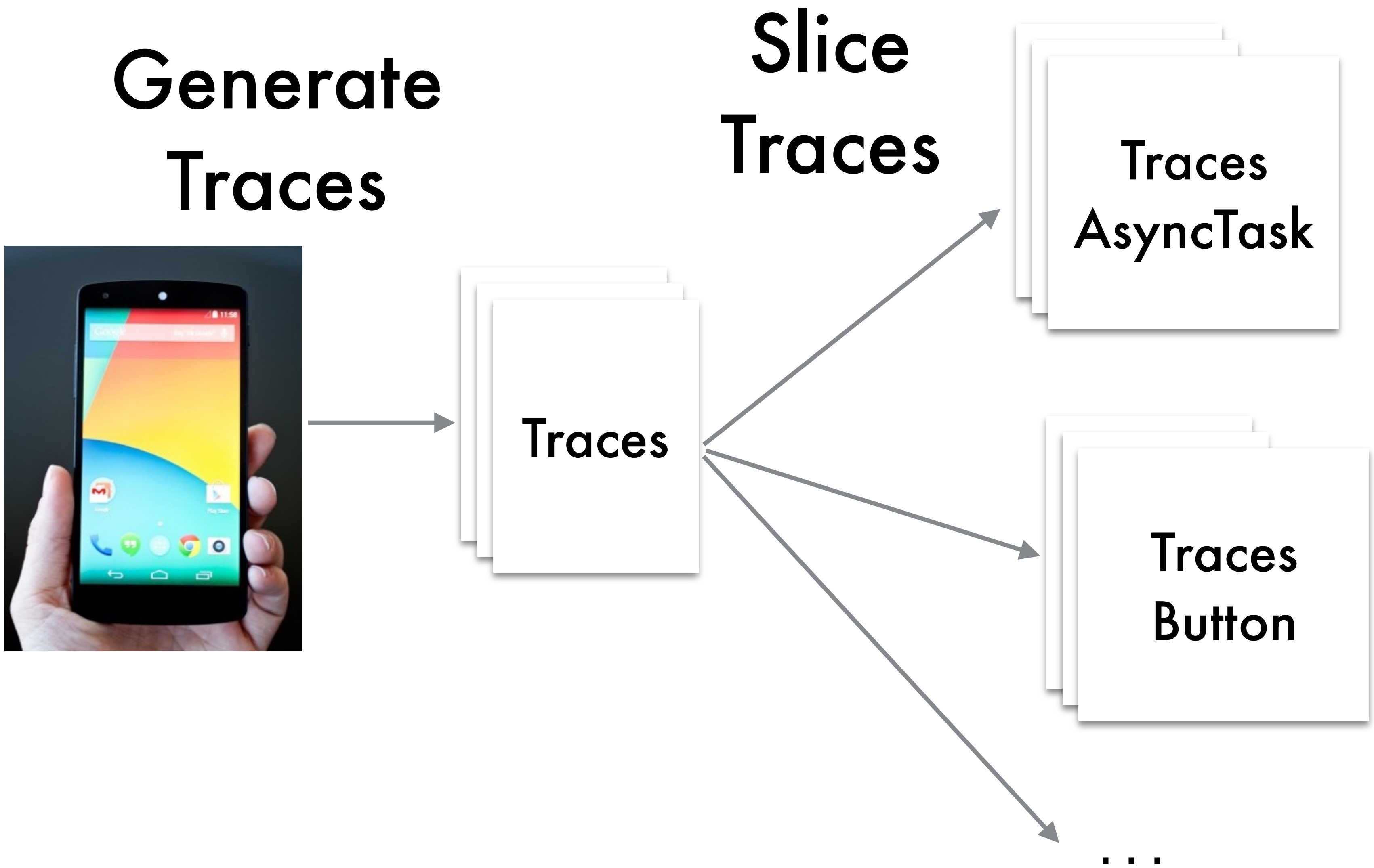
Learn lifestate rules
that explain observed traces



Generate Traces

Learn lifestate rules that explain observed traces





Learn lifestate rules that explain observed traces

Learn Rules (unsupervised)

Generate Traces

Slice Traces

Traces

Traces AsyncTask

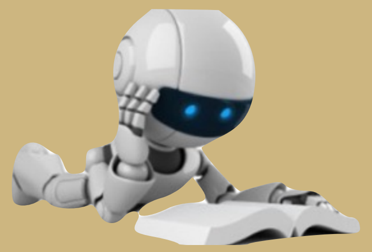
Traces Button

...

AsyncTask Specification

Button Specification

Learn lifestate rules that explain observed traces



Generate Traces

Traces

Slice Traces

Traces AsyncTask

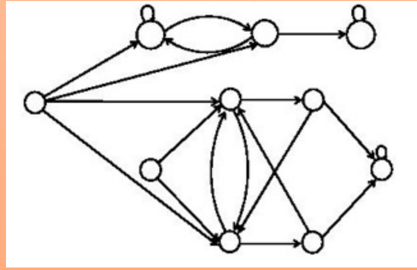
Traces Button

...

Learn Rules (unsupervised)

AsyncTask Specification

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



Learn lifestate rules
that explain observed traces

Learn Rules
(unsupervised)

Generate
Traces

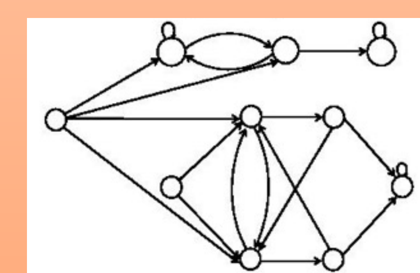
Slice
Traces

Traces
AsyncTask

AsyncTask
Specification

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.



. . .

Learn lifestate rules that explain observed traces

Generate Traces

Slice Traces

Learn Rules (unsupervised)

Traces AsyncTask

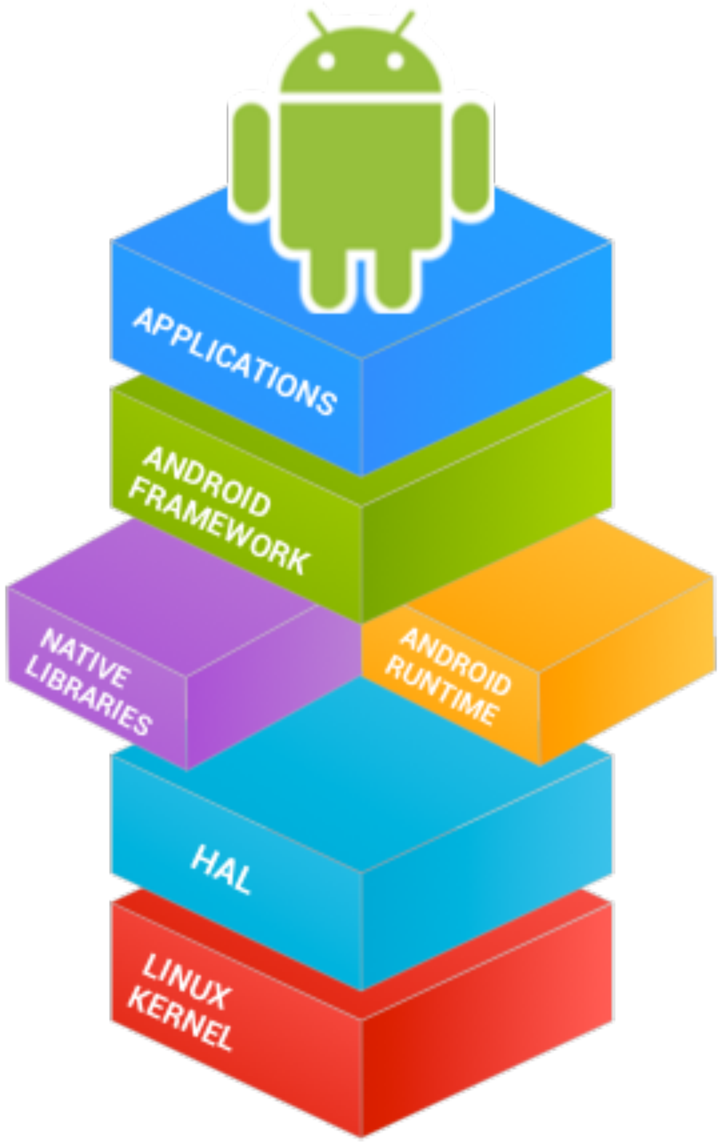
AsyncTask Specification

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

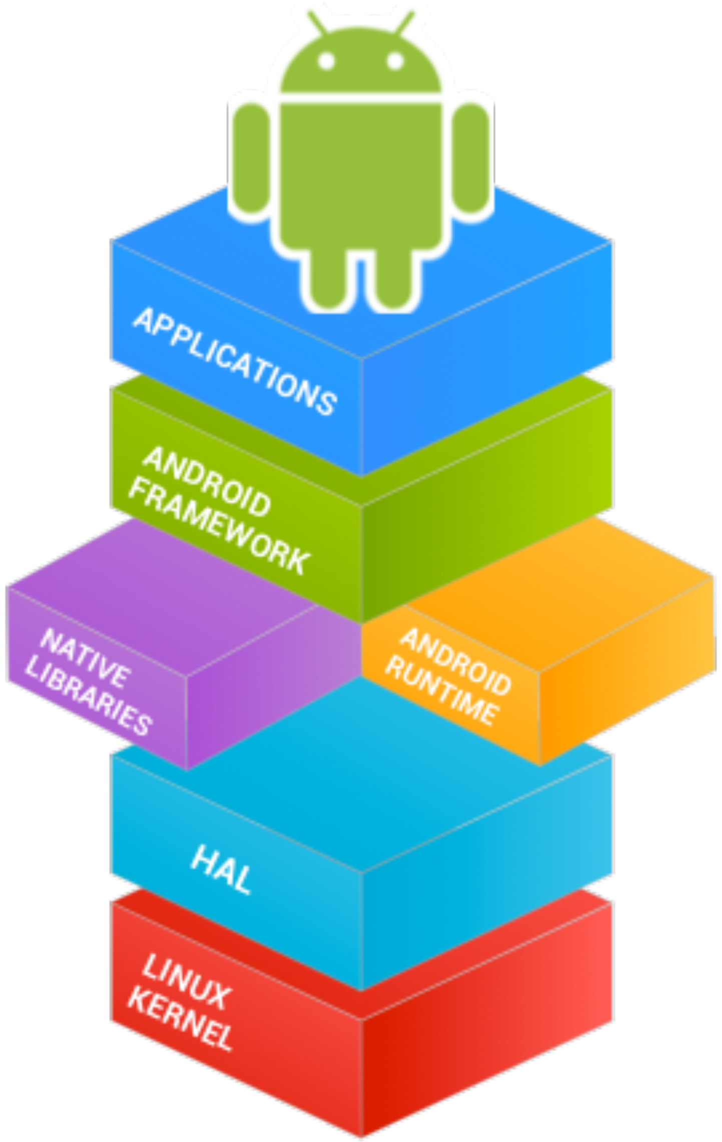
Learn automata models (hidden Markov models [HMMs], probabilistic finite state automata [PFSAs]). Abstract automata into lifestate rules.

A portfolio of approaches to learn candidate specifications

Evaluating lifestate specification mining

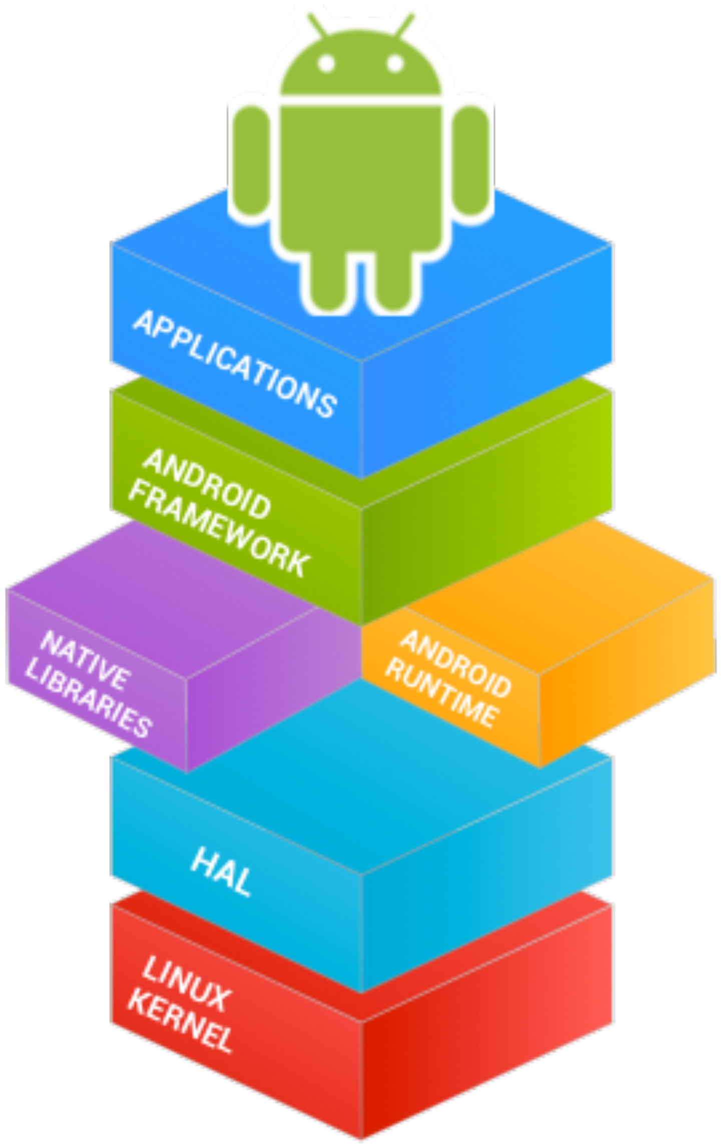


Evaluating lifestate specification mining



Do we learn rules that correspond to actual Android behavior?

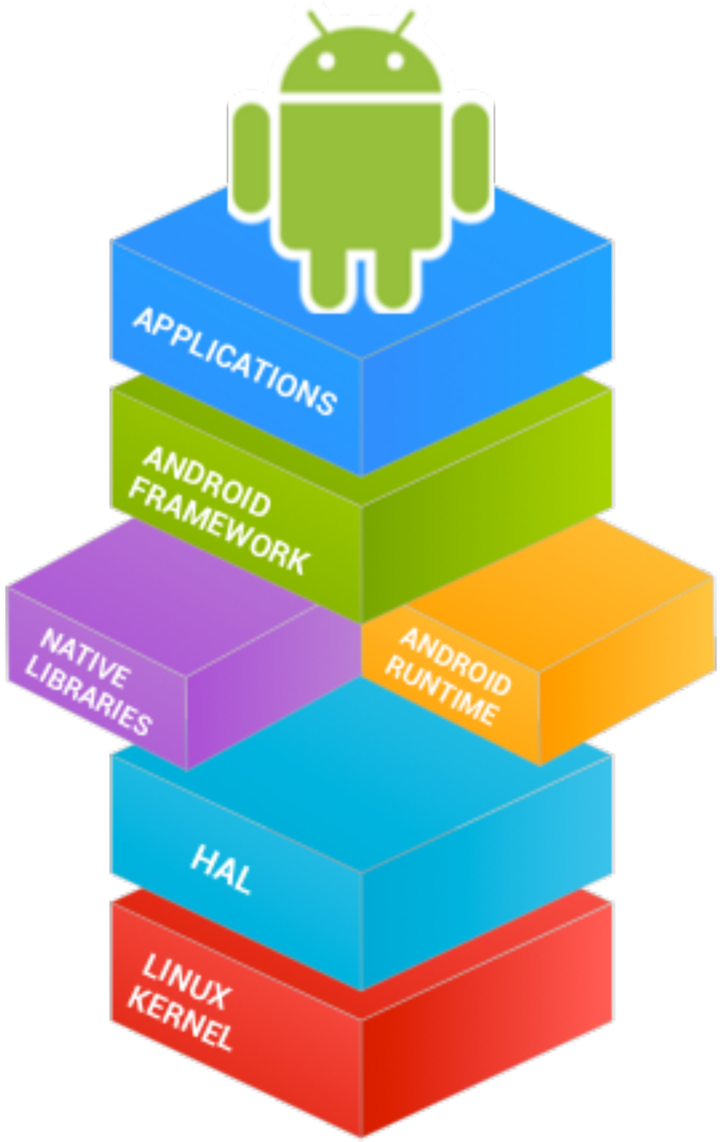
Evaluating lifestate specification mining



Do we learn rules that correspond to actual Android behavior?

4 Android framework classes

Evaluating lifestate specification mining

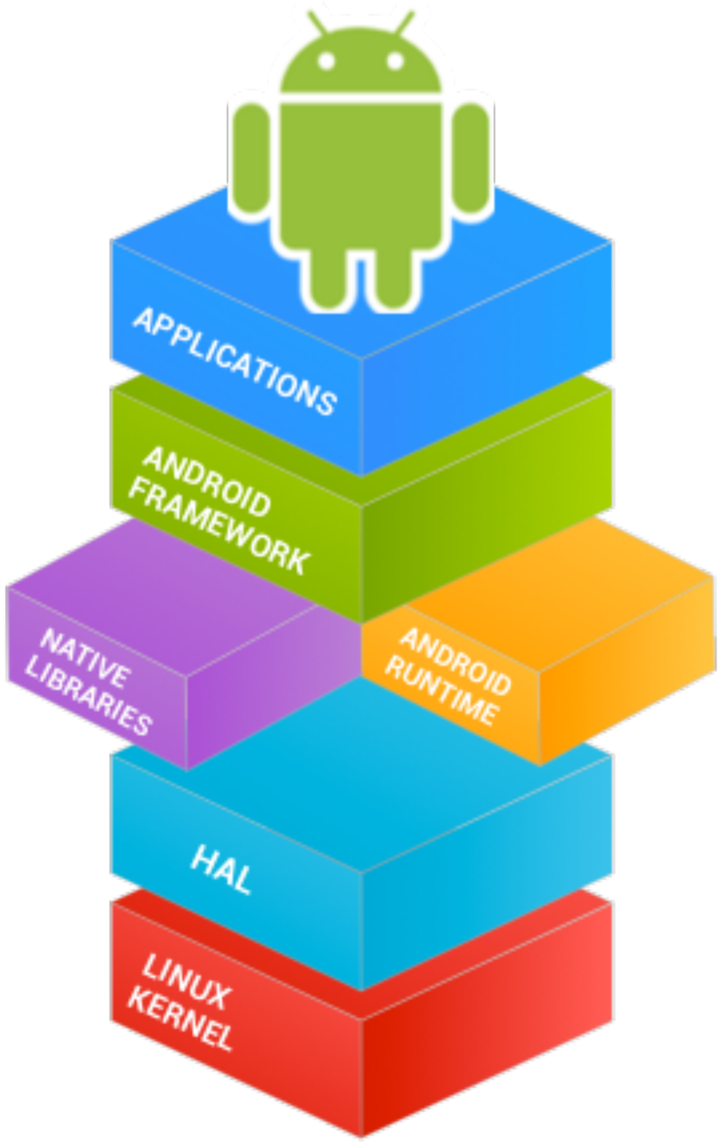


Do we learn rules that correspond to actual Android behavior?

4 Android framework classes

7326 possible specification rules

Evaluating lifestate specification mining



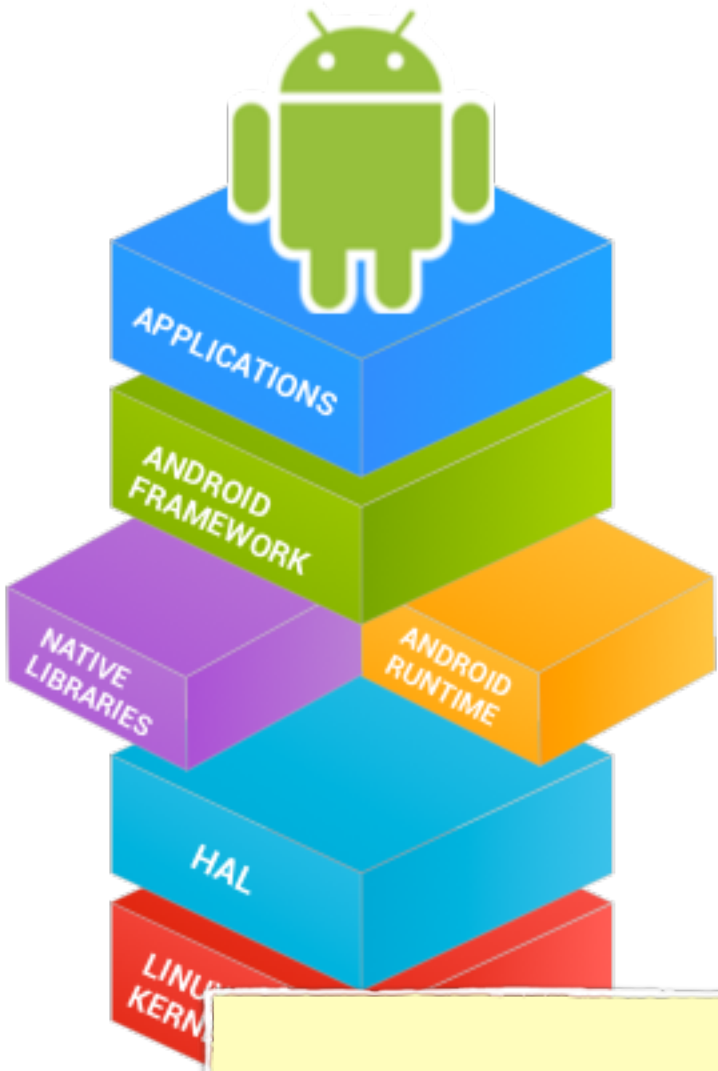
Do we learn rules that correspond to actual Android behavior?

4 Android framework classes

7326 possible specification rules

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

Evaluating lifestate specification mining



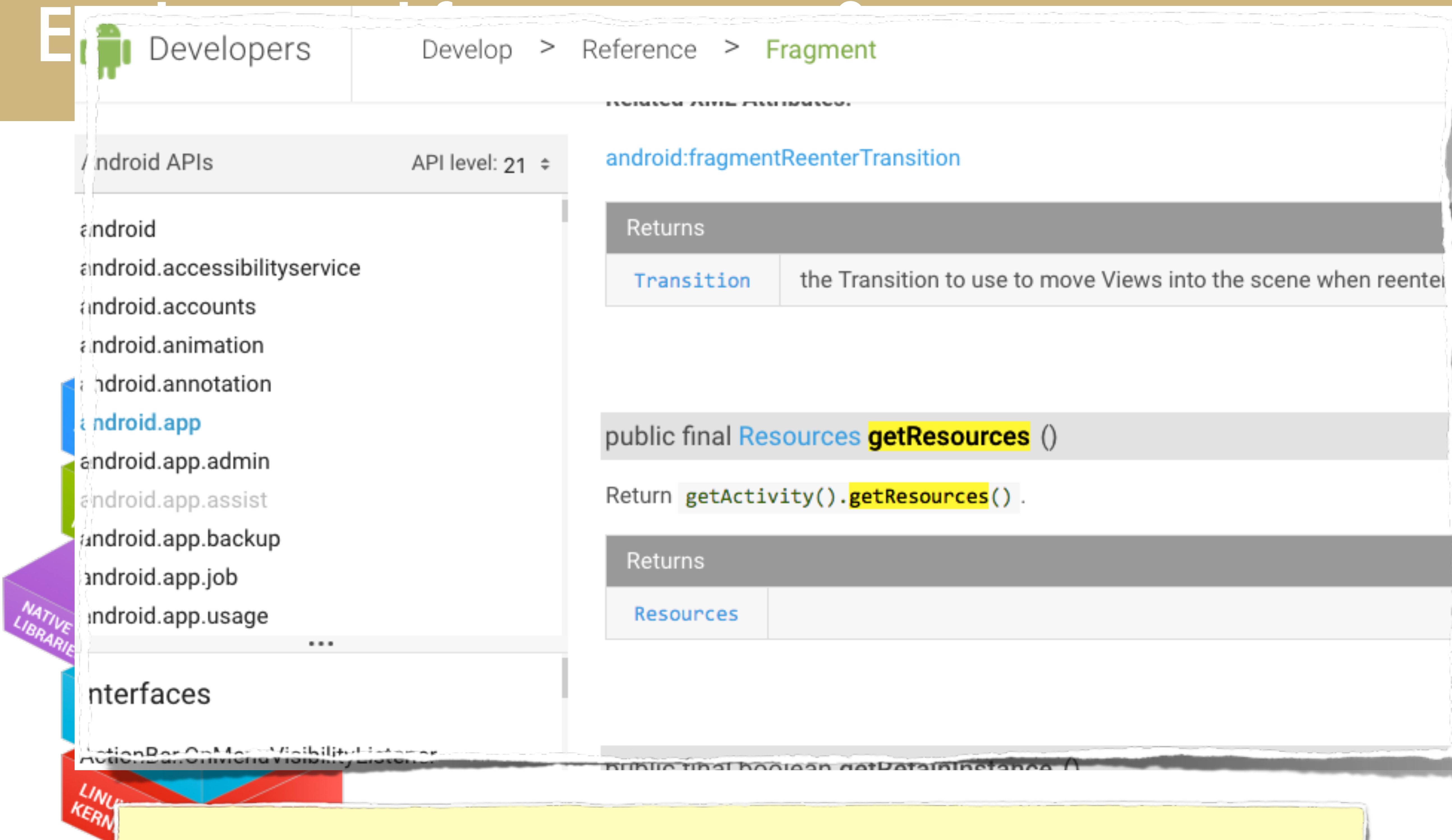
Do we learn rules that correspond to actual Android behavior?

4 Android framework classes

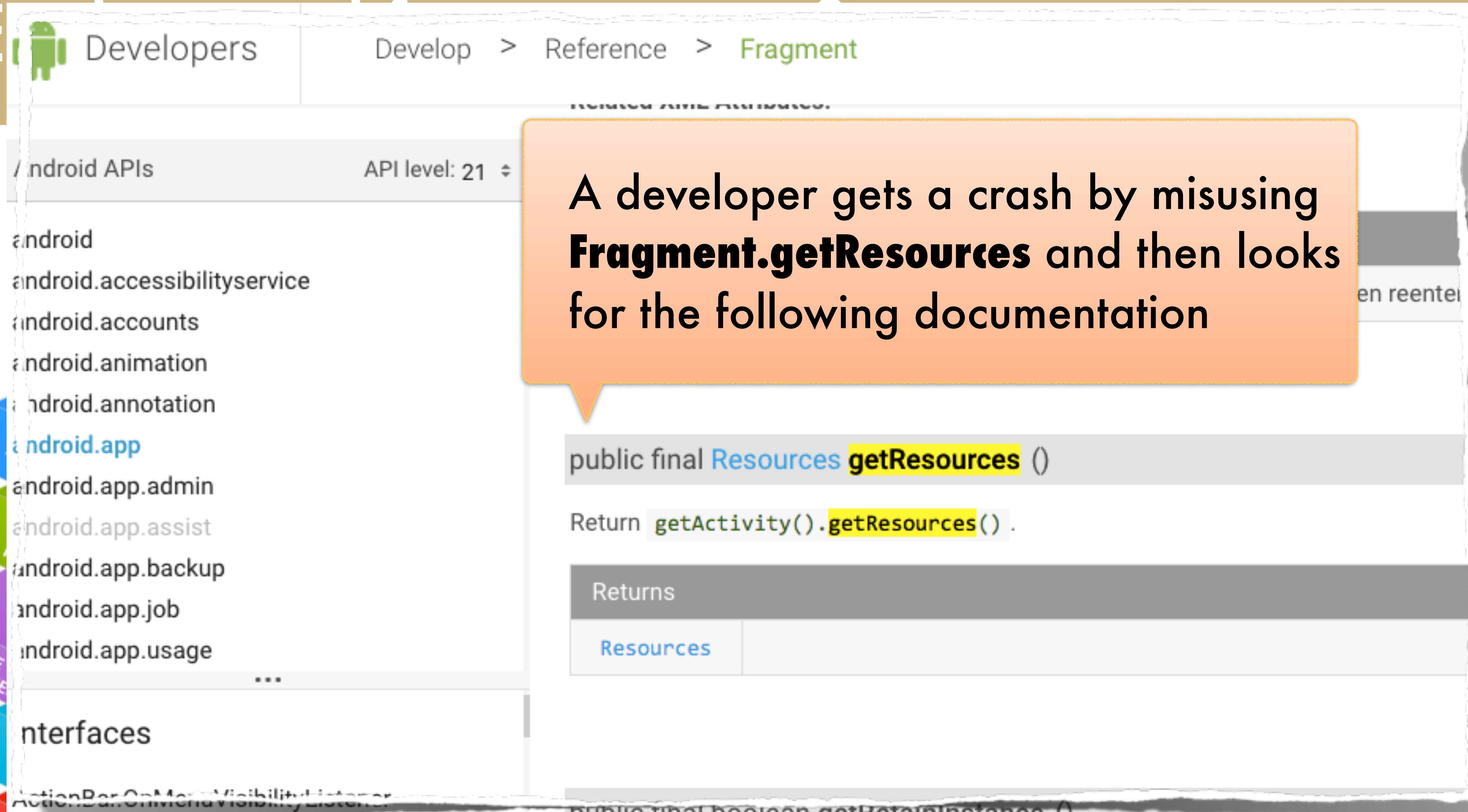
7326 possible specification rules

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

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



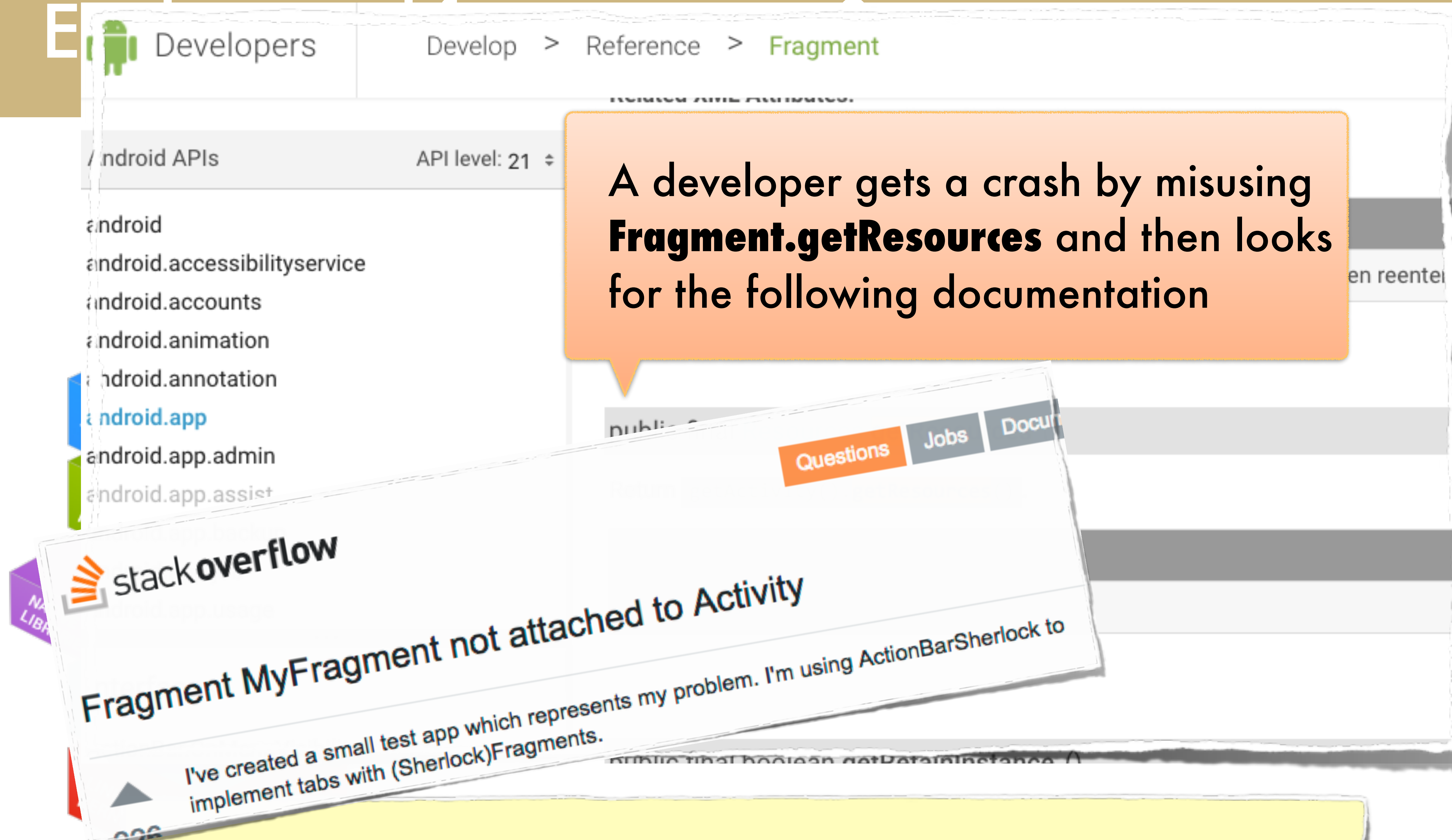
Found actual rules in under-constrained search space. Discovered undocumented rules.



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

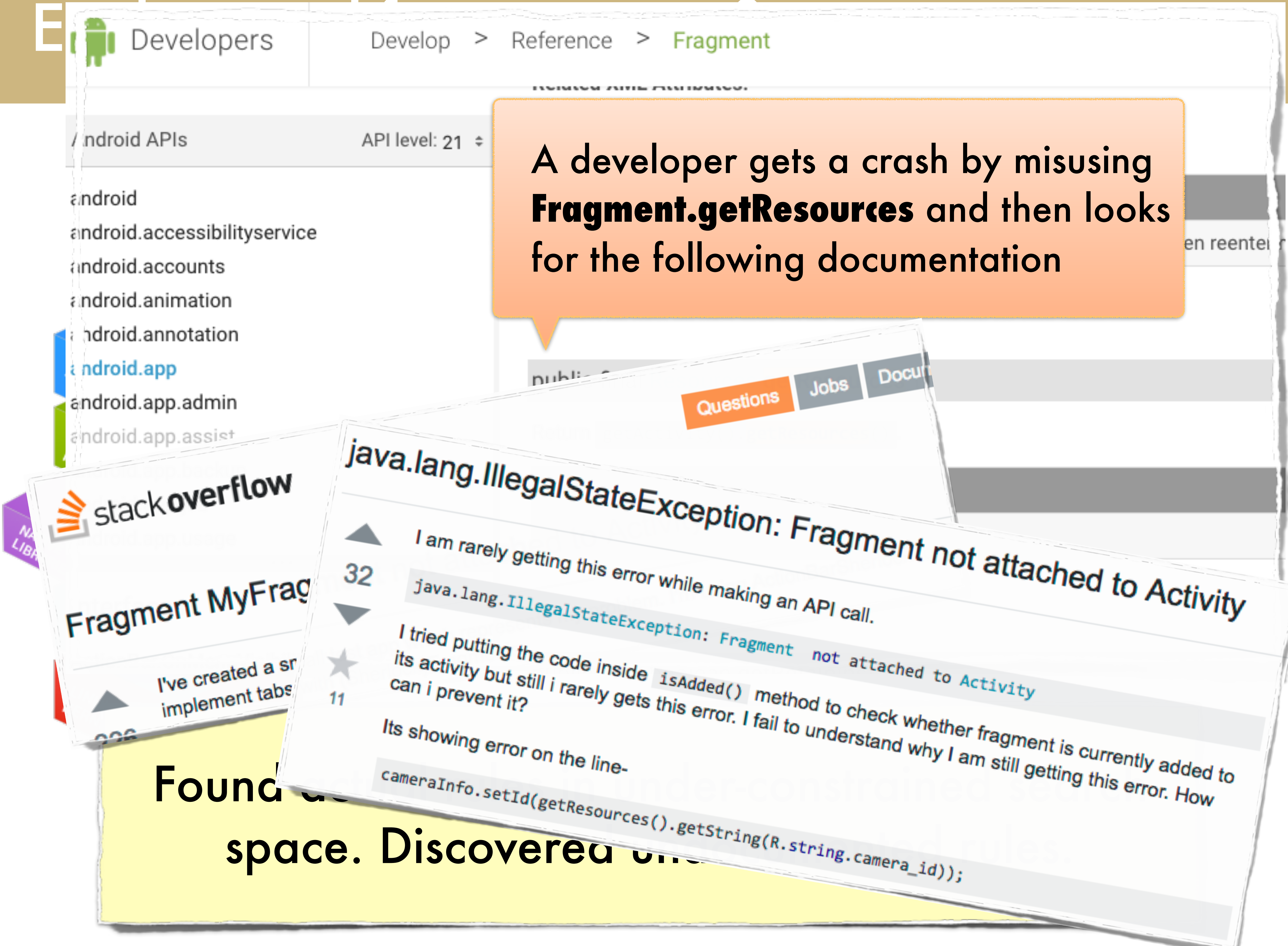
Found actual rules in under-constrained search space. Discovered undocumented rules.

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



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

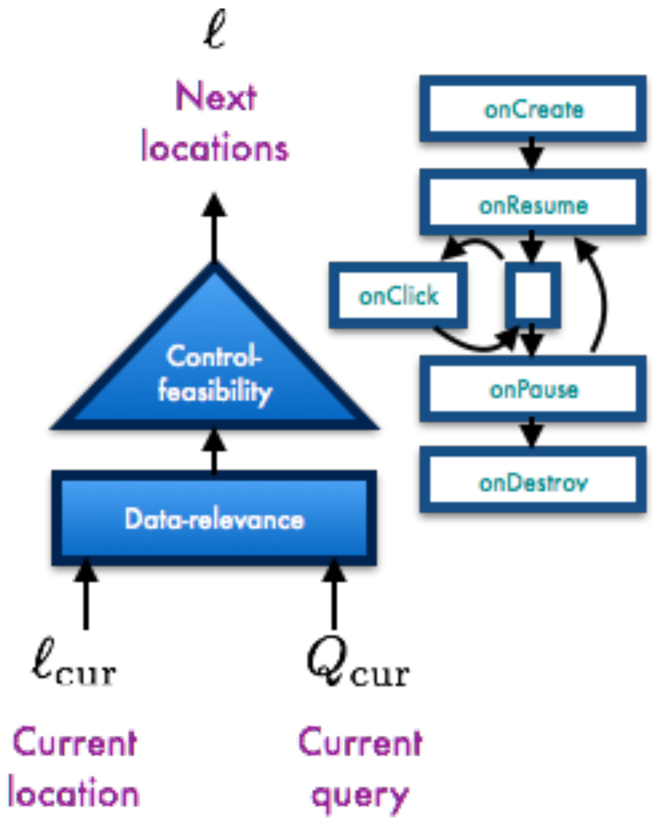


Found

space. Discovered

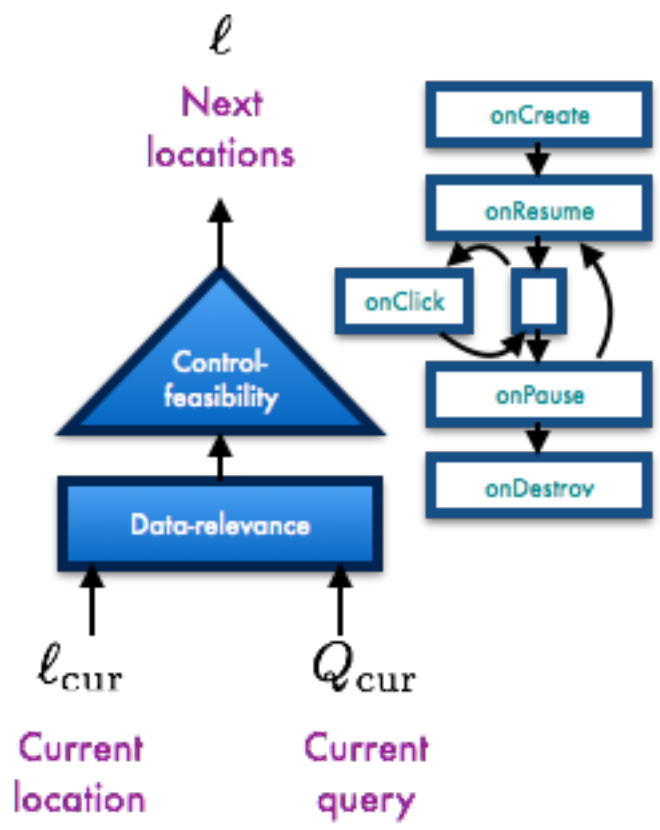
Conclusion

Conclusion

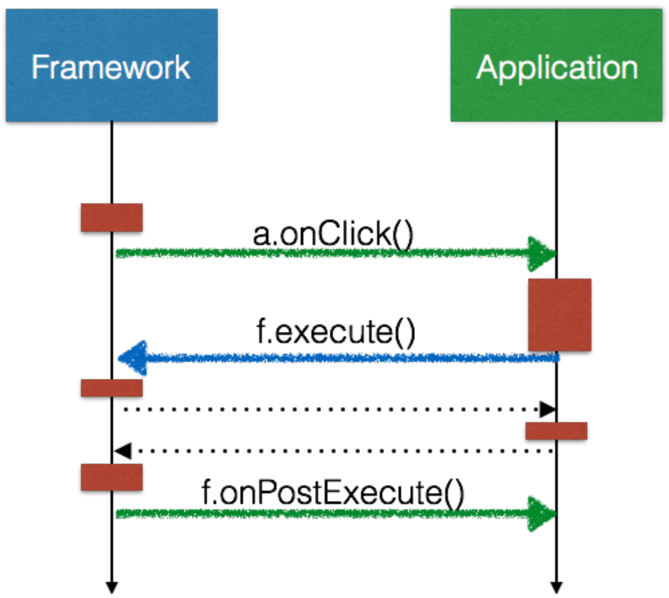


Hopper: Prove safety properties in event-driven applications by soundly jumping between callbacks

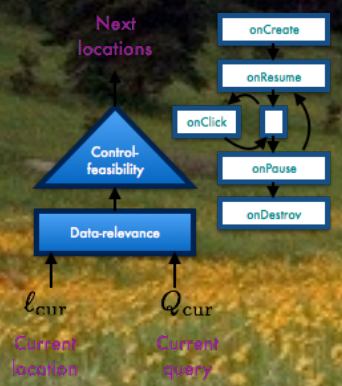
Conclusion

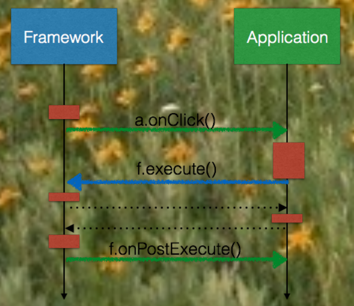


Hopper: Prove safety properties in event-driven applications by soundly jumping between callbacks



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


Cerny


Chang


Hammer


Sankaranarayanan


Somenzi

CUPLV