Refuting Heap Reachability

Bor-Yuh Evan Chang 張博聿
University of Colorado Boulder

July 31, 2014

National Chiao Tung University
國立交通大學

Sam Blackshear
CU Boulder

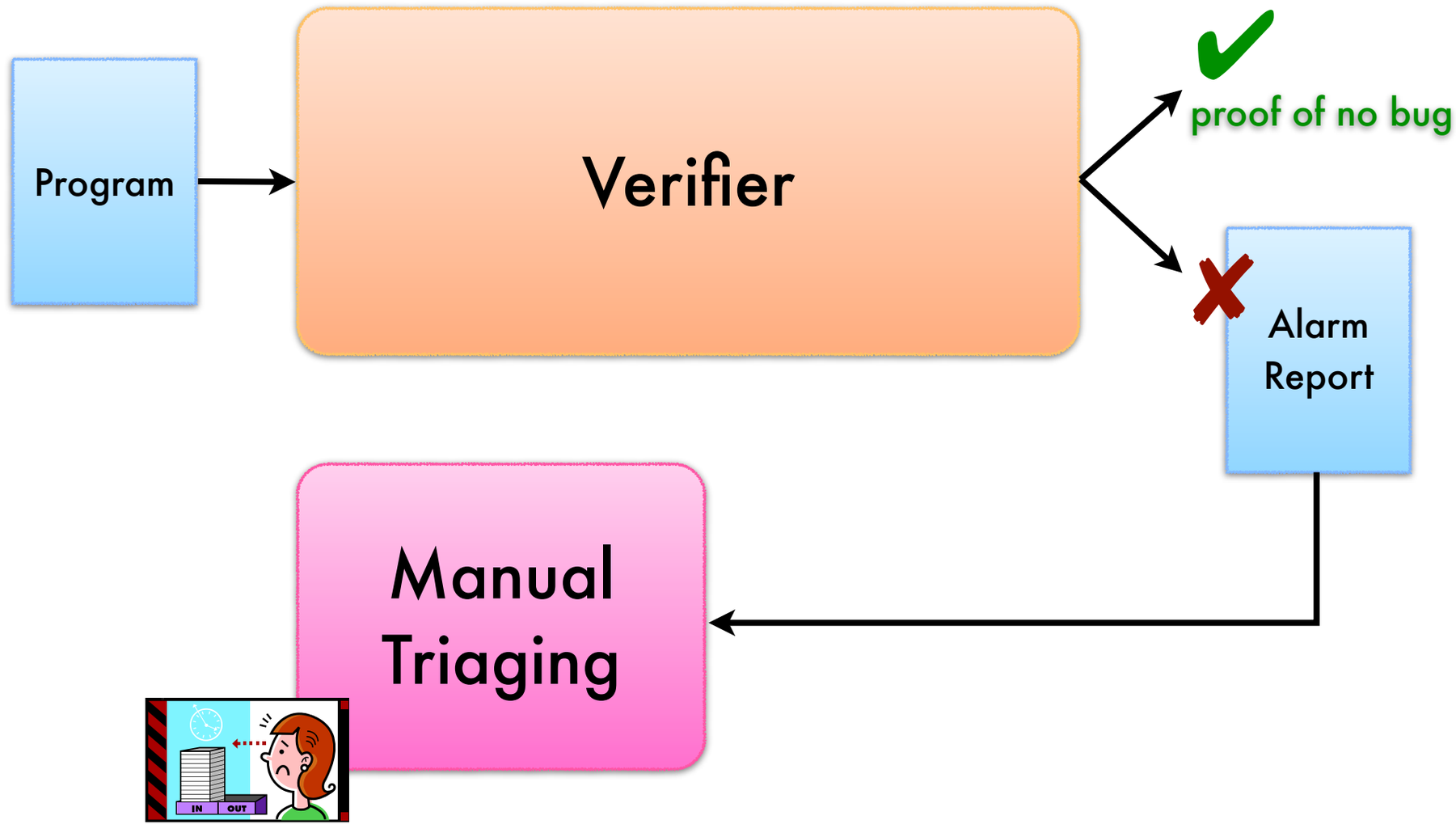
Manu Sridharan
Samsung

PLV

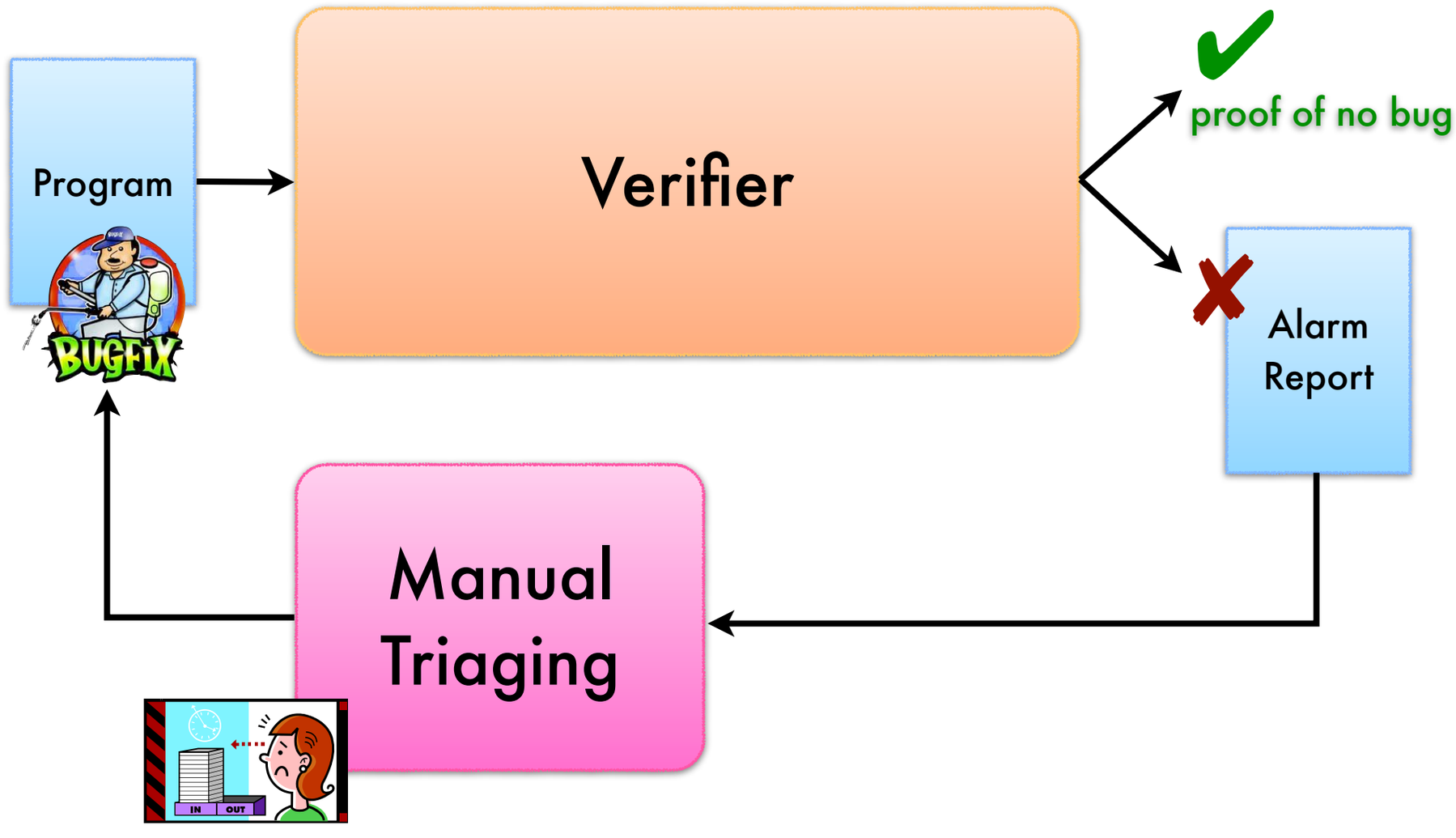
Lab: **Program analysis** in the **whole** bug mitigation **process**

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

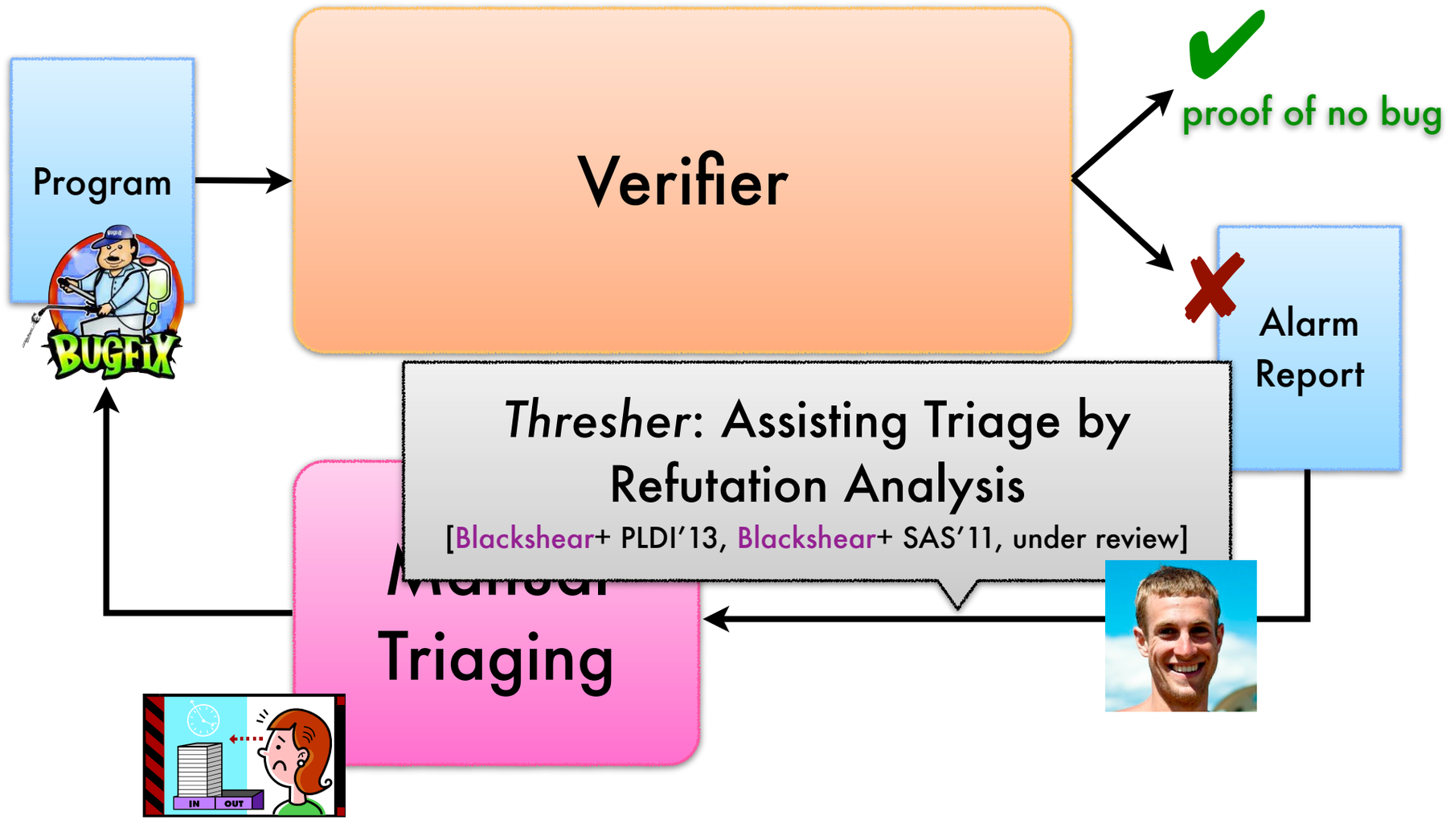
Lab: **Program analysis** in the **whole** bug mitigation **process**



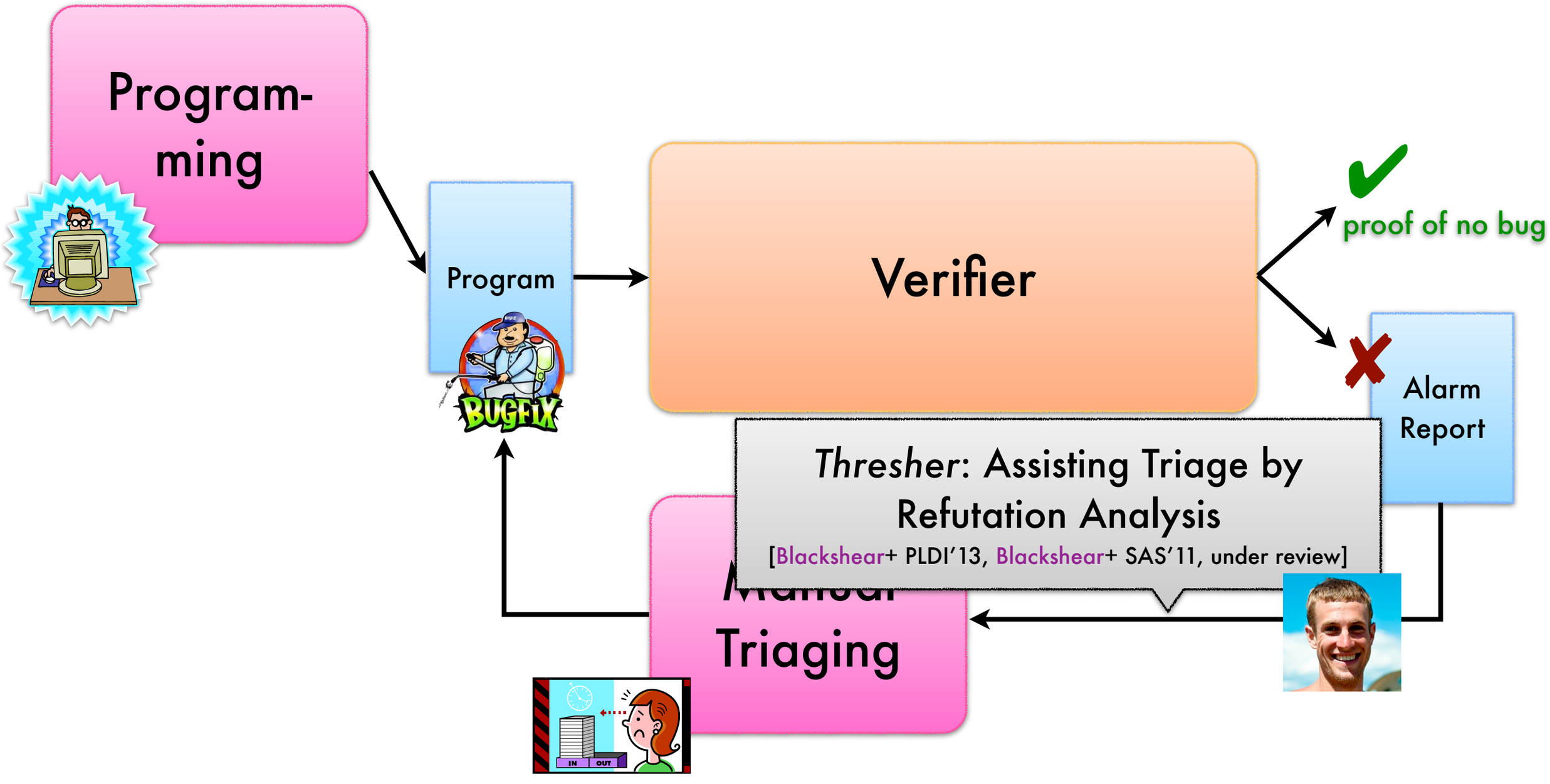
Lab: **Program analysis** in the **whole** bug mitigation **process**



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



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



Thresher: Assisting Triage by Refutation Analysis

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

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

Enforcement Windows: Measuring Bug Avoidance
[Coughlin+ ISSTA'12]

Program-ming

Program

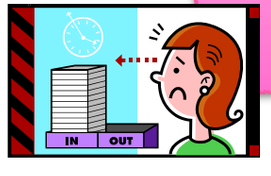
Verifier

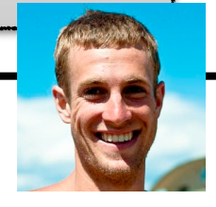
✔ proof of no bug

✘ Alarm Report

Thresher: Assisting Triage by Refutation Analysis
[Blackshear+ PLDI'13, Blackshear+ SAS'11, under review]

Manual Triaging





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

Enforcement Windows: Measuring Bug Avoidance
[Coughlin+ ISSTA'12]

Program-ming

Spec-ification

Program

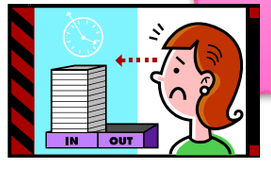
Verifier

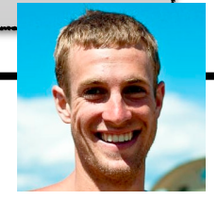
✔ proof of no bug

✘ Alarm Report

Thresher: Assisting Triage by Refutation Analysis
[Blackshear+ PLDI'13, Blackshear+ SAS'11, under review]

Manual Triaging





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

Enforcement Windows: Measuring Bug Avoidance
[Coughlin+ ISSTA'12]

Program-ming

Spec-ification

Program

Verifier

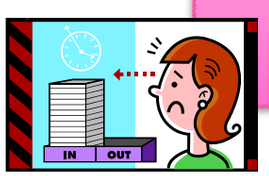
✔ proof of no bug

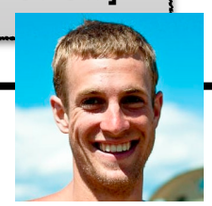
✘ Alarm Report

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

Thresher: Assisting Triage by Refutation Analysis
[Blackshear+ PLDI'13, Blackshear+ SAS'11, under review]

Manual Triaging





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

Enforcement Windows: Measuring Bug Avoidance
[Coughlin+ ISSTA'12]

Program-ming

Test Input

Spec-ification

Program

Runner

Test Output

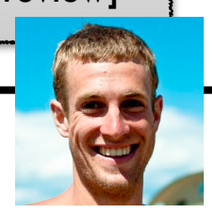
Verifier

✔ proof of no bug

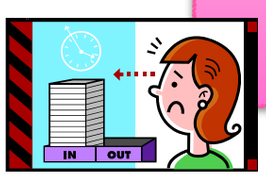
✘

Alarm Report

Thresher: Assisting Triage by Refutation Analysis
[Blackshear+ PLDI'13, Blackshear+ SAS'11, under review]



Manual Triaging



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

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

Enforcement Windows: Measuring Bug Avoidance
[Coughlin+ ISSTA'12]

Program-ming

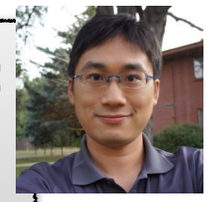
Fissile Types: Checking Almost Everywhere Invariants
[Coughlin+ POPL'14, in prep]

Test Input

Spec-ification

Program

Static Incrementalization of Data Structure Checks
[under review]



Runner

Test Output

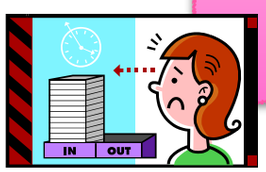
Verifier

✔ proof of no bug

✘ Alarm Report

Thresher: Assisting Triage by Refutation Analysis
[Blackshear+ PLDI'13, Blackshear+ SAS'11, under review]

Manual Triaging



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

Enforcement Windows: Measuring Bug Avoidance
[Coughlin+ ISSTA'12]

Program-ming

Test Input

Spec-ification

Program

Runner

Static Incrementalization of Data Structure Checks
[under review]

Test Output

Jsana: Abstract Domain Combinators for Dynamic Languages
[Cox+ ECOOP'13, Cox+ SAS'14, under review]

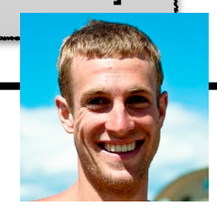
Verifier

✔ proof of no bug

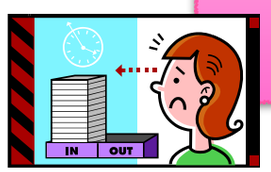
✘

Alarm Report

Thresher: Assisting Triage by Refutation Analysis
[Blackshear+ PLDI'13, Blackshear+ SAS'11, under review]



Manual Triaging



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

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

Enforcement Windows: Measuring Bug Avoidance
[Coughlin+ ISSTA'12]

Static Incrementalization of Data Structure Checks
[under review]

Test Input

Runner

Test Output

Program-ming

Spec-ification

Program

Jsana: Abstract Domain Combinators for Dynamic Languages
[Cox+ ECOOP'13, Cox+ SAS'14, under review]

Verifier

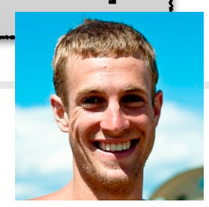
proof of no bug

Alarm Report

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

This Talk

Thresher: Assisting Triage by Refutation Analysis
[Blackshear+ PLDI'13, Blackshear+ SAS'11, under review]



Manual Triaging

Refuting Heap Reachability

Bor-Yuh Evan Chang 張博聿
University of Colorado Boulder

July 31, 2014

National Chiao Tung University
國立交通大學

Sam Blackshear
CU Boulder

Manu Sridharan
Samsung

PLV

A bug that manifests spectacularly …

A bug that manifests spectacularly …

A bug that manifests spectacularly …

A bug that manifests spectacularly …

Wow! Android memory leaks underly rotation-based crashes.

Wow! Android memory leaks underly rotation-based crashes.

How?!?

Wow! Android memory leaks underly rotation-based crashes.

How?!?

`Activity` objects encapsulate the UI

Wow! Android memory leaks underly rotation-based crashes.

How?!?

`Activity` objects encapsulate the UI

Android OS

of type `Activity`

Wow! Android memory leaks underly rotation-based crashes.

How?!?

`Activity` objects encapsulate the UI

Android OS

of type `Activity`

Wow! Android memory leaks underly rotation-based crashes.

How?!?

`Activity` objects encapsulate the UI

of type `Activity`

of type `Activity`

Wow! Android memory leaks underly rotation-based crashes.

How?!?

`a_static_field`

program heap

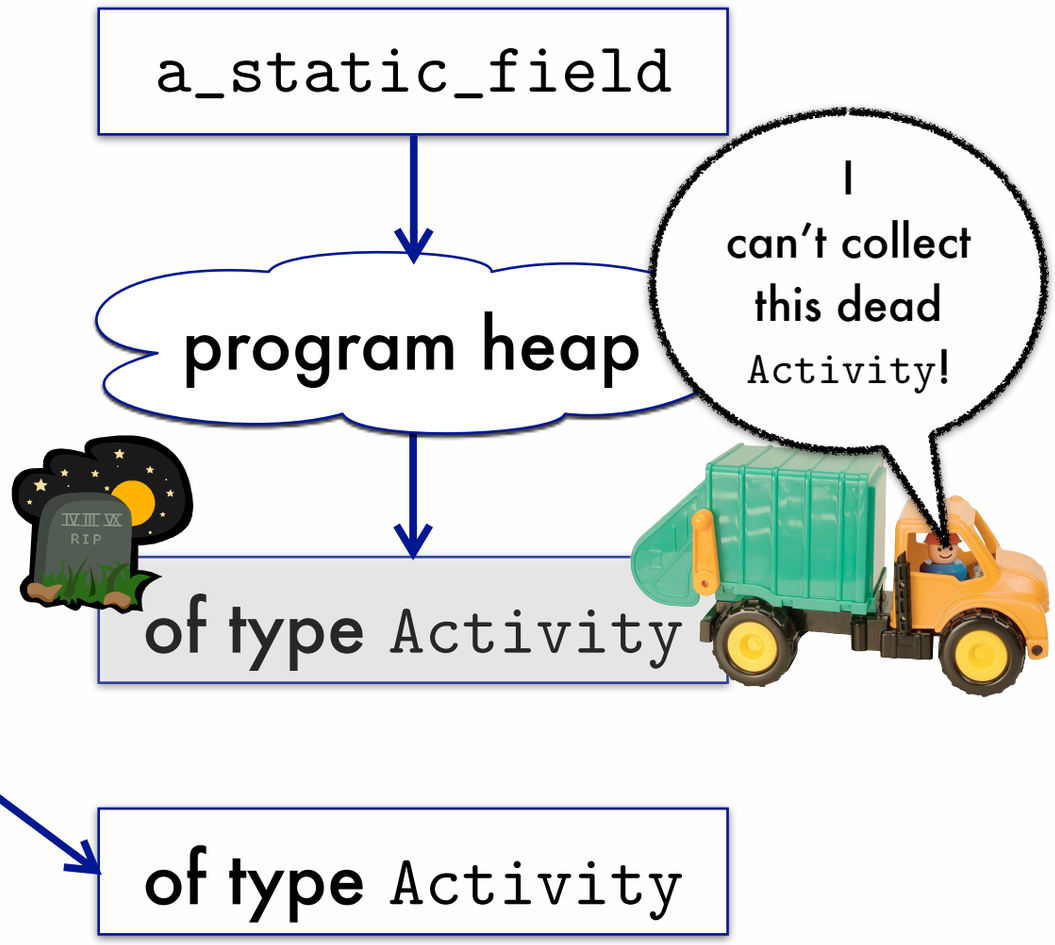
of type `Activity`

of type `Activity`

`Activity` objects encapsulate the UI

Wow! Android memory leaks underly rotation-based crashes.

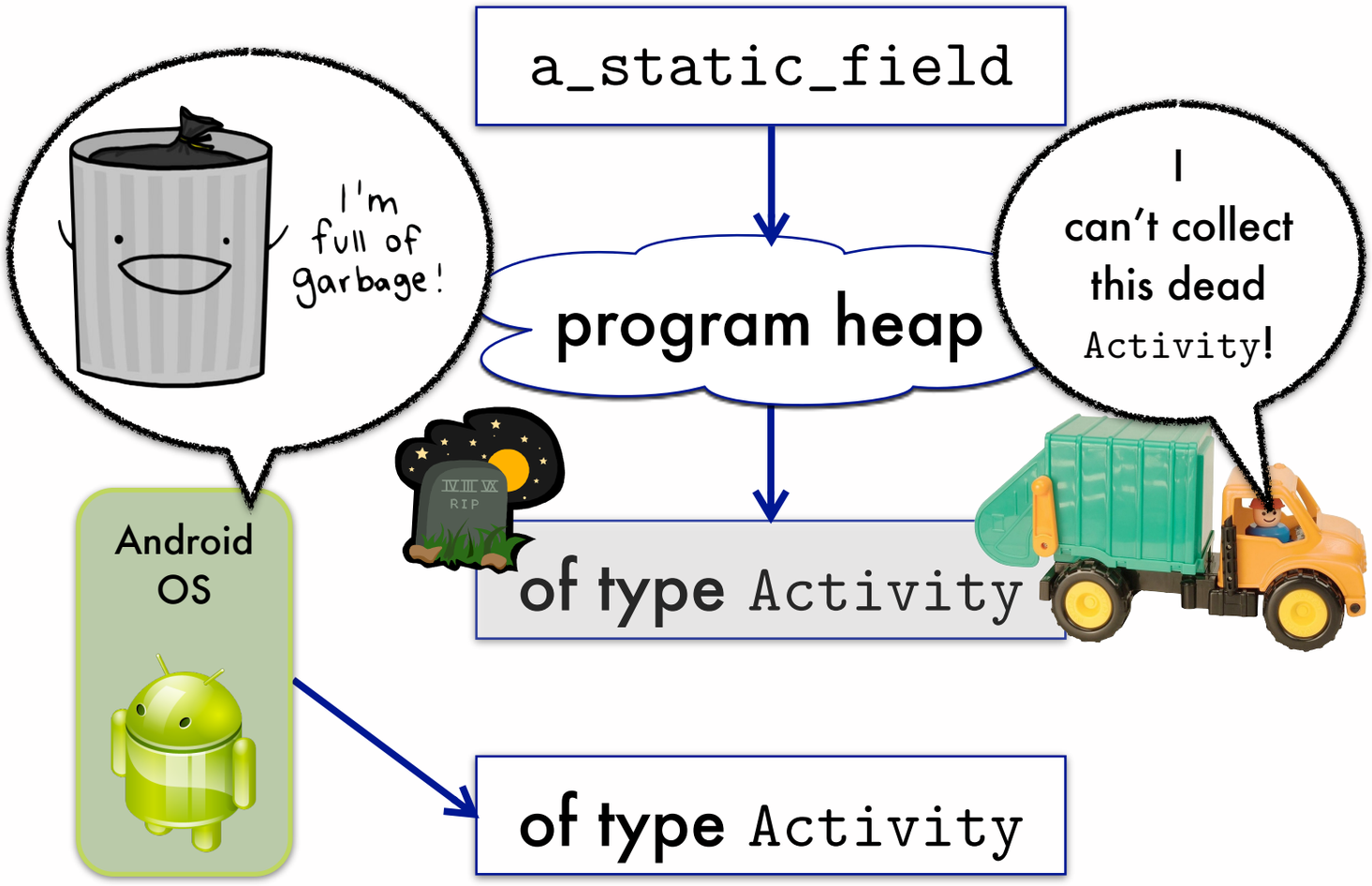
How?!?



`Activity` objects encapsulate the UI

Wow! Android memory leaks underly rotation-based crashes.

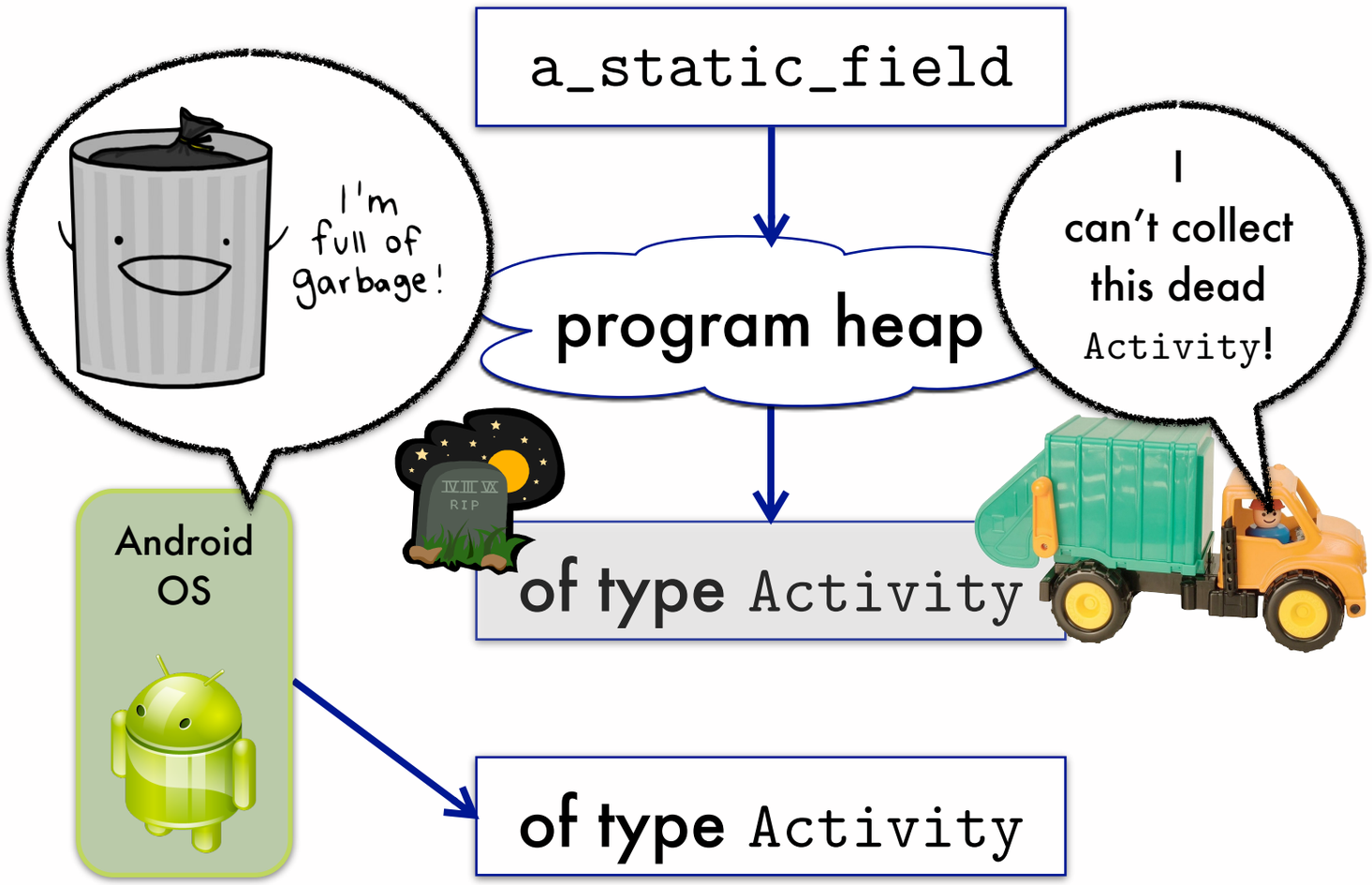
How?!?



`Activity` objects encapsulate the UI

Wow! Android memory leaks underly rotation-based crashes.

How?!?

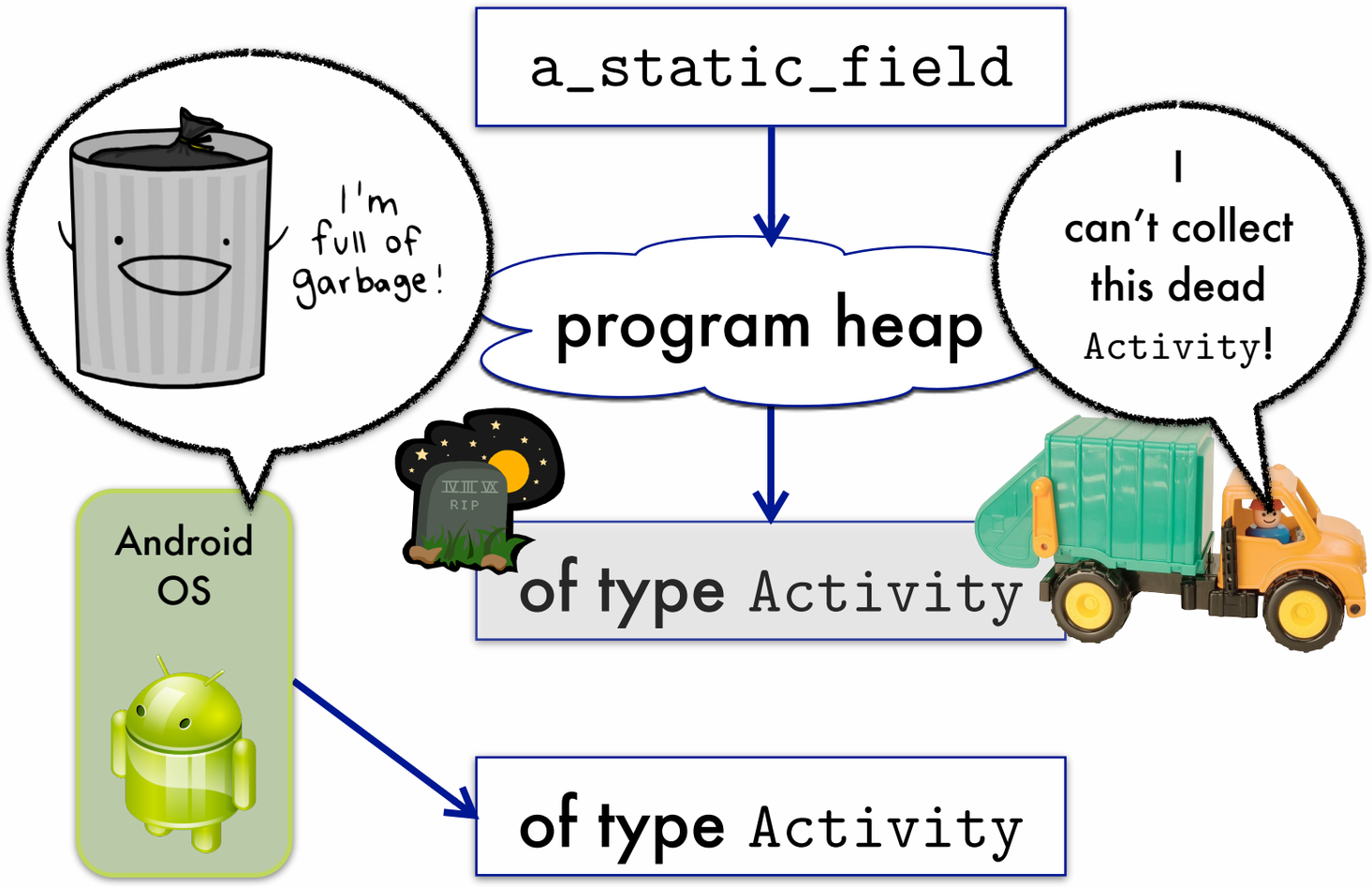


`Activity` objects encapsulate the UI

Bug: Holding reference to "old" `Activity`

Wow! Android memory leaks underly rotation-based crashes.

How?!?



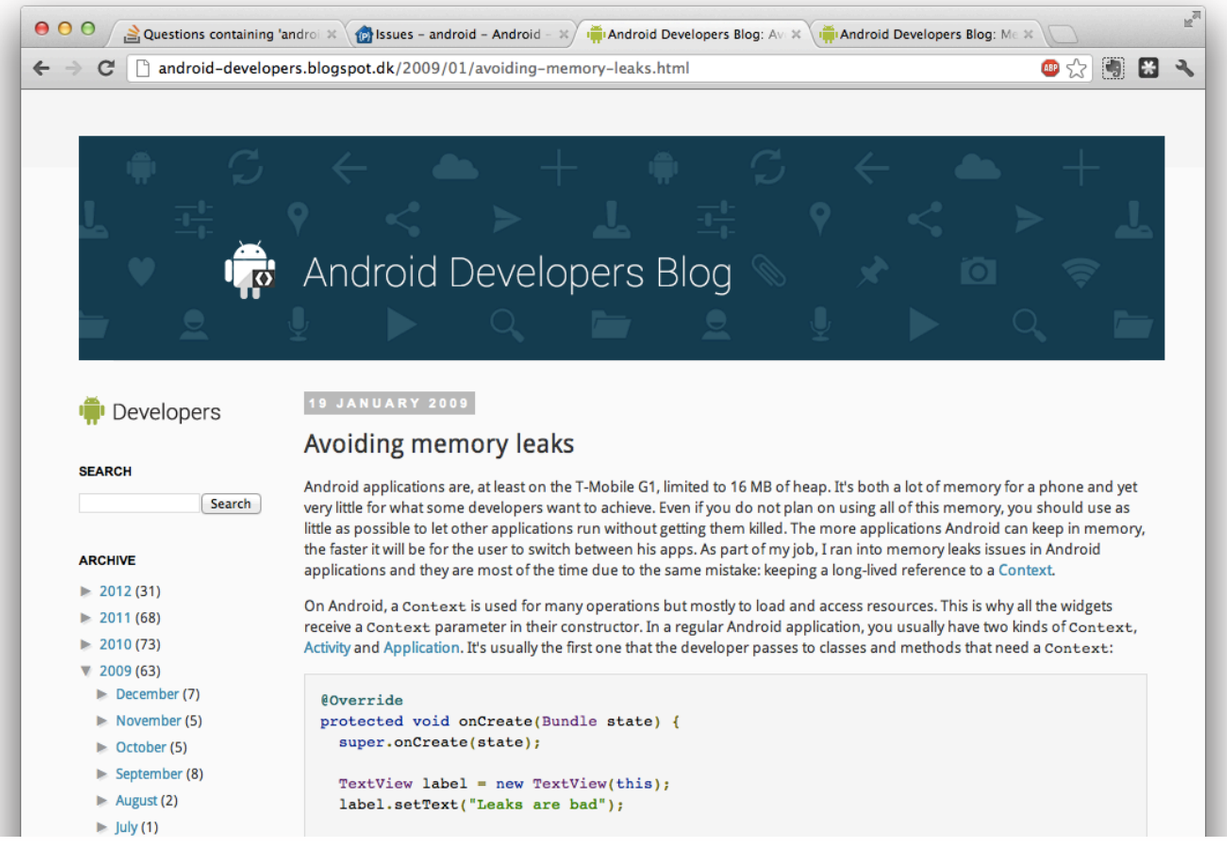
`Activity` objects encapsulate the UI

"an `Activity` leak"

Bug: Holding reference to "old" `Activity`

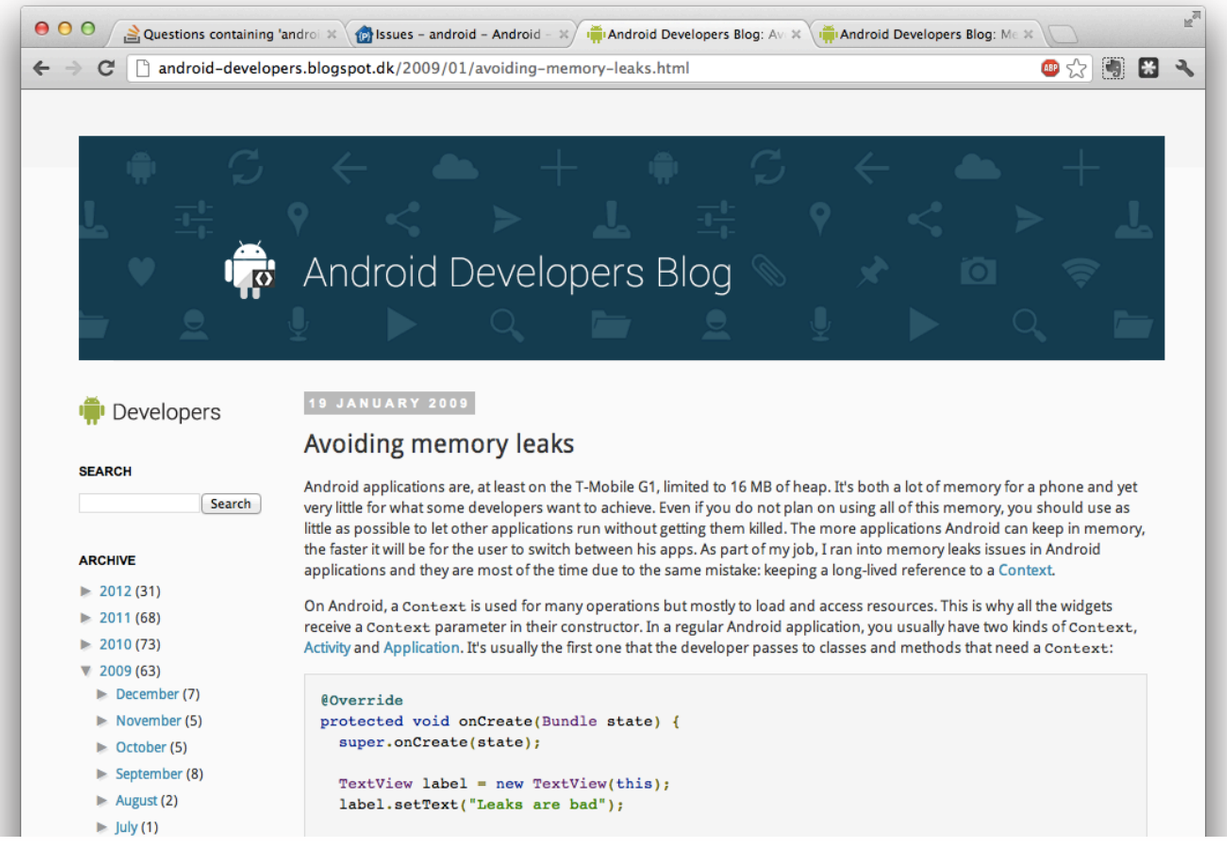
The expert recommendation ...

The expert recommendation ...



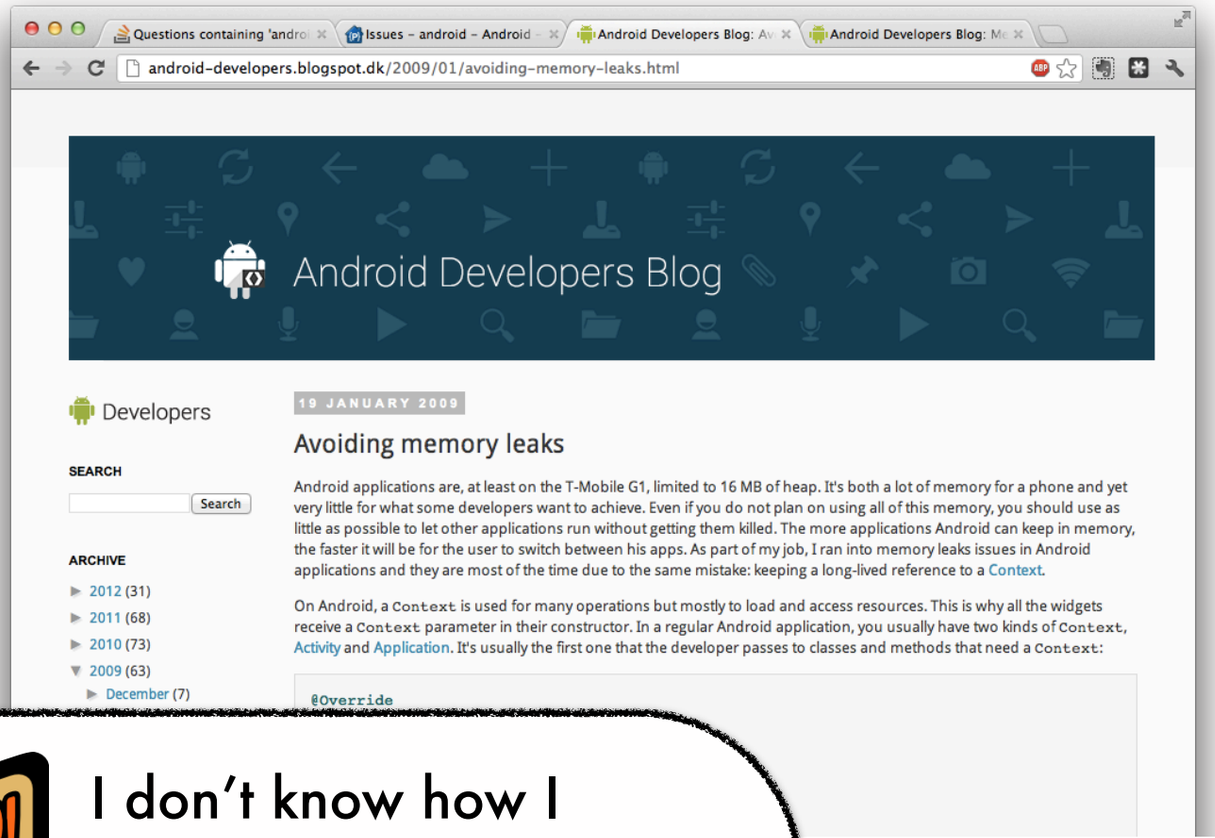
The expert recommendation …

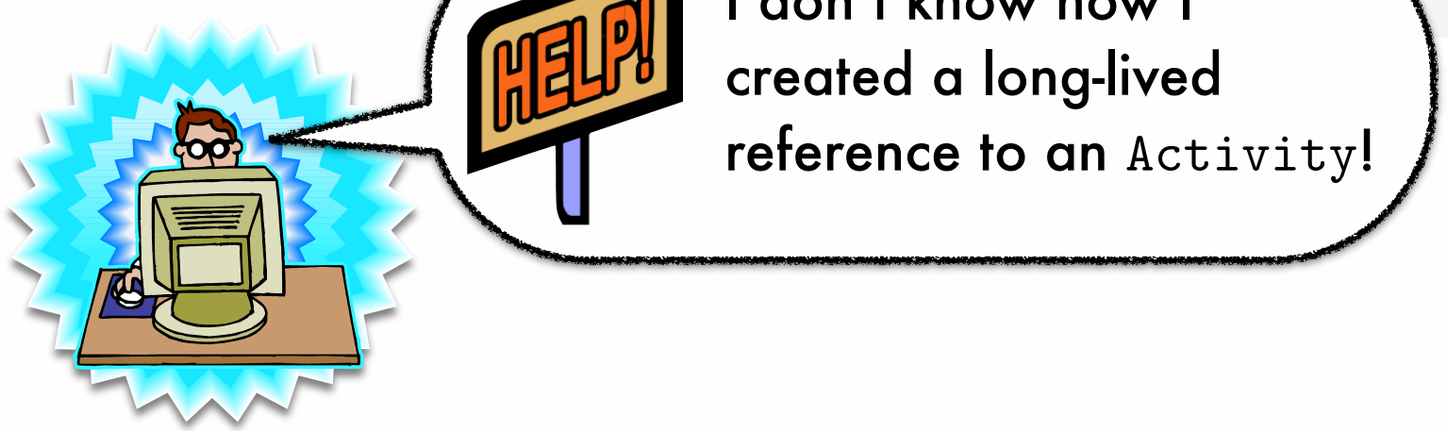
"Do not keep long-lived references to a context-activity"



The expert recommendation …

"Do not keep long-lived references to a context-activity"





The expert recommendation …

"Do not keep long-lived references to a context-activity"

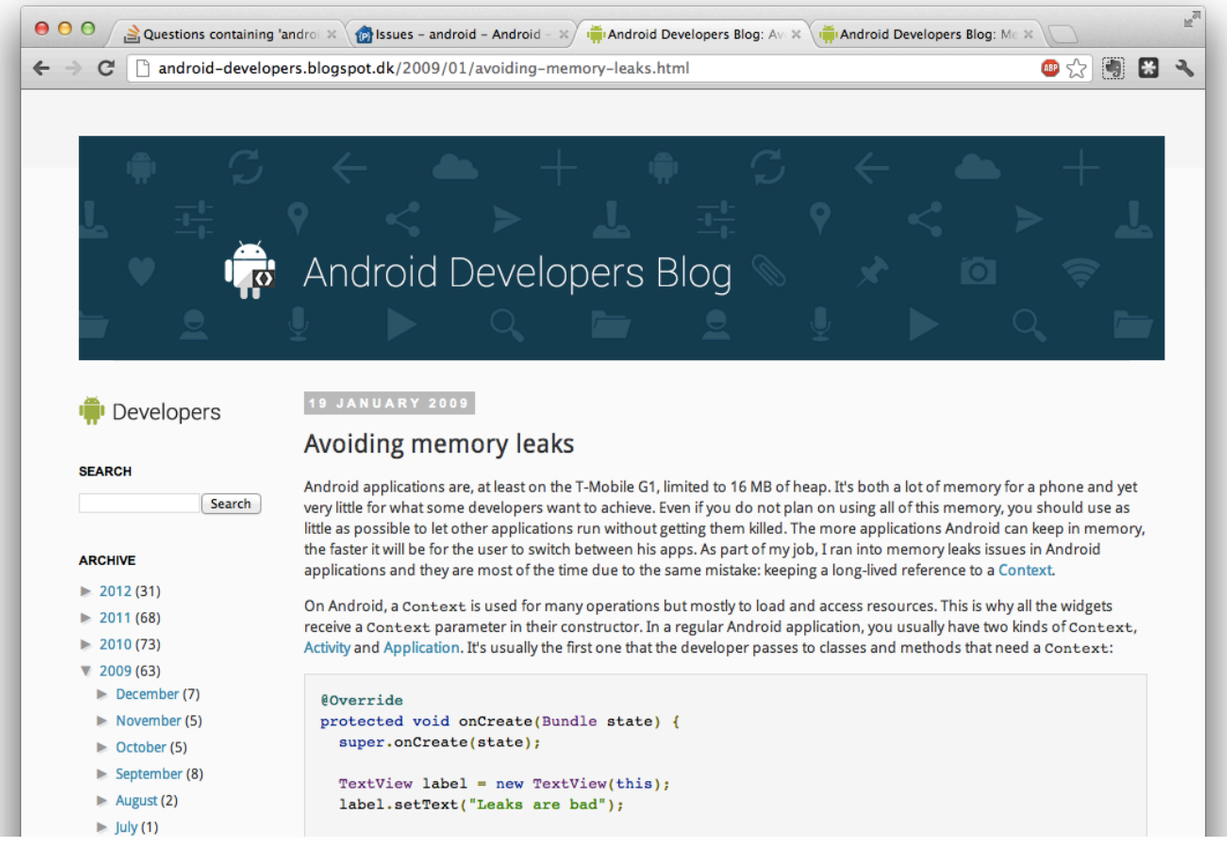
I don't know how I created a long-lived reference to an `Activity`!

Often: A **misunderstanding** of a library causes the **library** to keep the `Activity` reference.


Verification

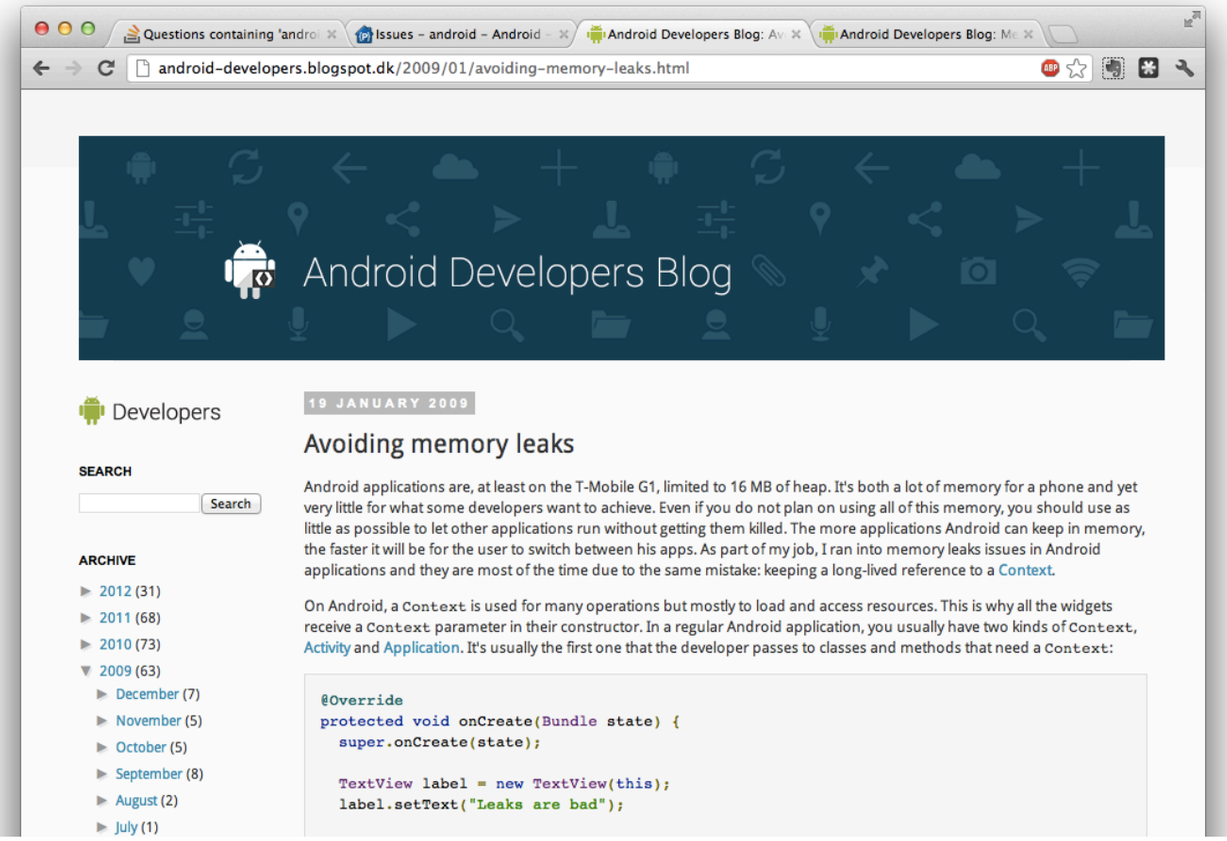
The expert recommendation ...

The expert recommendation ...



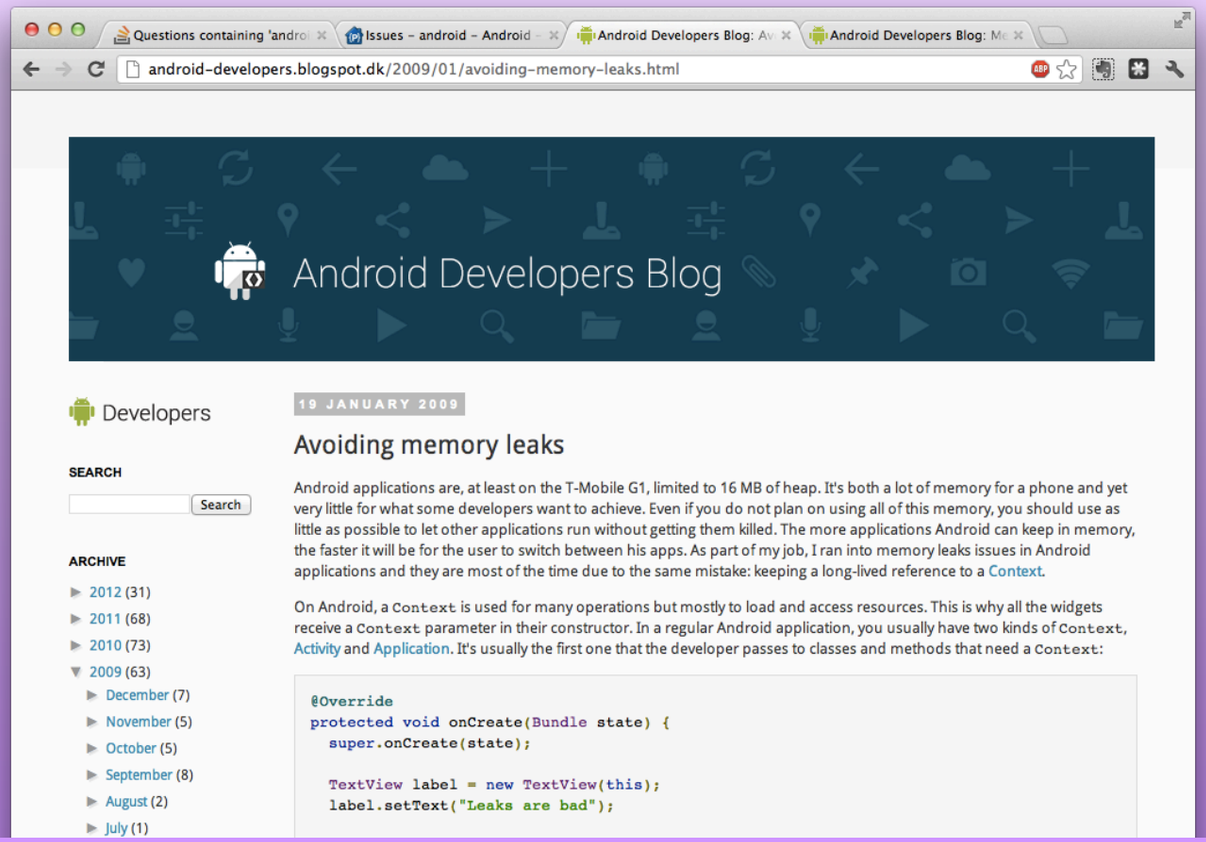
The expert recommendation …

"Do not keep long-lived references to a context-activity"



The expert recommendation ...

"Do not keep long-lived references to a context-activity"



A Specific Property to Check:

No `Activity` is ever reachable from a static field.

A candidate for statically answering, "Is there an `Activity` leak?"

Is there a program execution where at some time

`a_static_field`

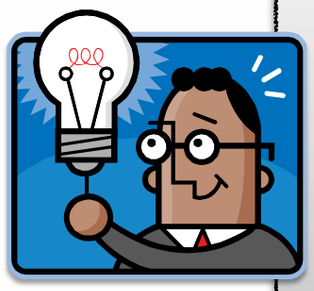
of type `Activity` ?

A candidate for statically answering, "Is there an `Activity` leak?"

Is there a program execution where at some time

`a_static_field`

of type `Activity` ?



Can be answered with a points-to analysis

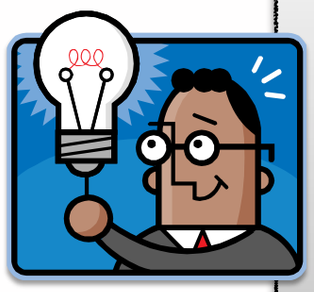
A candidate for statically answering, "Is there an `Activity` leak?"

Is there a program execution where at some time

`a_static_field`

⋮

of type `Activity` ?



Can be answered with a points-to analysis

Compute a points-to graph and look for such points-to paths

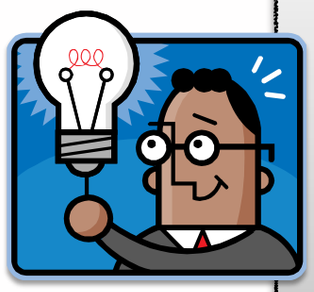
A candidate for statically answering, "Is there an `Activity` leak?"

Is there a program execution where at some time

`a_static_field`

⋮

of type `Activity` ?



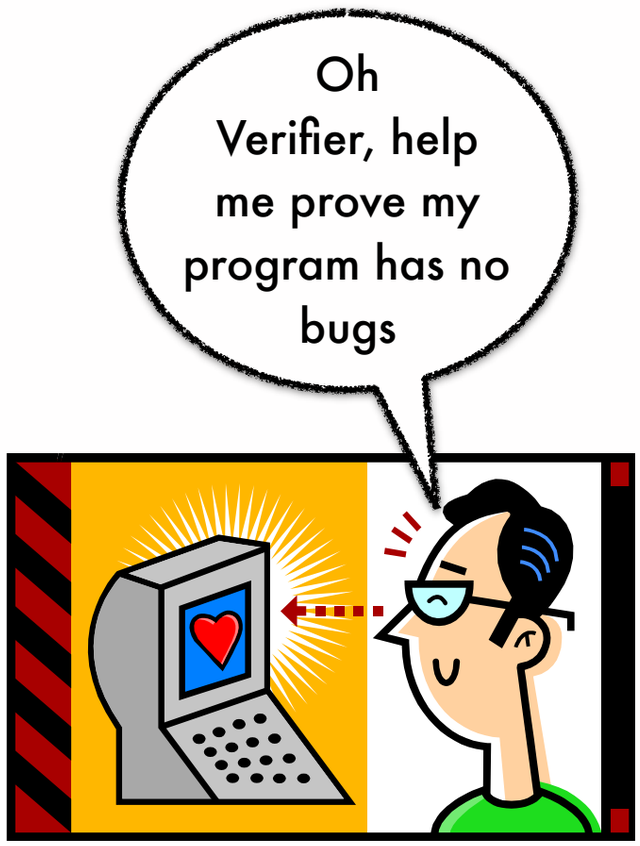
Can be answered with a points-to analysis

Compute a points-to graph and look for such points-to paths

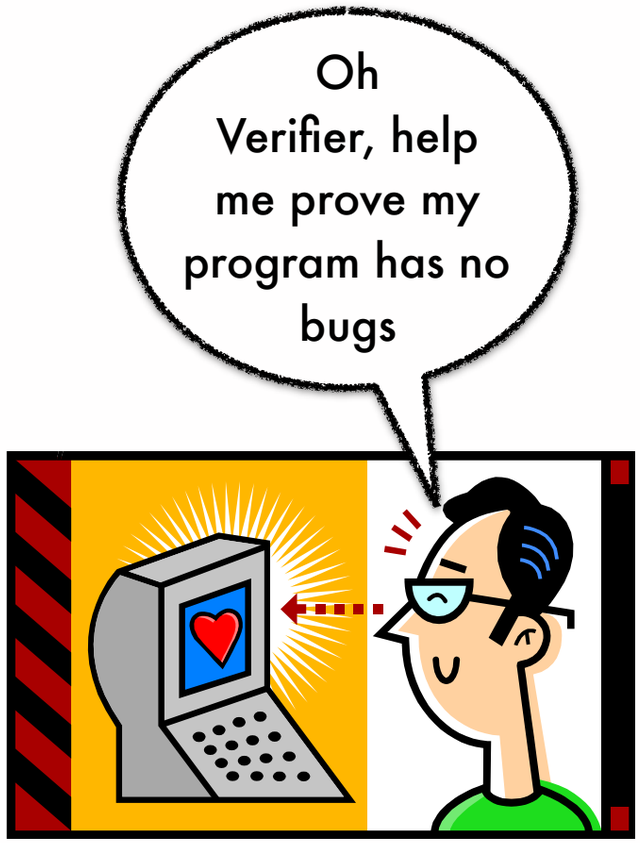
This won't work because …

The well-known false alarm problem!

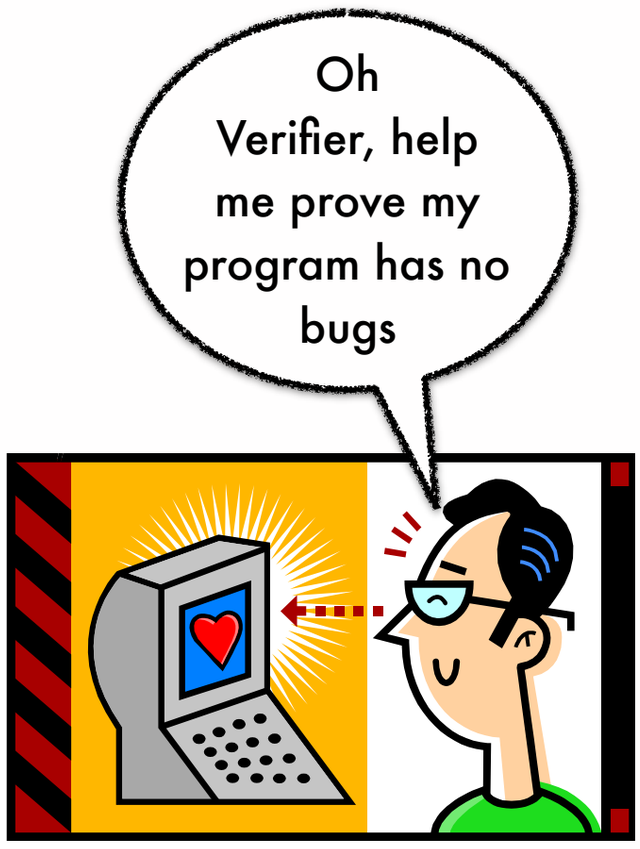
The well-known false alarm problem!



The well-known false alarm problem!



The well-known false alarm problem!

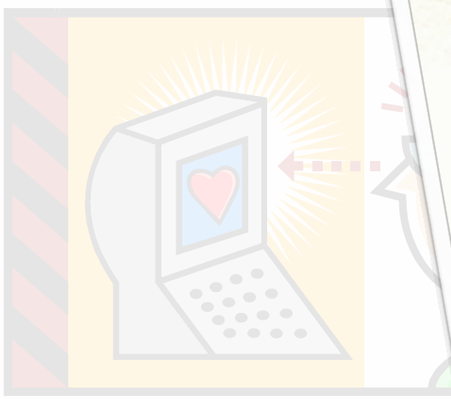


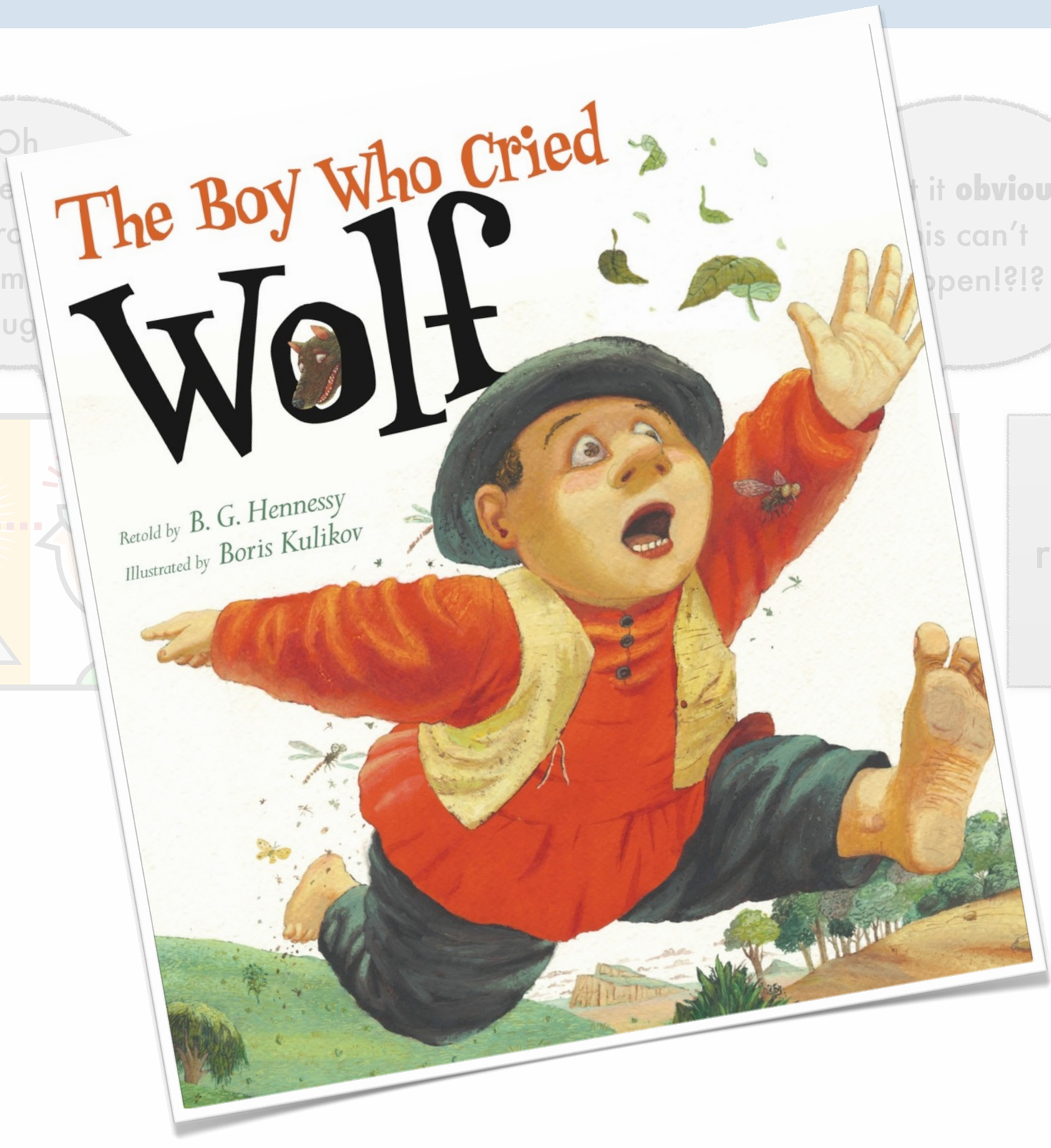
And noisily repeated over and over!

The well-known false alarm problem!

Oh
Verifie
me pro
program
bug

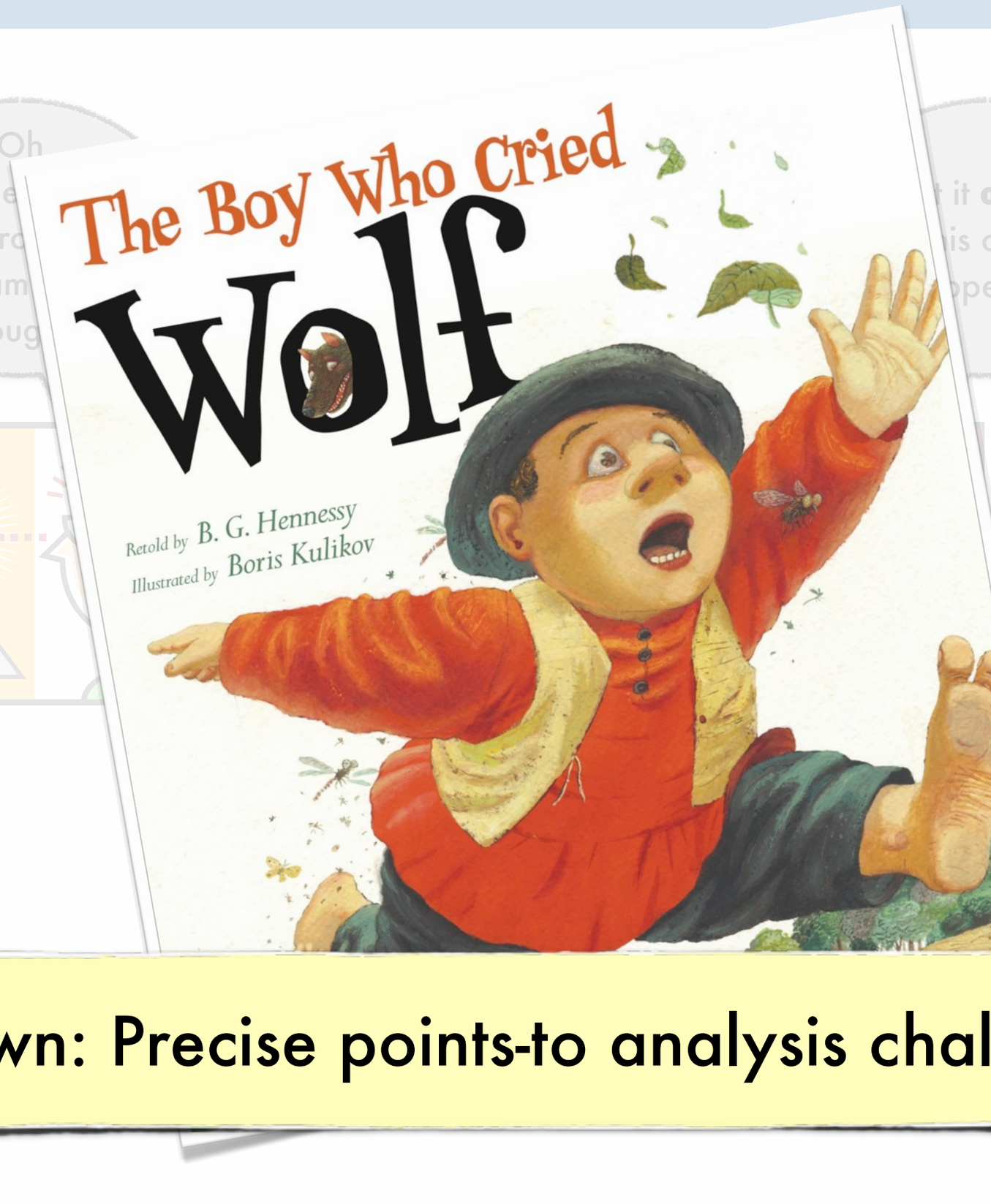
it **obvious**
is can't
pen!?!?





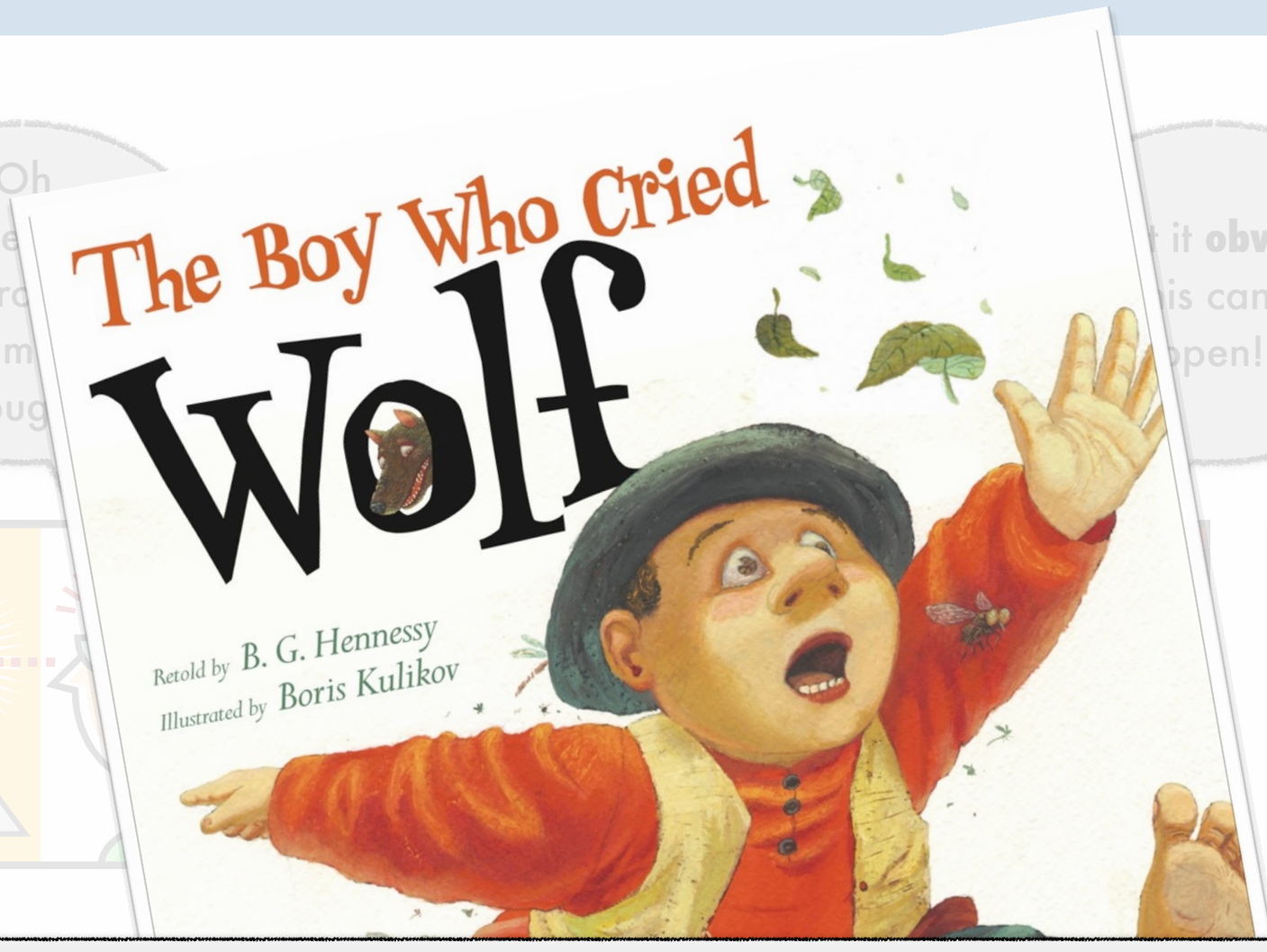
And noisily
repeated over
and over!

The well-known false alarm problem!



Known: Precise points-to analysis challenging

The well-known false alarm problem!

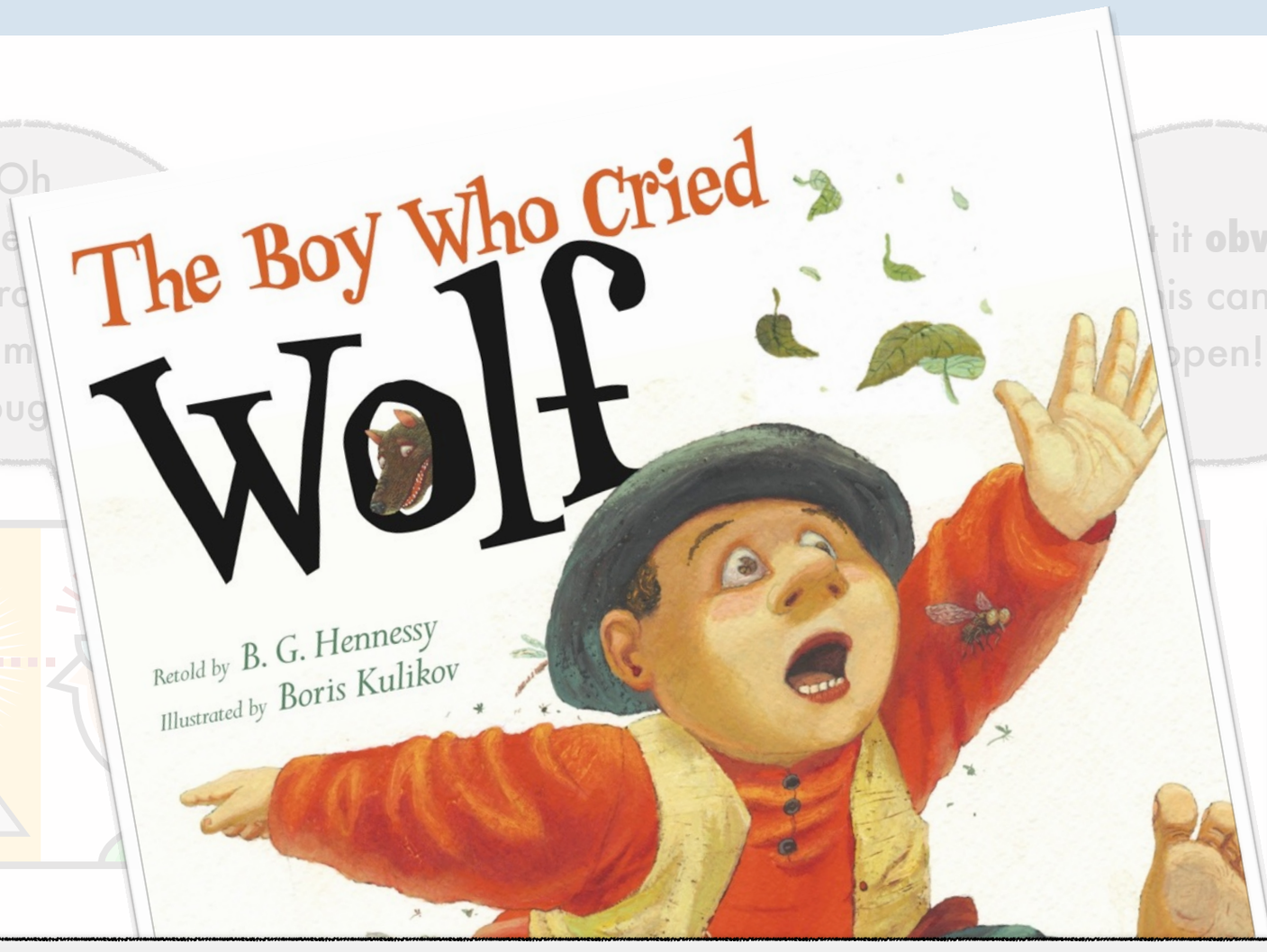


Hind (2001). "Pointer Analysis: Haven't We Solved This Problem Yet?"

- 75 papers, 9 PhD theses

Known: Precise points-to analysis challenging

The well-known false alarm problem!



Hind (2001). "Pointer Analysis: Haven't We Solved This Problem Yet?"

- 75 papers, 9 PhD theses

Known: Precise points-to analysis ~~challenging~~

enough

impossible?


Verification

Perspective: The false alarm problem

Perspective: The false alarm problem

Verifier

Perspective: The false alarm problem

A "union" of tools and techniques (e.g., model checking, abstract interpretation)

Verifier

Perspective: The false alarm problem

A "union" of tools and techniques (e.g., model checking, abstract interpretation)

Spec-ification

Program

Verifier

Perspective: The false alarm problem

A "union" of tools and techniques (e.g., model checking, abstract interpretation)

Spec-ification

Program

Verifier

✔

proof of no bug

or

witness to a bug

Perspective: The false alarm problem

A "union" of tools and techniques (e.g., model checking, abstract interpretation)

Spec-ification

Program

Verifier

✔

Happy!

proof of no bug

or

witness to a bug

Perspective: The false alarm problem

A "union" of tools and techniques (e.g., model checking, abstract interpretation)

Spec-ification

Program

Verifier

✔

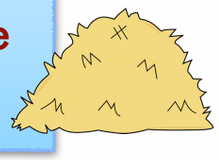
Happy!

proof of no bug

or

witness to a bug

✘ alarms of maybe bugs



or

timeout

"bound-out"

Perspective: The false alarm problem

A "union" of tools and techniques (e.g., model checking, abstract interpretation)

Spec-ification

Program

Verifier

✔

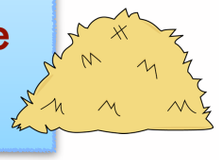
proof of no bug

or

witness to a bug

Happy!

✘ alarms of maybe bugs



or

timeout "bound-out"

Unhappy but inevitable. Research work to minimize

Perspective: The false alarm problem

A "union" of tools and techniques (e.g., model checking, abstract interpretation)

Spec-ification

Program

Verifier

✔

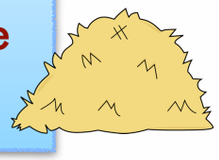
proof of no bug

or

witness to a bug

Happy!

✘ alarms of maybe bugs



or

timeout "bound-out"

Unhappy but inevitable. Research work to minimize

Different classes of tools make different trade-offs:
Recognize strength in combining MC-AI approaches

Perspective: The false alarm problem

A "union" of tools and techniques (e.g., model checking, abstract interpretation)

Spec-ification

Program

Verifier

✔

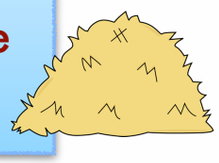
proof of no bug

or

witness to a bug

Happy!

✘ alarms of maybe bugs



or

timeout "bound-out"

This talk: Applied to heap reachability

Unhappy but inevitable. Research work to minimize

Different classes of tools make different trade-offs: Recognize strength in combining MC-AI approaches

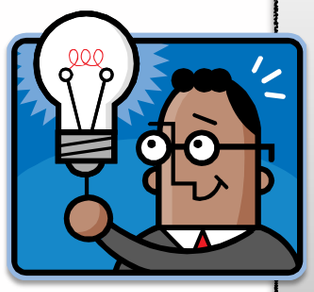
A candidate for statically answering, "Is there an `Activity` leak?"

Is there a program execution where at some time

`a_static_field`

⋮

of type `Activity` ?



Can be answered with a points-to analysis

Compute a points-to graph and look for such points-to paths

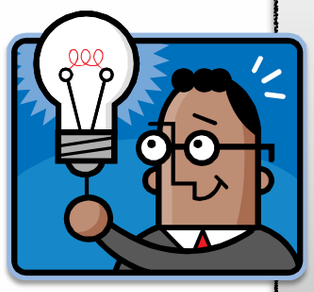
A candidate for statically answering, "Is there an `Activity` leak?"

Is there a program execution where at some time

`a_static_field`

⋮

of type `Activity` ?

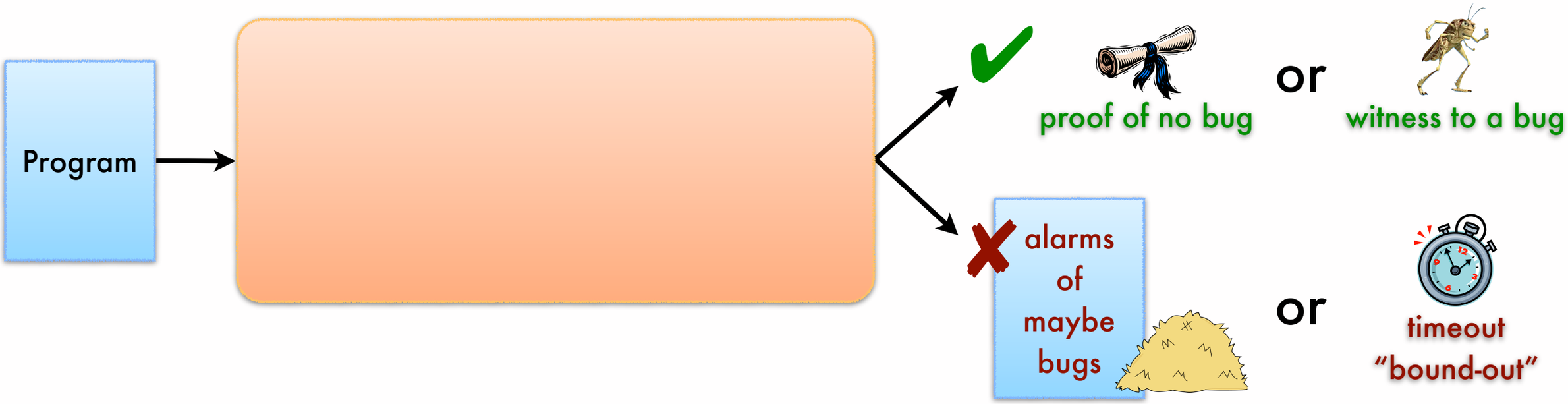


Can be answered with a points-to analysis

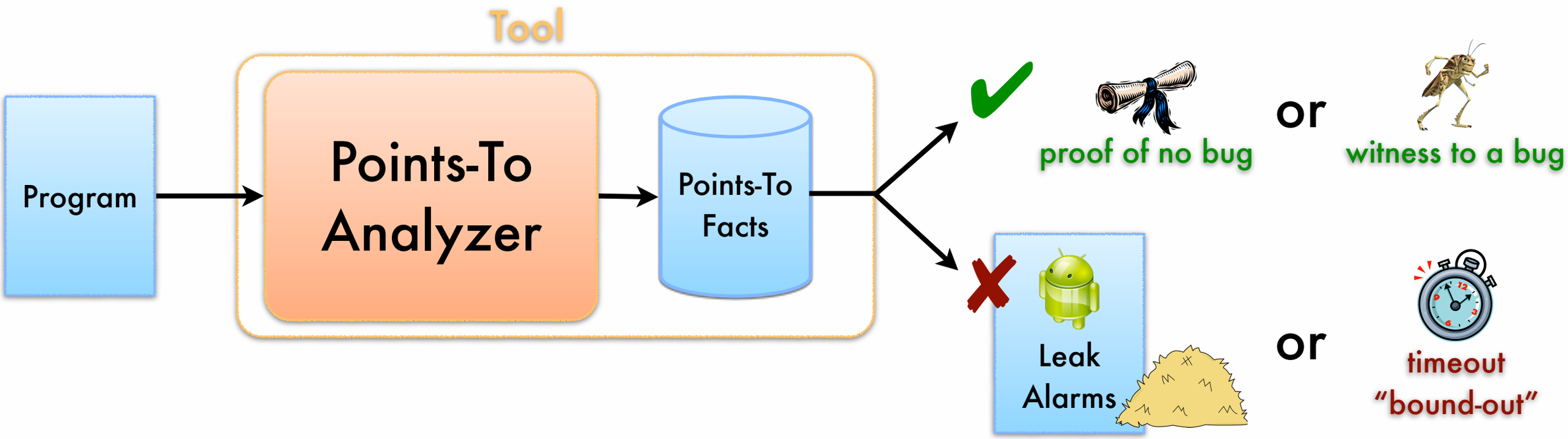
Compute a points-to graph and look for such points-to paths

Let's make it work!

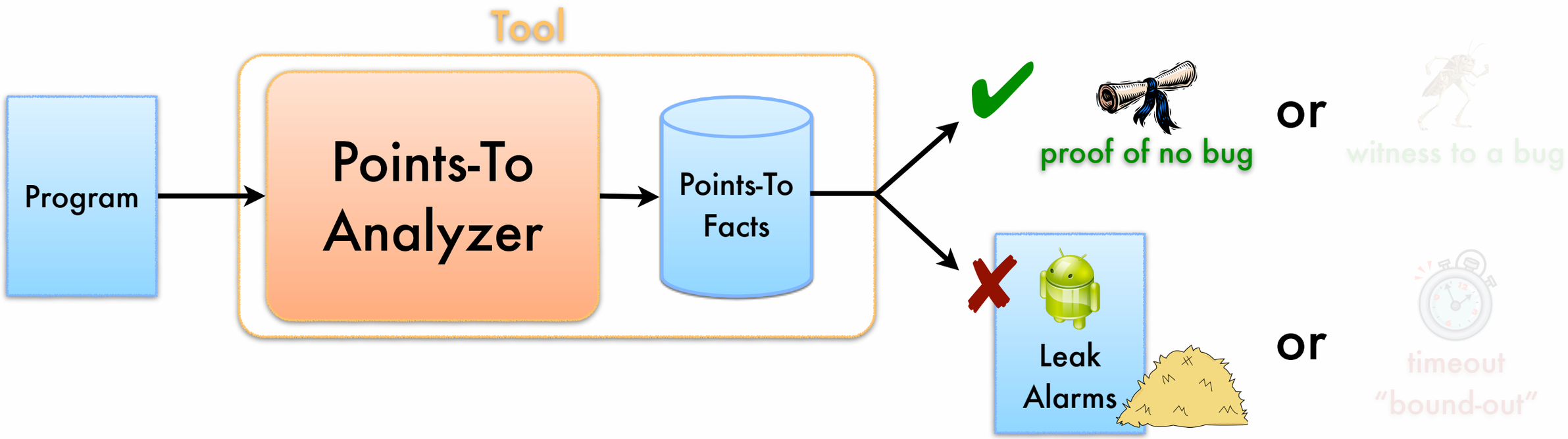
Thresher [SAS'11,PLDI'13] attacks alarm triage for heap reachability properties



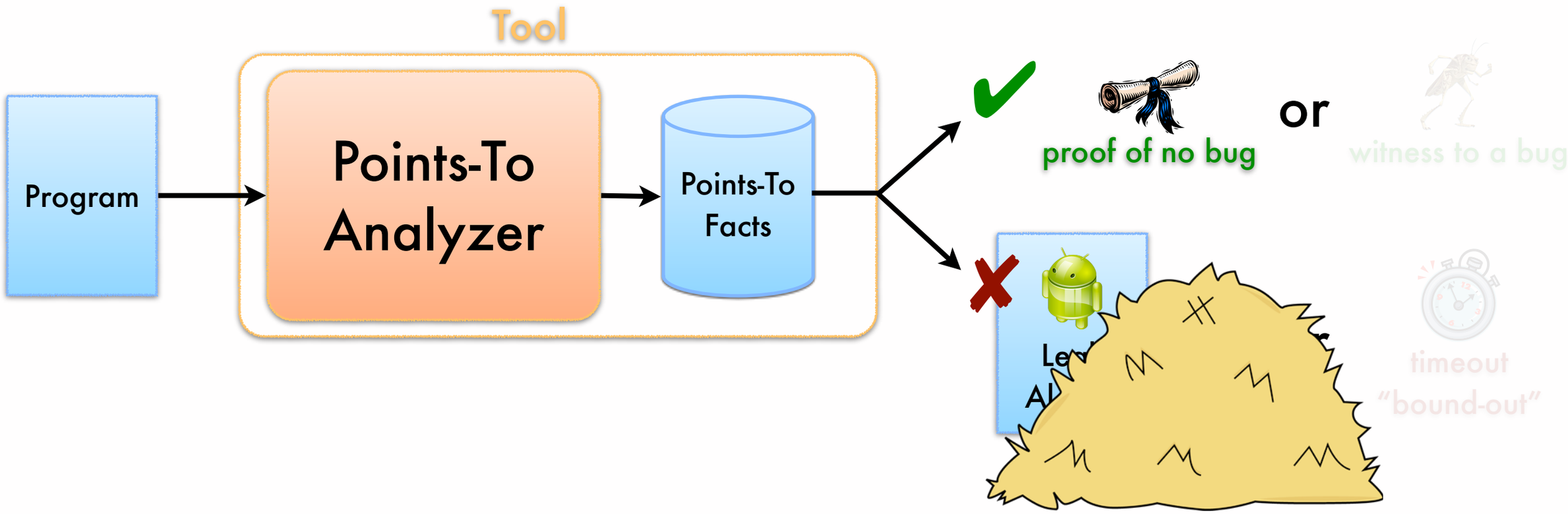
Thresher [SAS'11,PLDI'13] attacks alarm triage for heap reachability properties



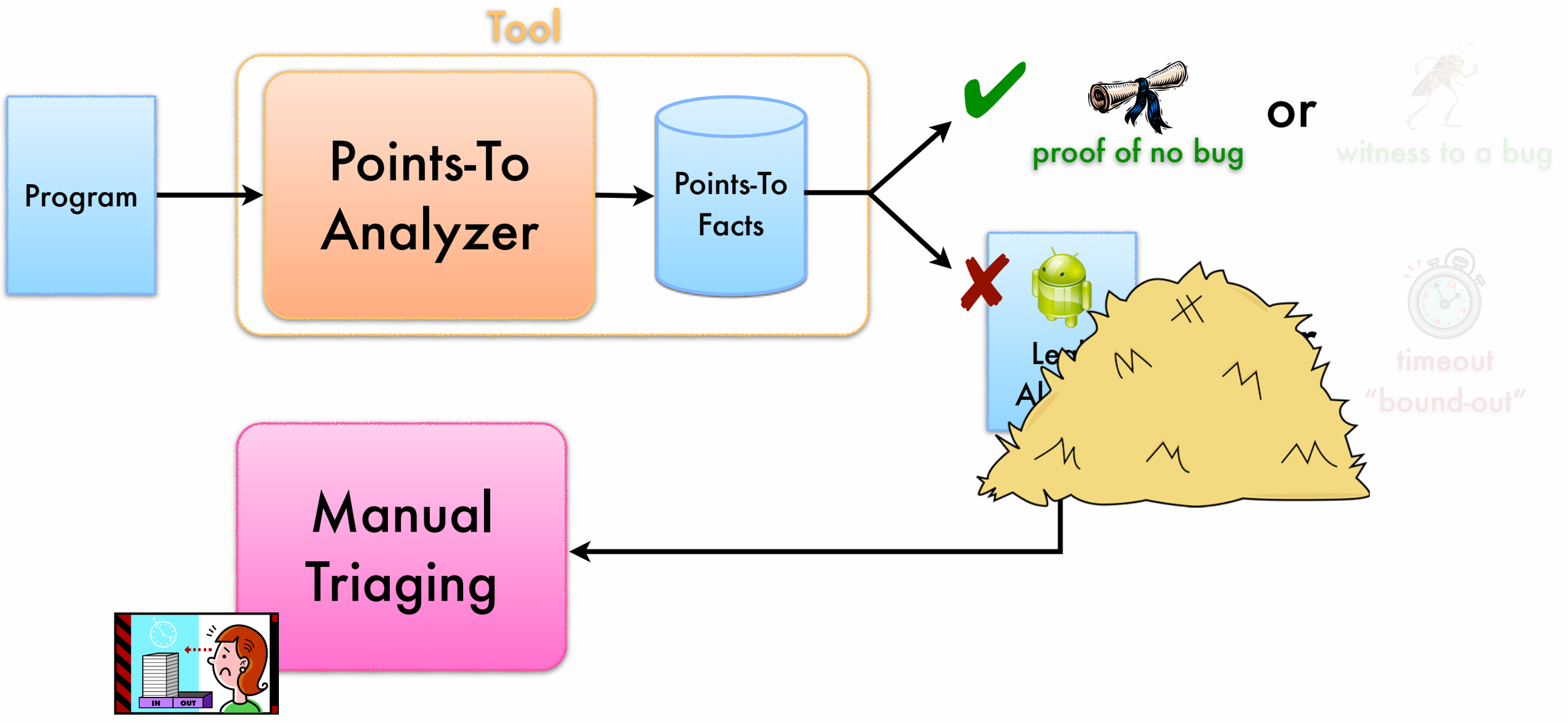
Thresher [SAS'11,PLDI'13] attacks alarm triage for heap reachability properties



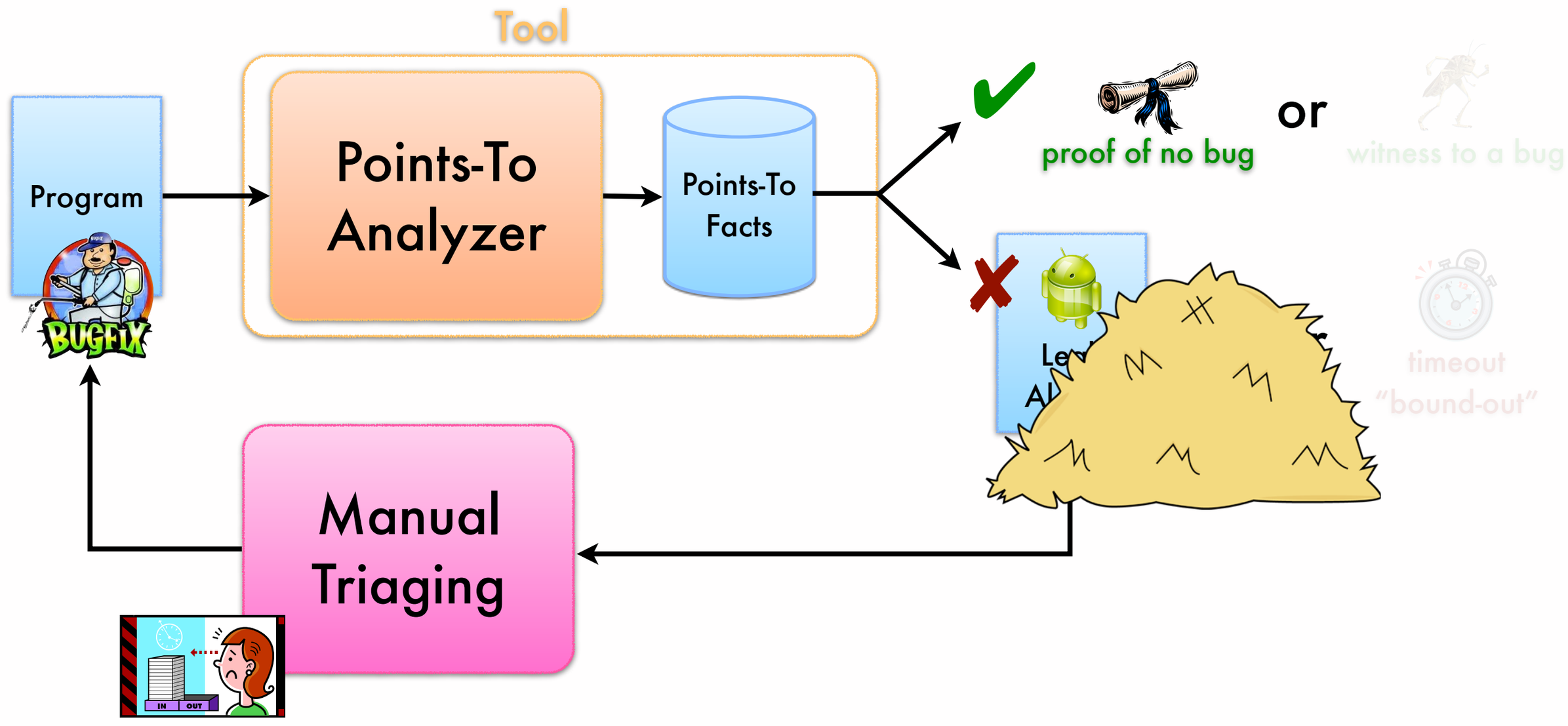
Thresher [SAS'11,PLDI'13] attacks alarm triage for heap reachability properties



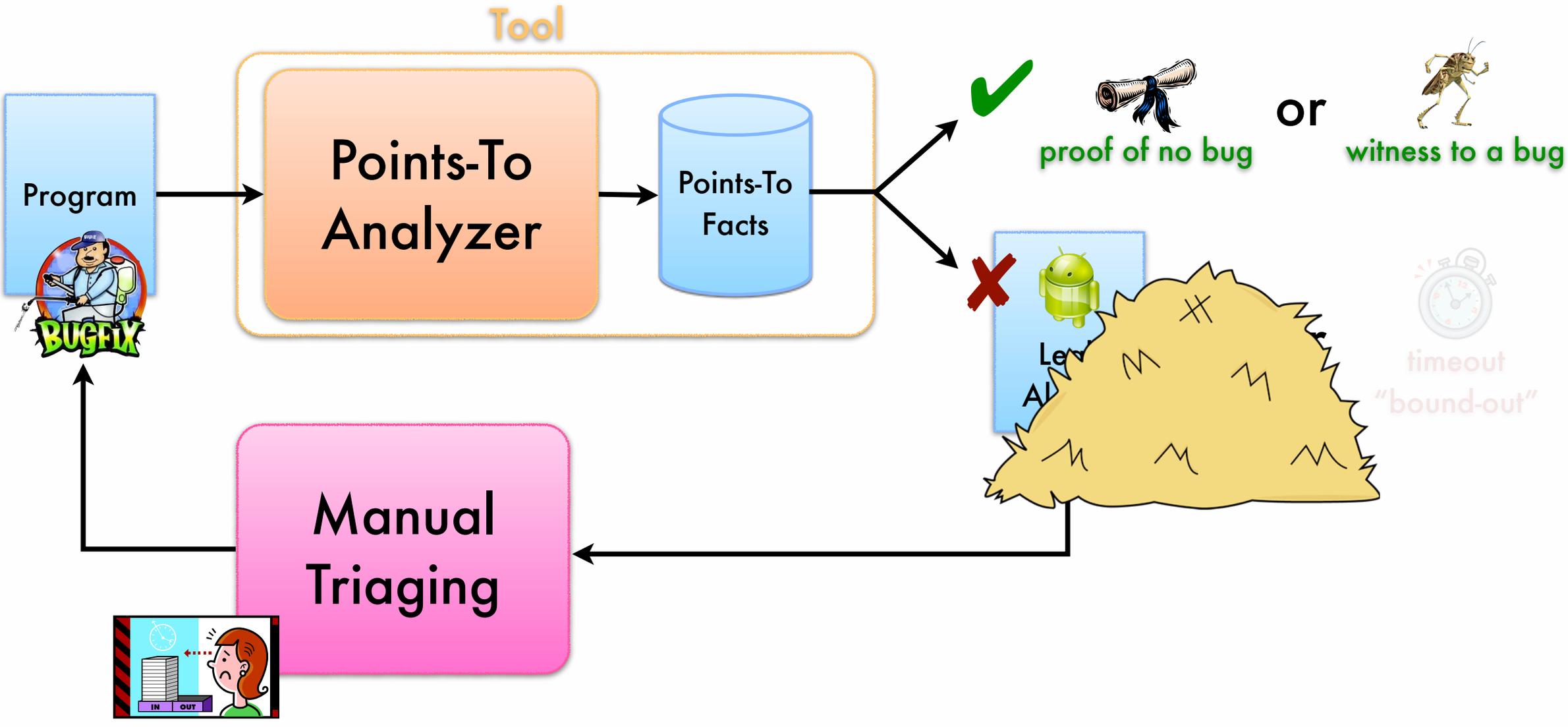
Thresher [SAS'11,PLDI'13] attacks alarm triage for heap reachability properties



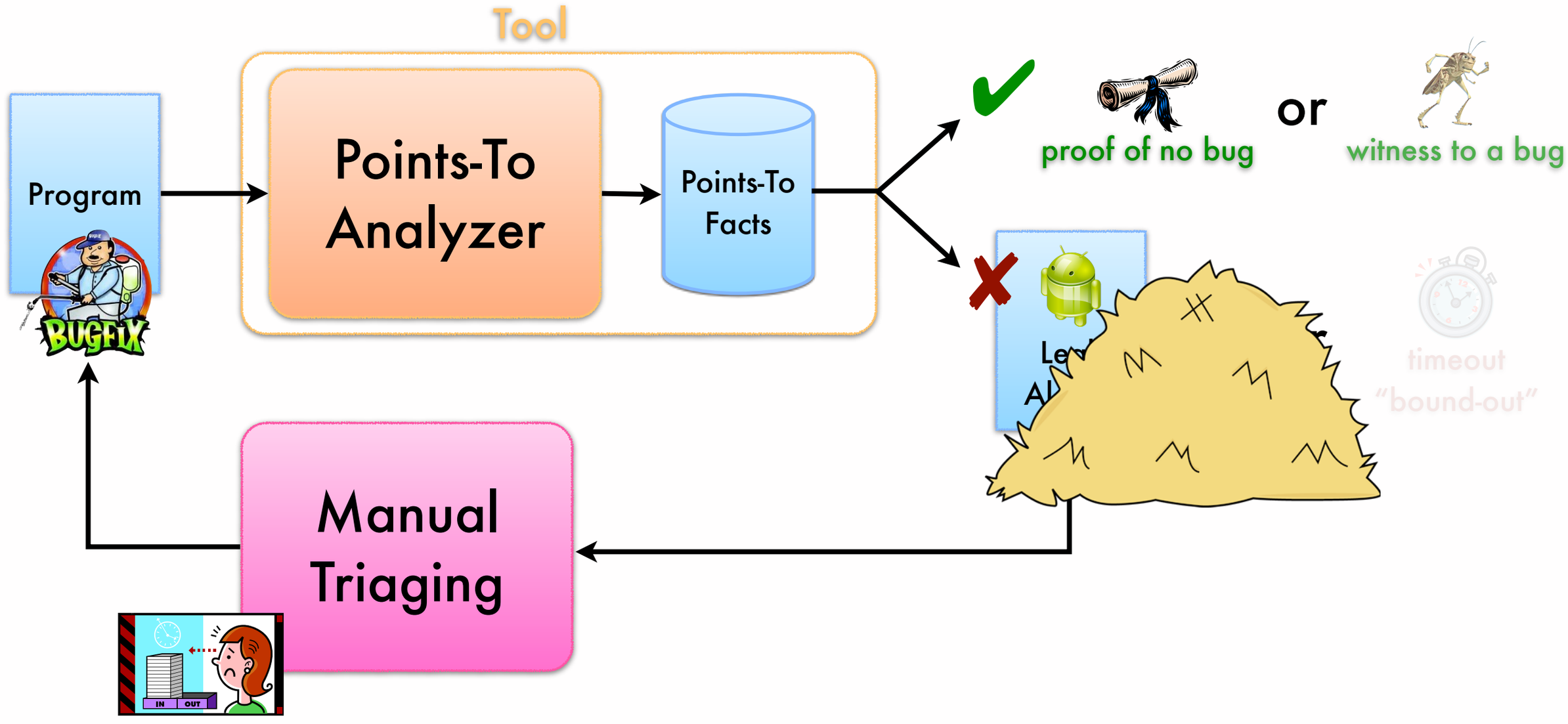
Thresher [SAS'11,PLDI'13] attacks alarm triage for heap reachability properties



Thresher [SAS'11,PLDI'13] attacks alarm triage for heap reachability properties



Thresher [SAS'11,PLDI'13] attacks alarm triage for heap reachability properties



Prove alarms false with a witness search

Thresher [SAS'11,PLDI'13] attacks alarm triage for heap reachability properties

Tool

Program

Points-To Analyzer

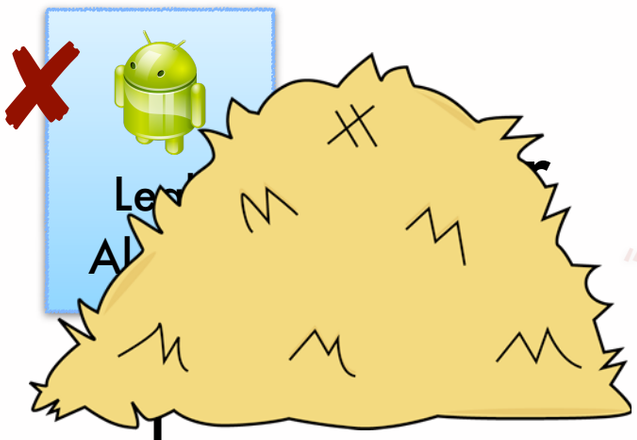
Points-To Facts

✔ proof of no bug

or

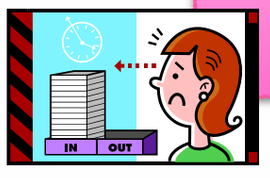
witness to a bug

✘



timeout "bound-out"

Manual Triaging

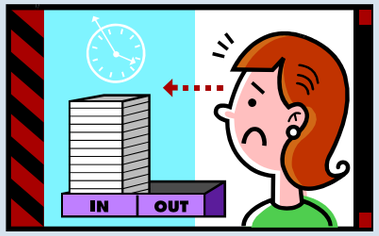


Proof Obligation User-Query
[Dillig, Dillig, Aiken (2012)]

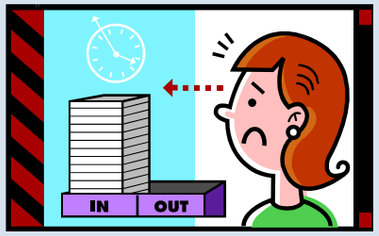
Alarm Clustering
[Lee, Lee, Yi (2012)]

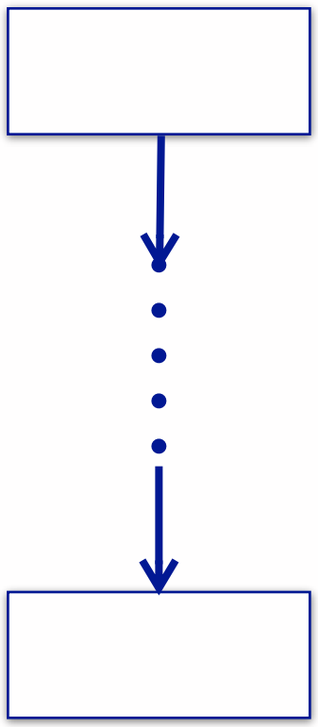
Prove alarms false with a witness search

Manual triage for heap reachability reports

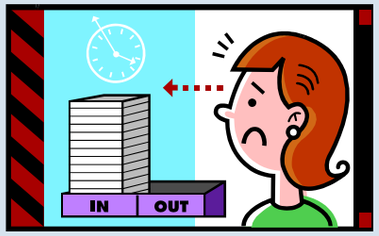


Manual triage for heap reachability reports

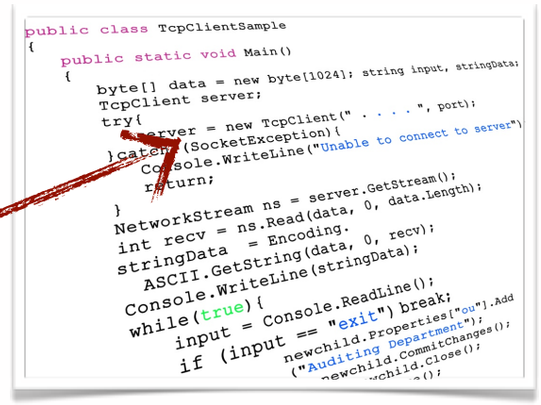




Manual triage for heap reachability reports

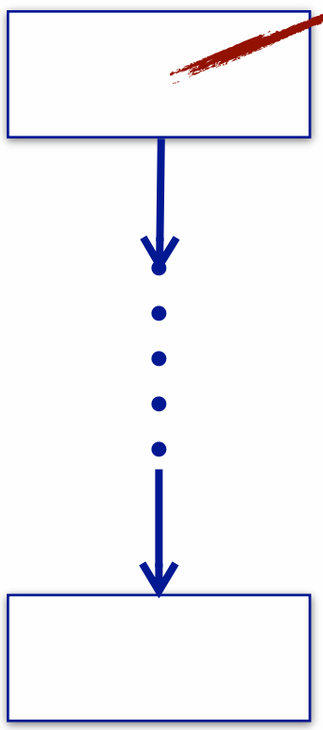


allocated here

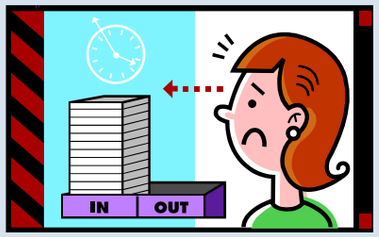


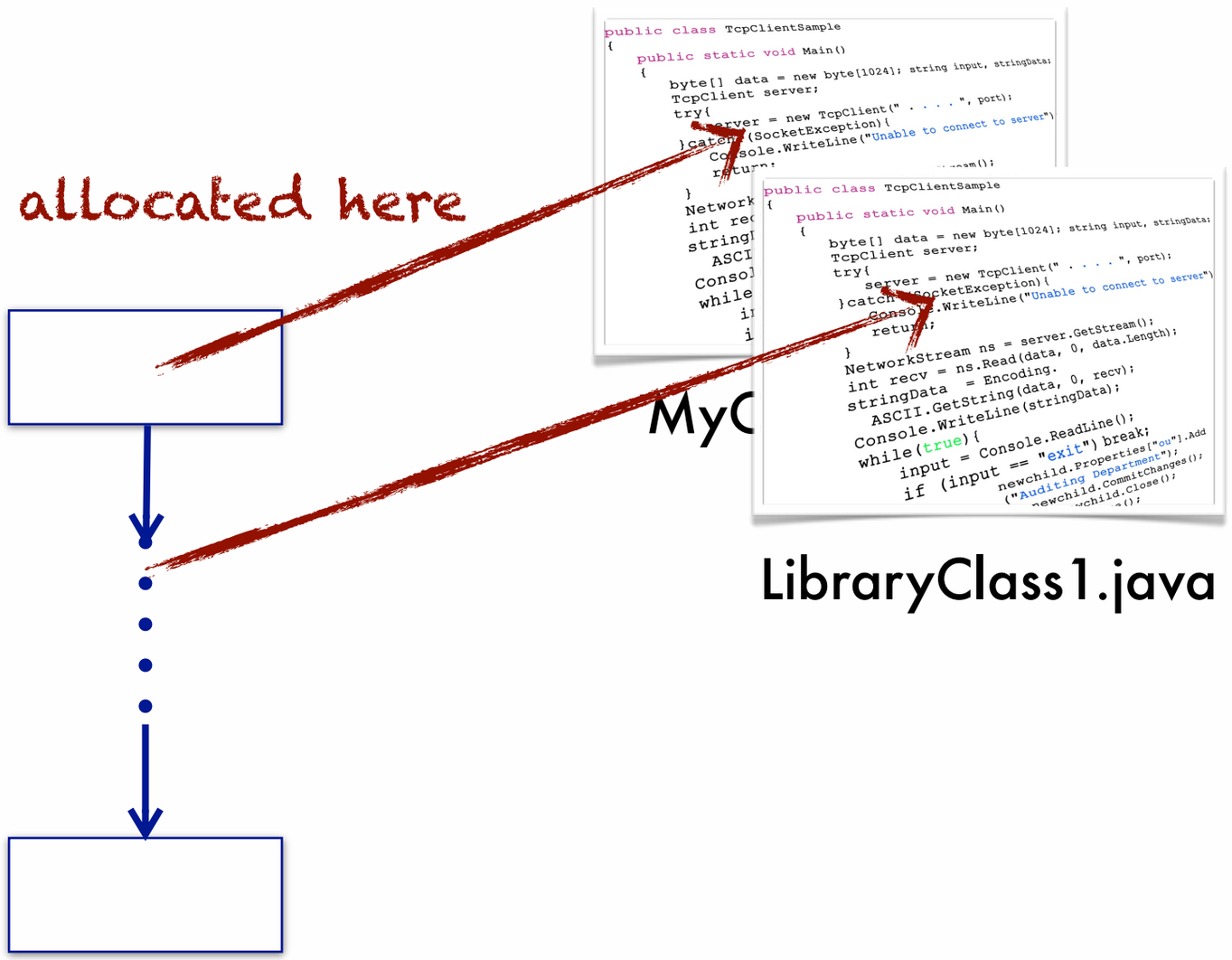
```
public class TcpClientSample
{
    public static void Main()
    {
        byte[] data = new byte[1024]; string input, stringData;
        TcpClient server;
        try{
          server = new TcpClient(" . . . . ", port);
        }catch(SocketException){
          Console.WriteLine("Unable to connect to server")
          return;
        }
        NetworkStream ns = server.GetStream();
        int recv = ns.Read(data, 0, data.Length);
        stringData = Encoding.
          ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
        while(true){
          input = Console.ReadLine();
          if (input == "exit") break;
                newchild.Properties["ou"].Add
                ("Auditing Department");
                newchild.CommitChanges();
                newchild.Close();
```

MyClass1.java

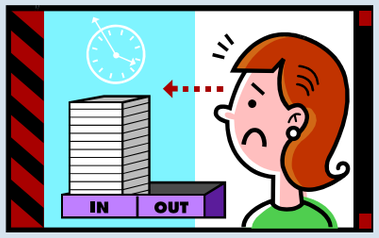


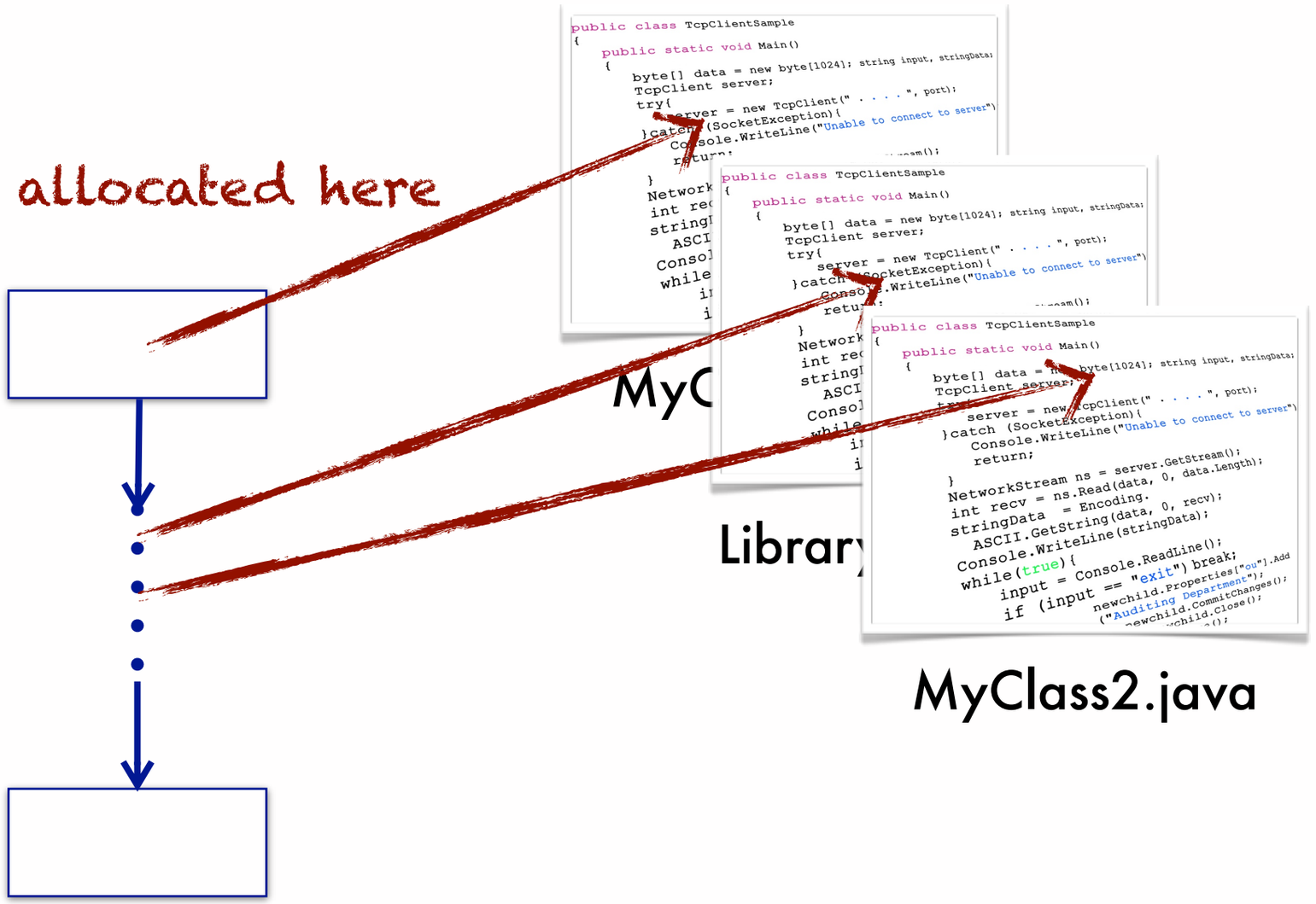
Manual triage for heap reachability reports



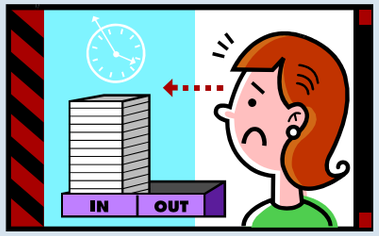


Manual triage for heap reachability reports

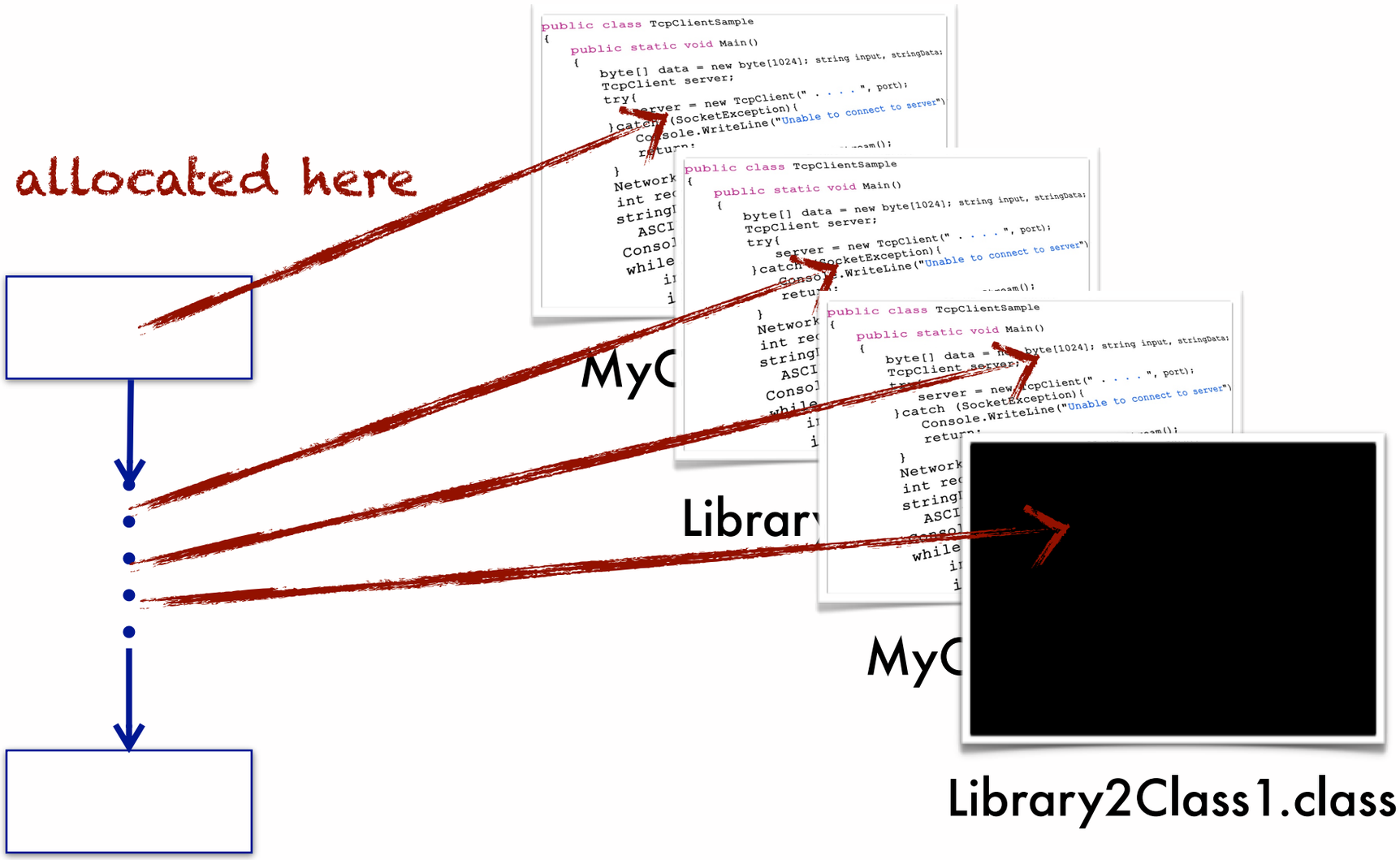




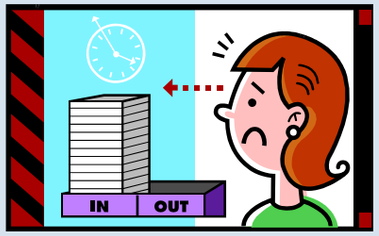
Manual triage for heap reachability reports



allocated here



Manual triage for heap reachability reports



allocated here

MyC

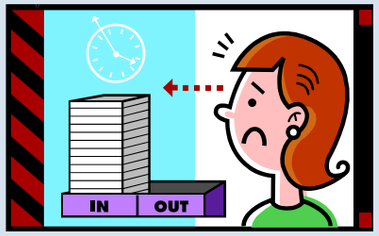
Library

MyC

Library

java.util.HashMap.class

Manual triage for heap reachability reports



allocated here

MyC

Library

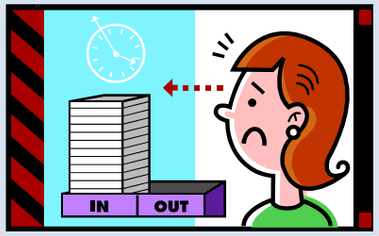
MyC

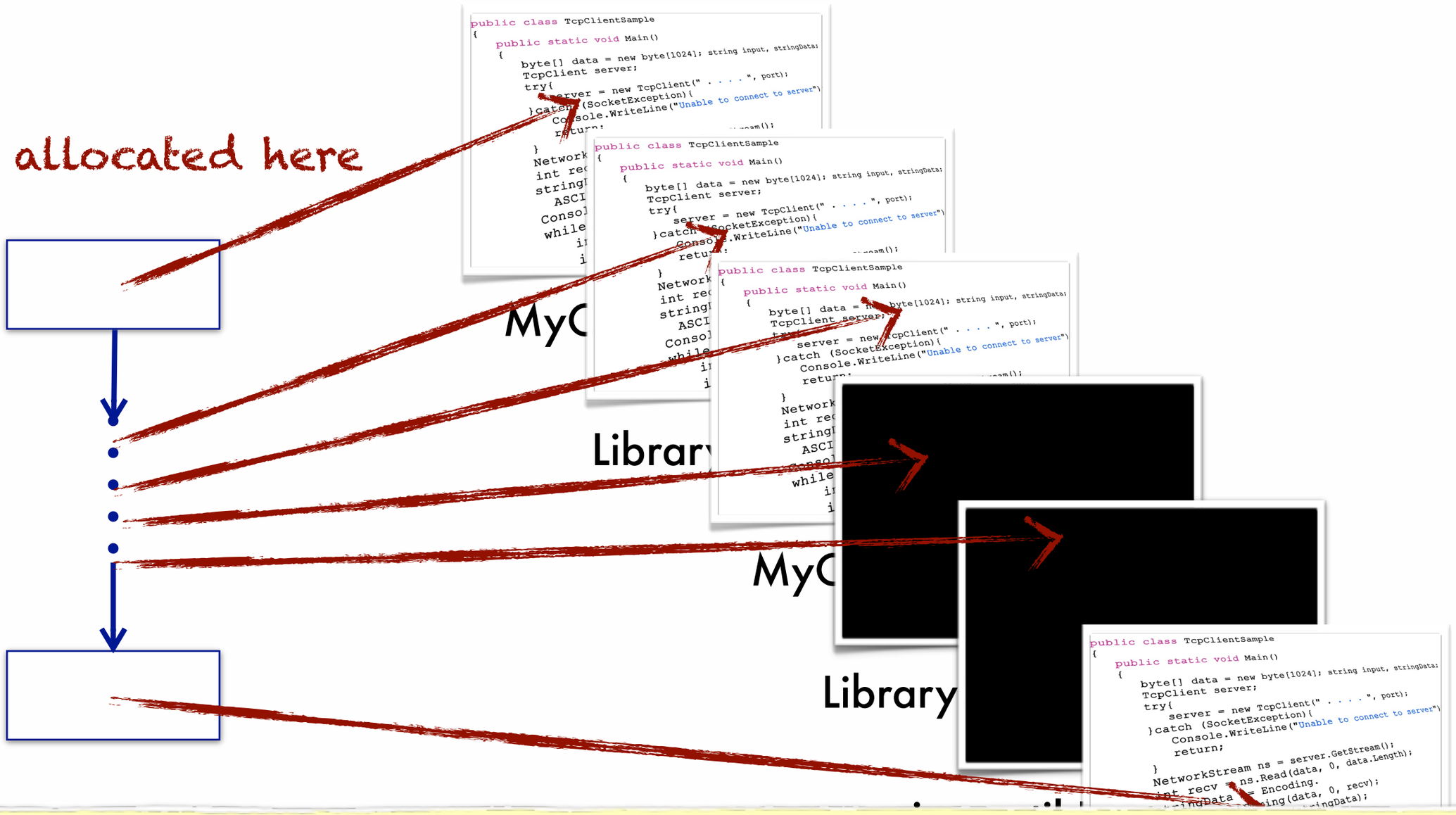
Library

java.util.

MyClass3.java

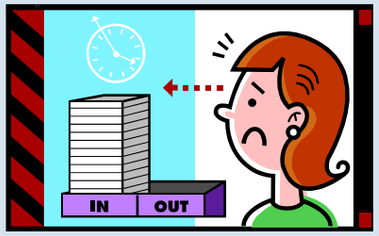
Manual triage for heap reachability reports



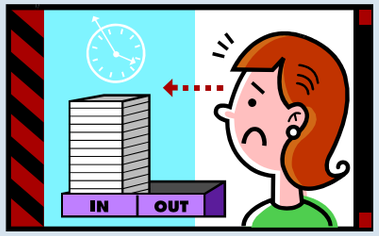


Get abstract heap path + allocation sites

Assuming the user starts to triage an alarm …

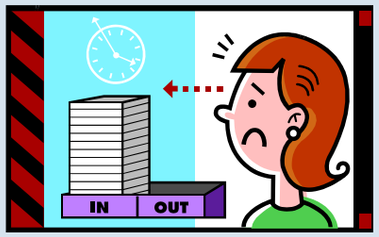


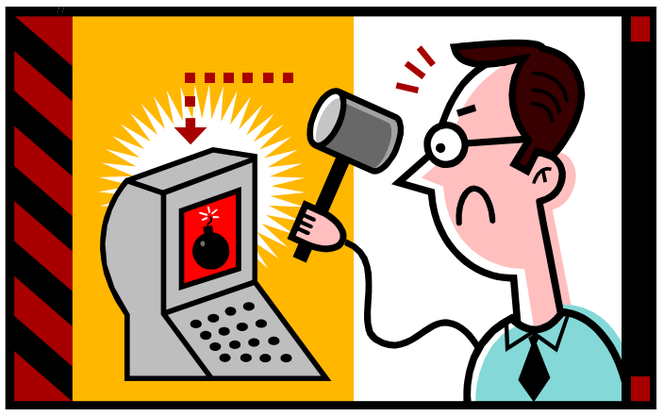
Assuming the user starts to triage an alarm …



What does the user need to do? He starts at, say, line 142 and traces back to see if a bug is possible given what's happening.

Assuming the user starts to triage an alarm …

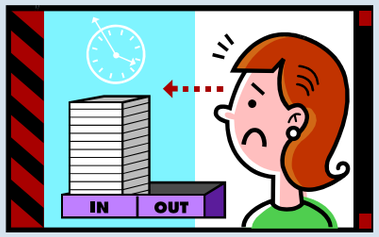


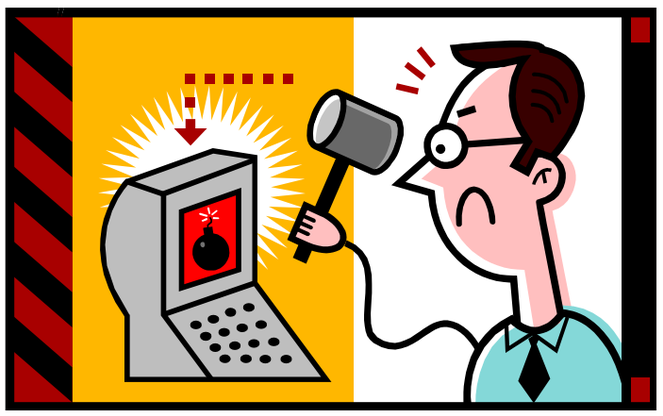


What does the user need to do? He starts at, say, line 142 and traces back to see if a bug is possible given what's happening.

We can do this with analysis!

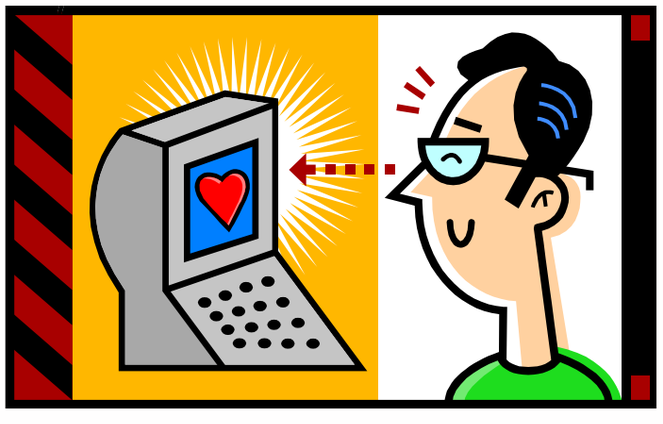
Assuming the user starts to triage an alarm …





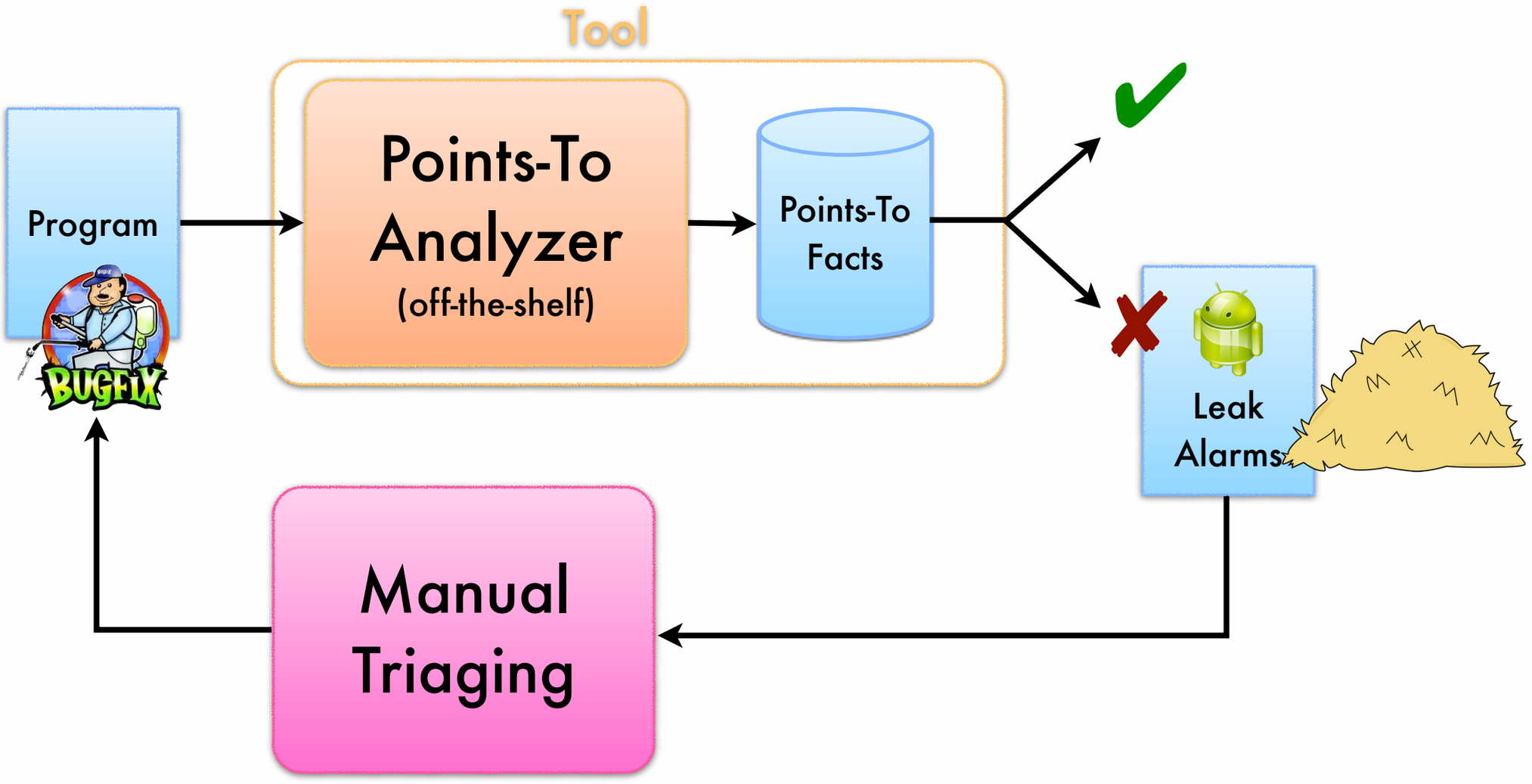
What does the user need to do? He starts at, say, line 142 and traces back to see if a bug is possible given what's happening.

We can do this with analysis!

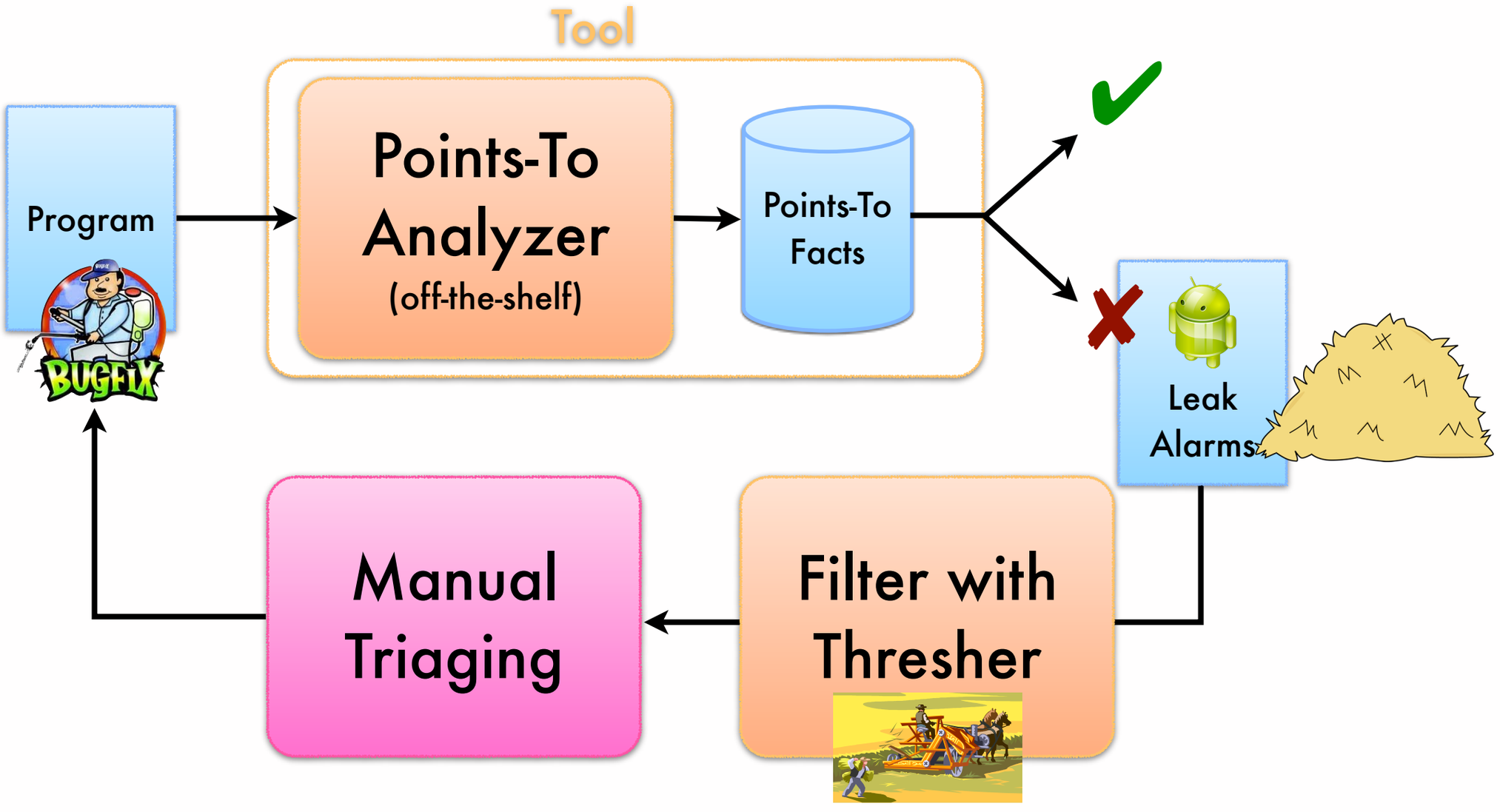


If we filter most false alarms, the user can triage more quickly and get to true bugs earlier (without frustration).

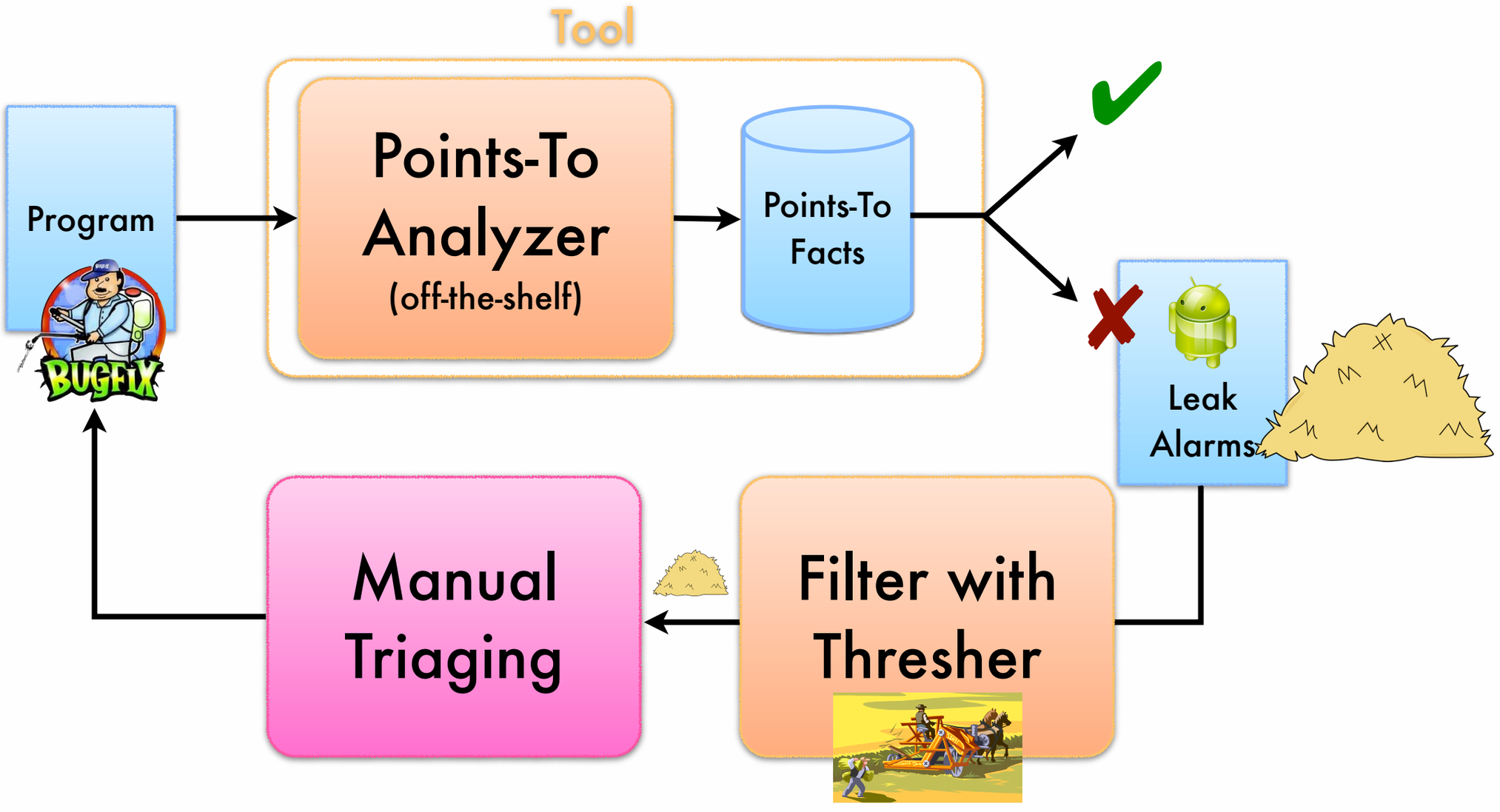
Roadmap: Thresher filters out false alarms by refuting them one-by-one.



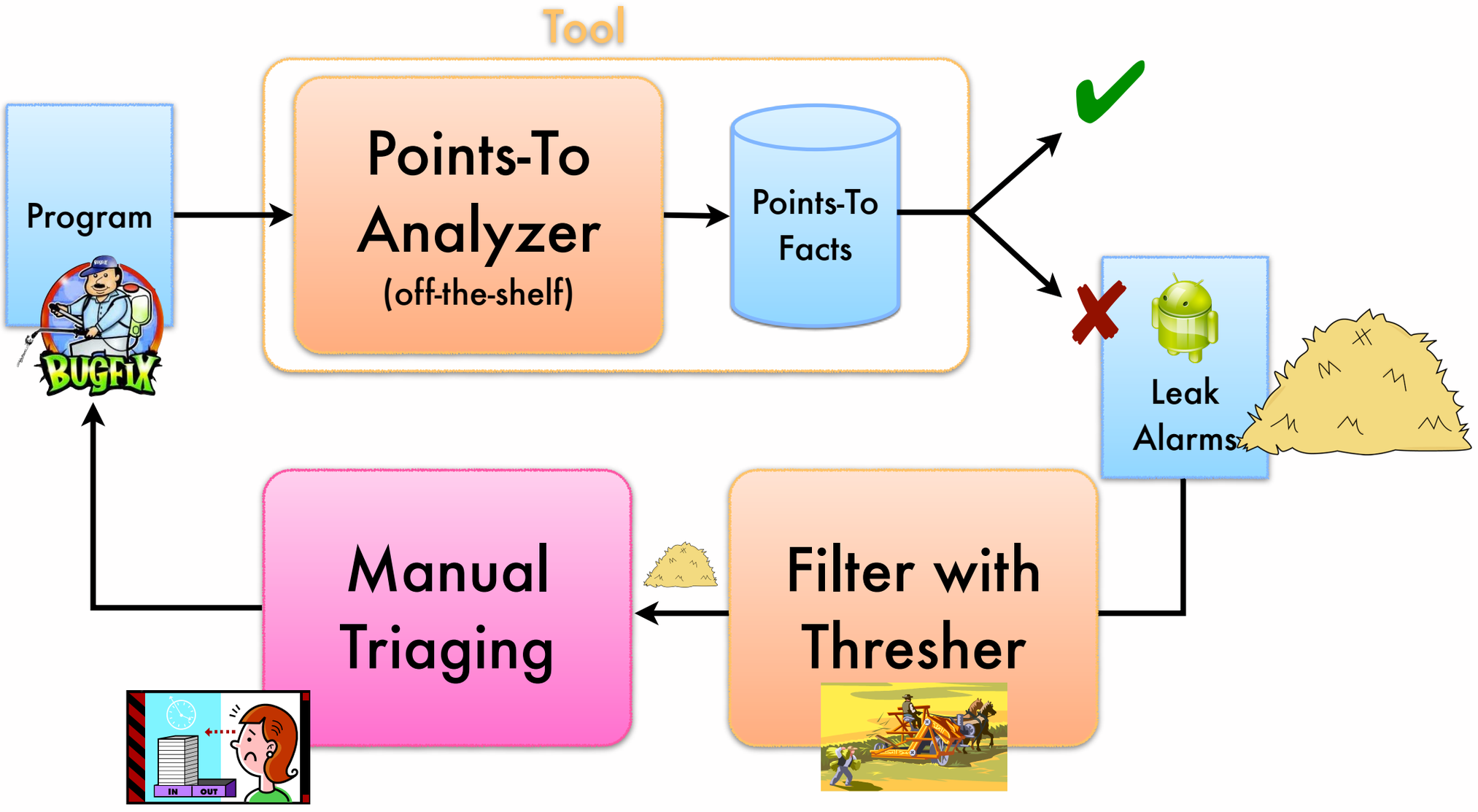
Roadmap: Thresher filters out false alarms by refuting them one-by-one.



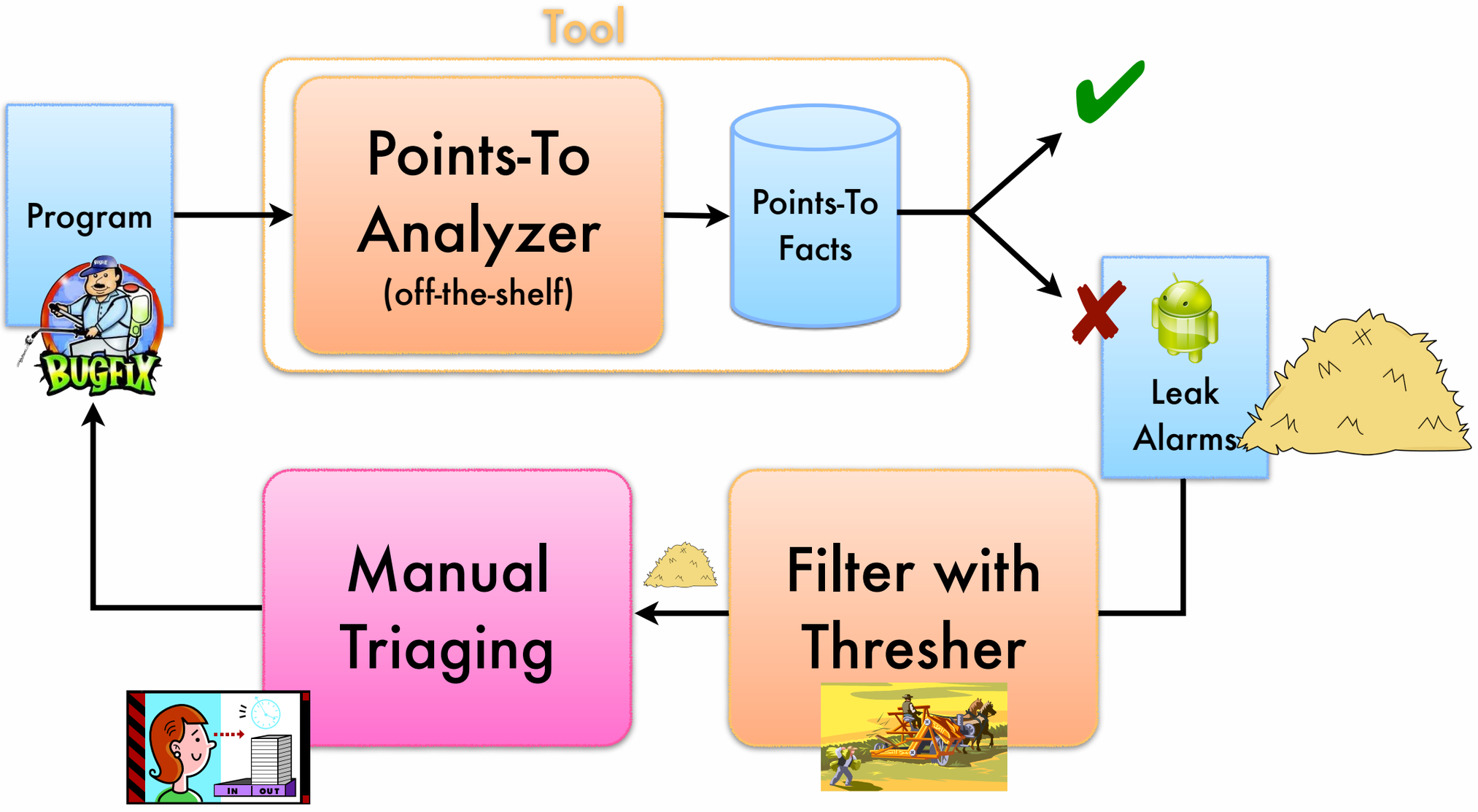
Roadmap: Thresher filters out false alarms by refuting them one-by-one.



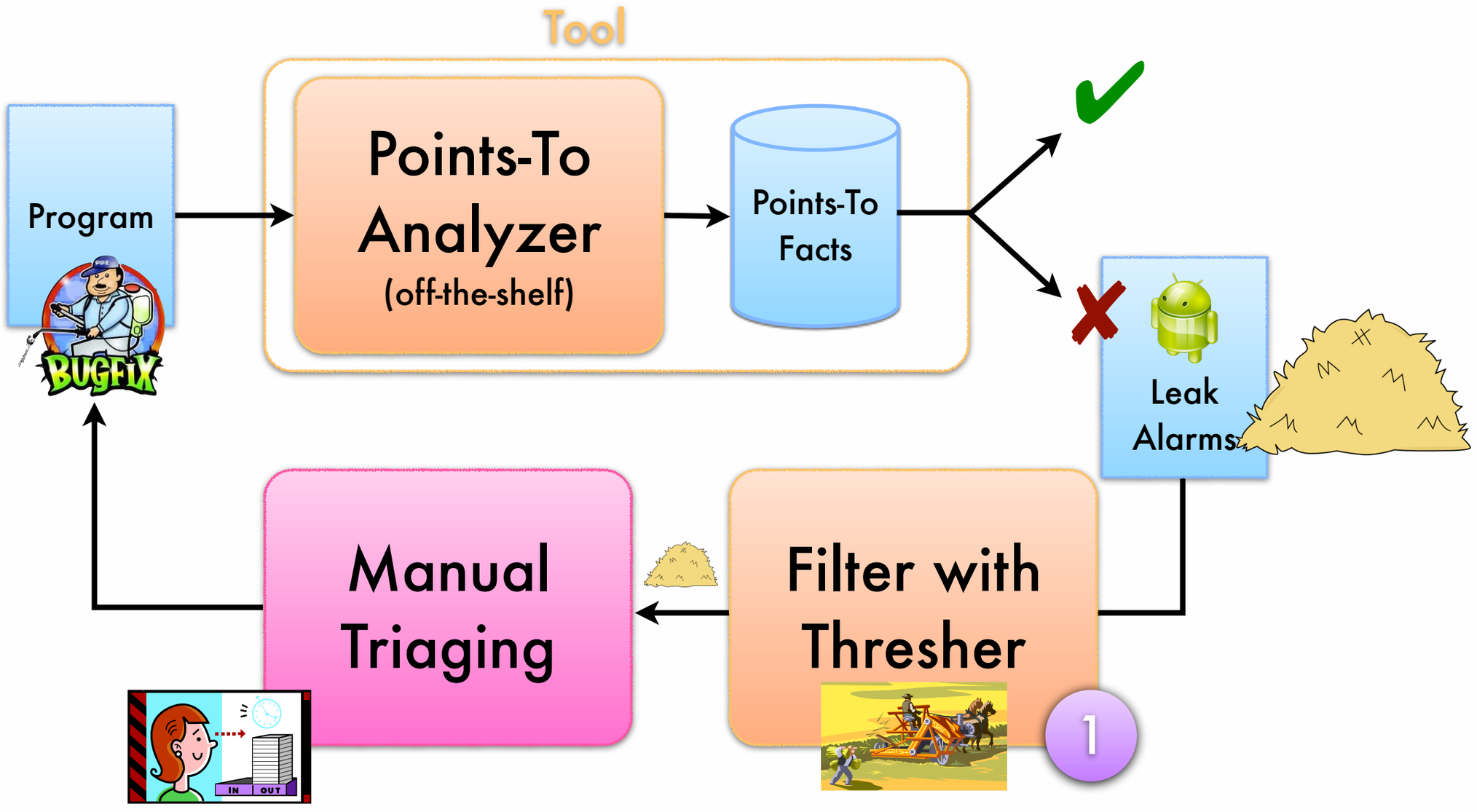
Roadmap: Thresher filters out false alarms by refuting them one-by-one.



Roadmap: Thresher filters out false alarms by refuting them one-by-one.

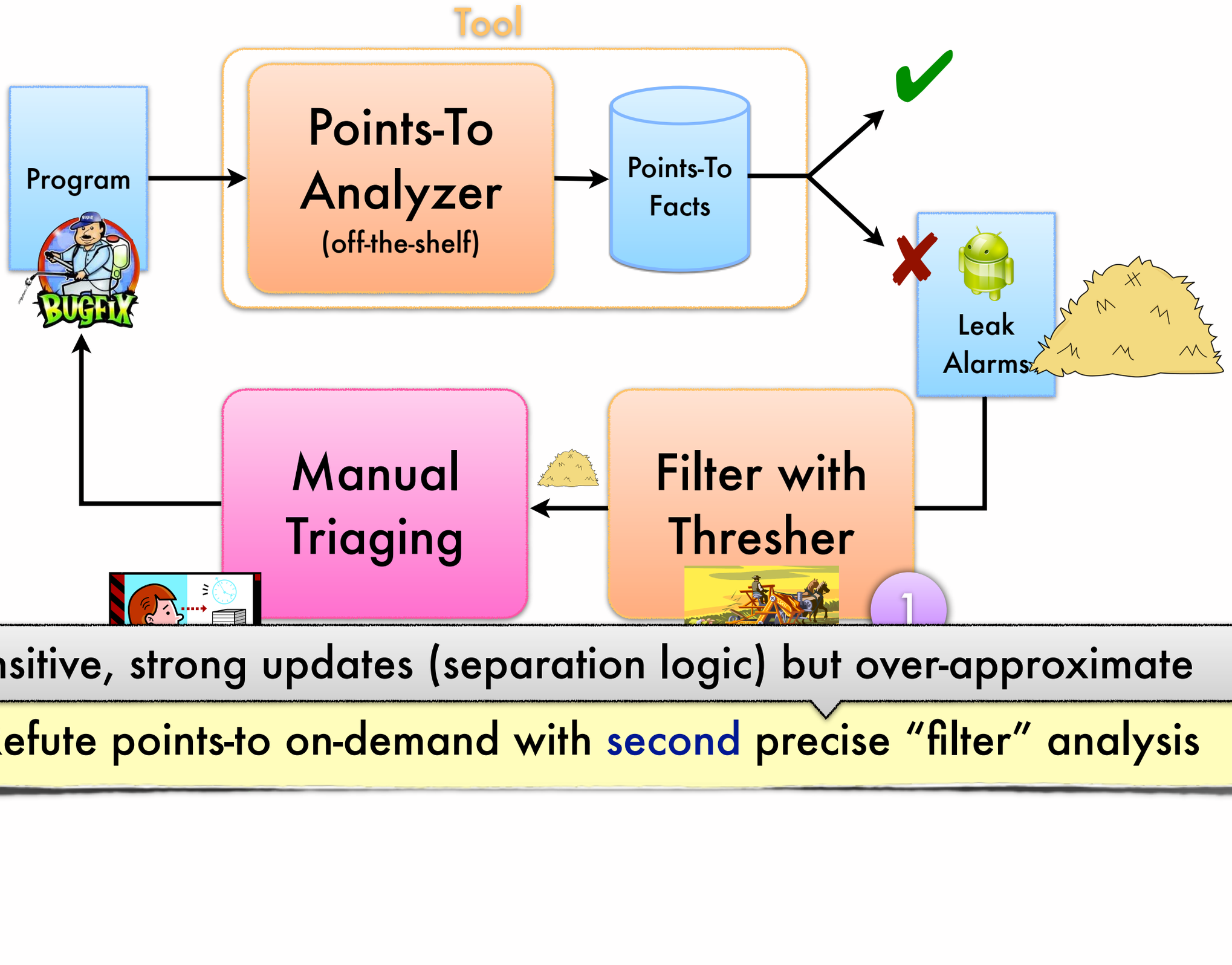


Roadmap: Thresher filters out false alarms by refuting them one-by-one.



Idea 1: Refute points-to on-demand with second precise "filter" analysis

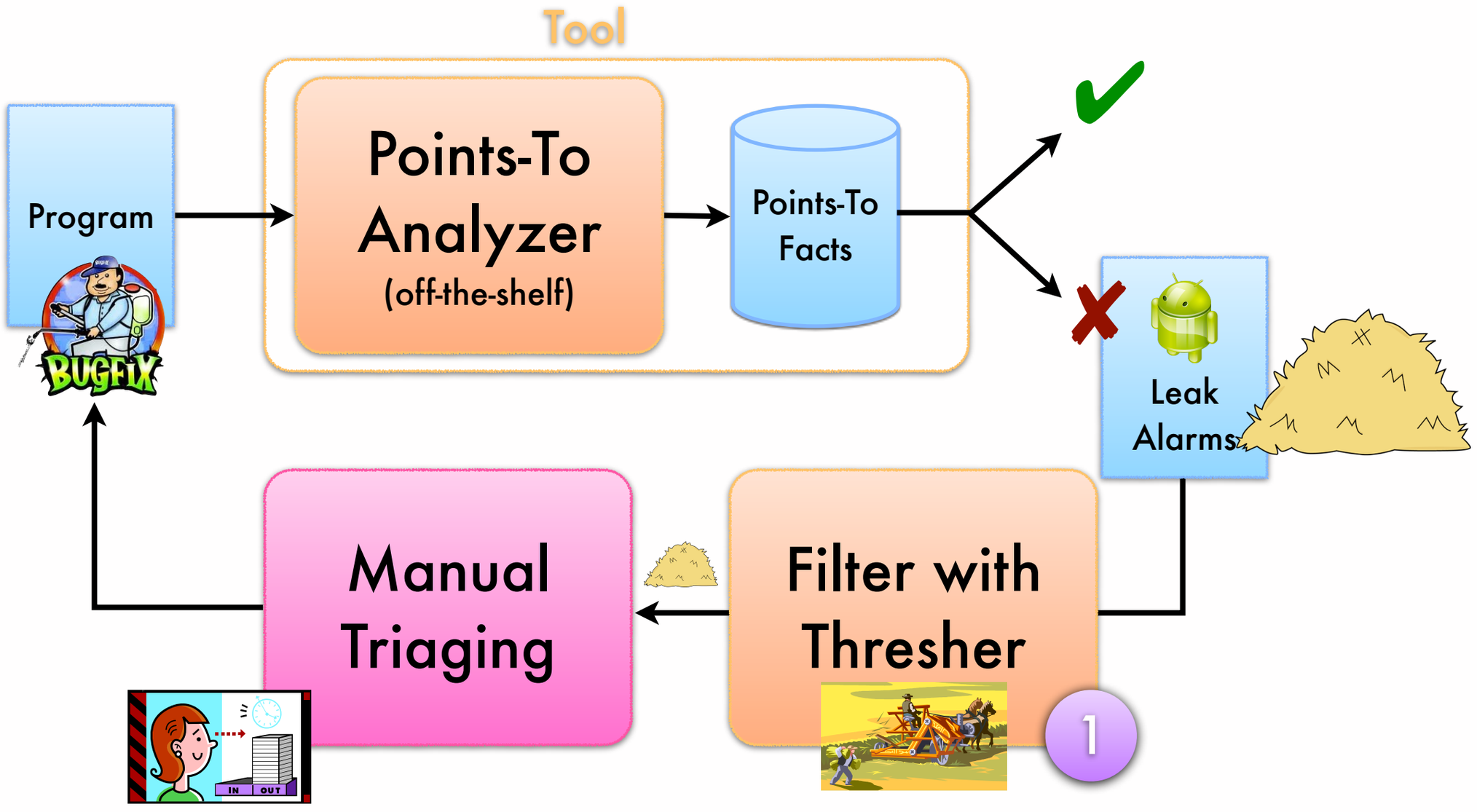
Roadmap: Thresher filters out false alarms by refuting them one-by-one.



*-sensitive, strong updates (separation logic) but over-approximate

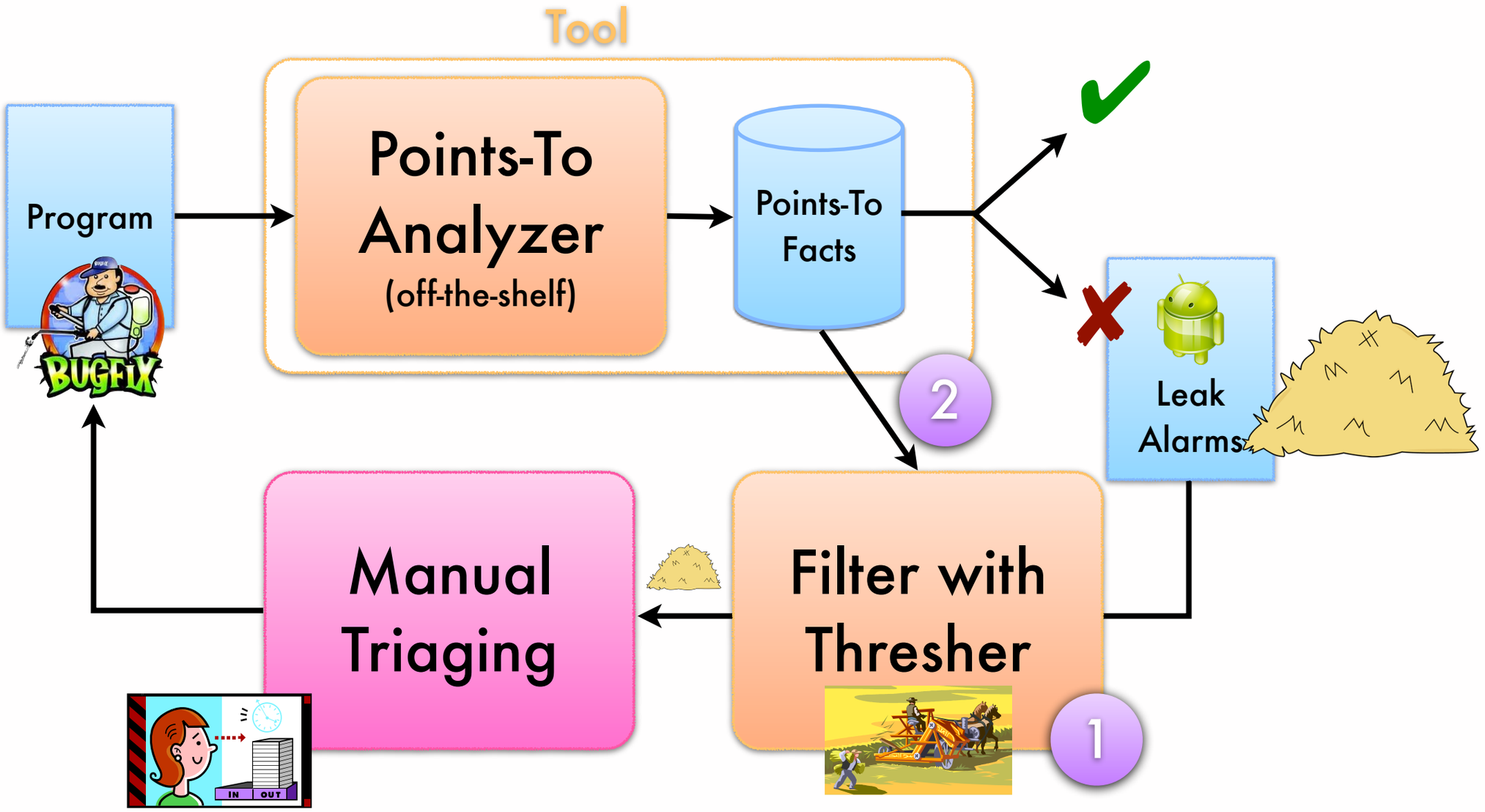
Idea 1: Refute points-to on-demand with second precise "filter" analysis

Roadmap: Thresher filters out false alarms by refuting them one-by-one.



Idea 1: Refute points-to on-demand with second precise "filter" analysis

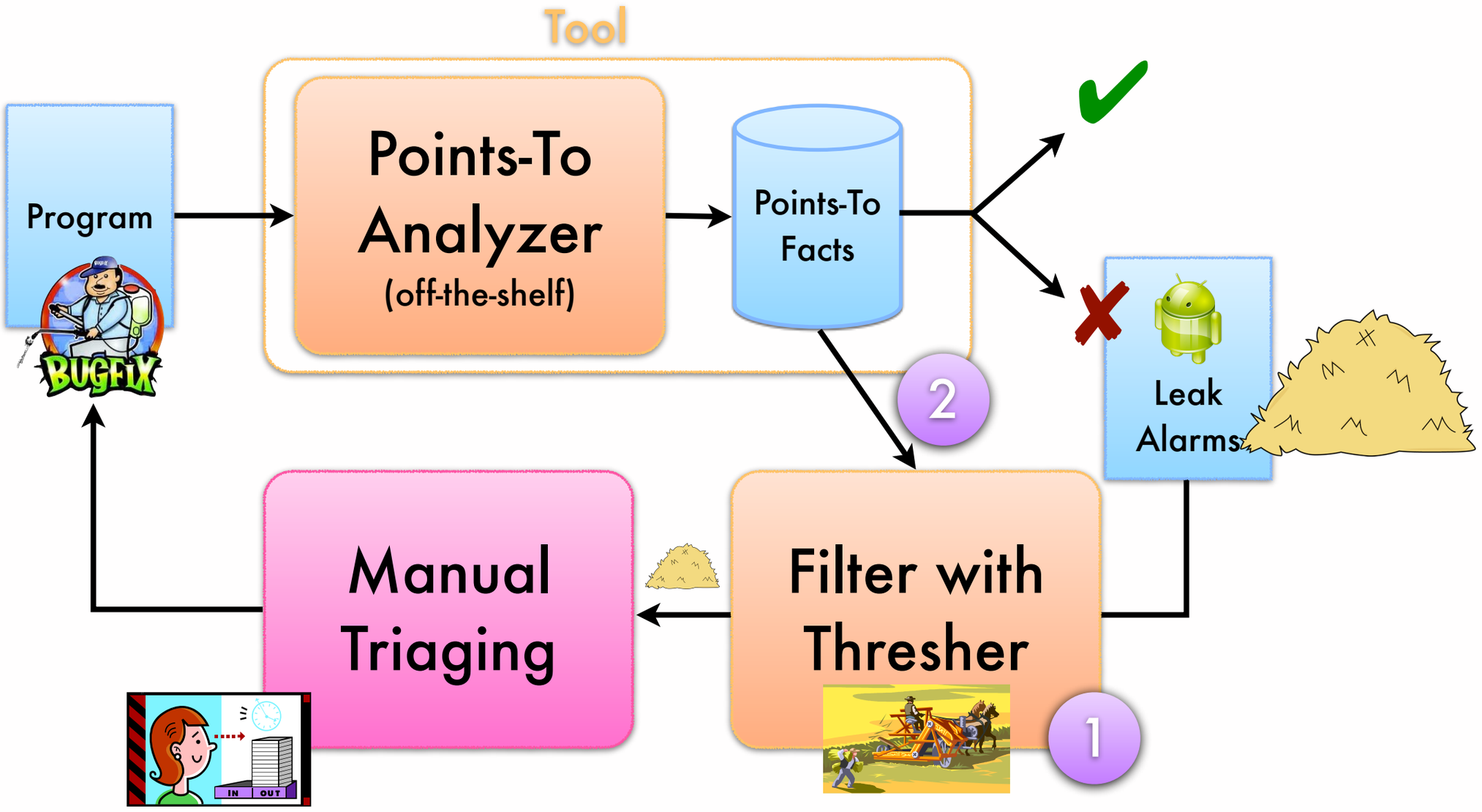
Roadmap: Thresher filters out false alarms by refuting them one-by-one.



Idea 1: Refute points-to on-demand with second precise "filter" analysis

Idea 2: Leverage the facts from the first analysis in the filter analysis to scale

Roadmap: Thresher filters out false alarms by refuting them one-by-one.

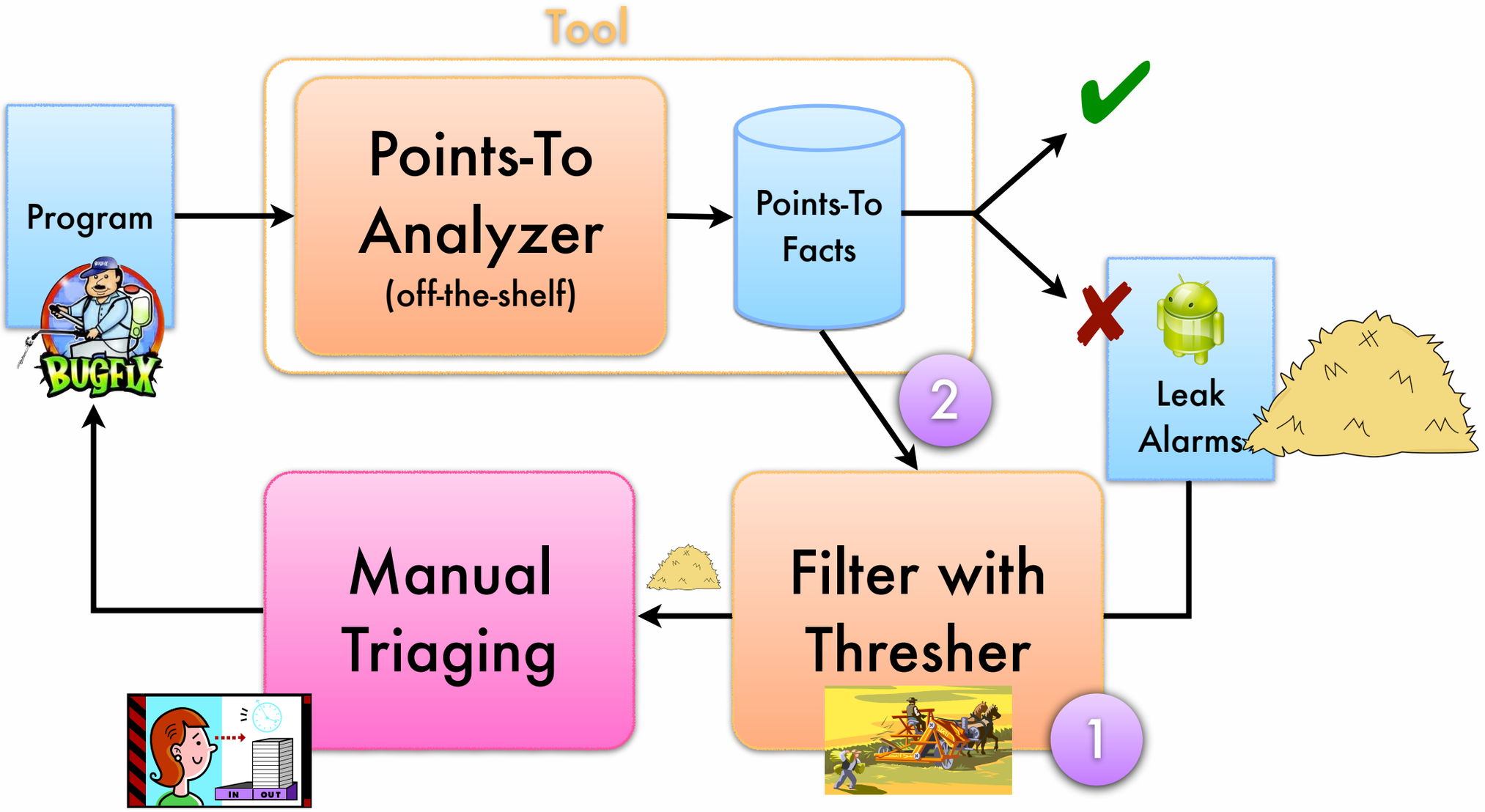


Idea 1: Refute points-to on-demand with second precise "filter" analysis

"**from** constraints" to reduce with the points-to domain

Idea 2: Leverage the facts from the first analysis in the filter analysis to scale

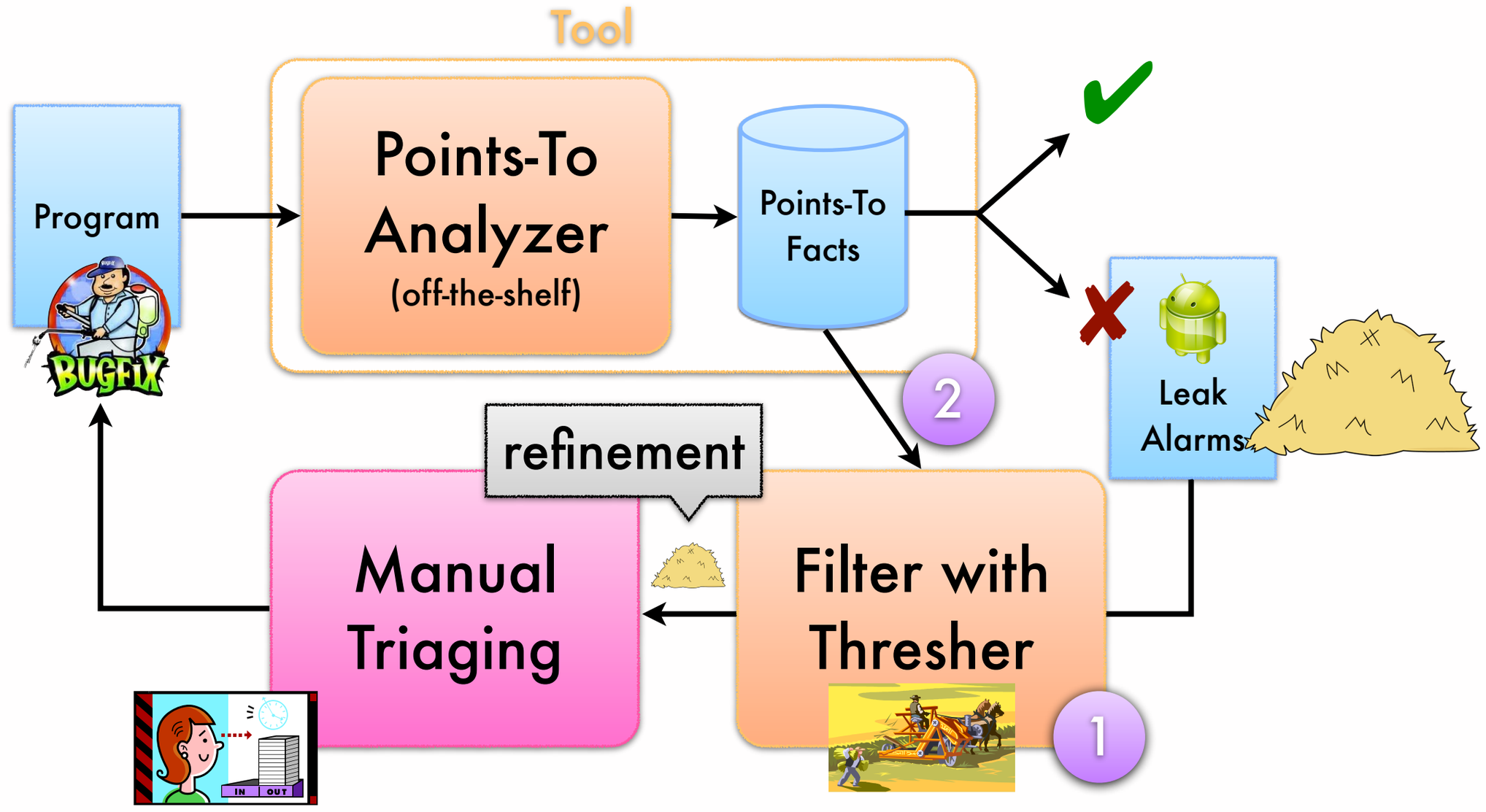
Roadmap: Thresher filters out false alarms by refuting them one-by-one.



Idea 1: Refute points-to on-demand with second precise "filter" analysis

Idea 2: Leverage the facts from the first analysis in the filter analysis to scale

Roadmap: Thresher filters out false alarms by refuting them one-by-one.

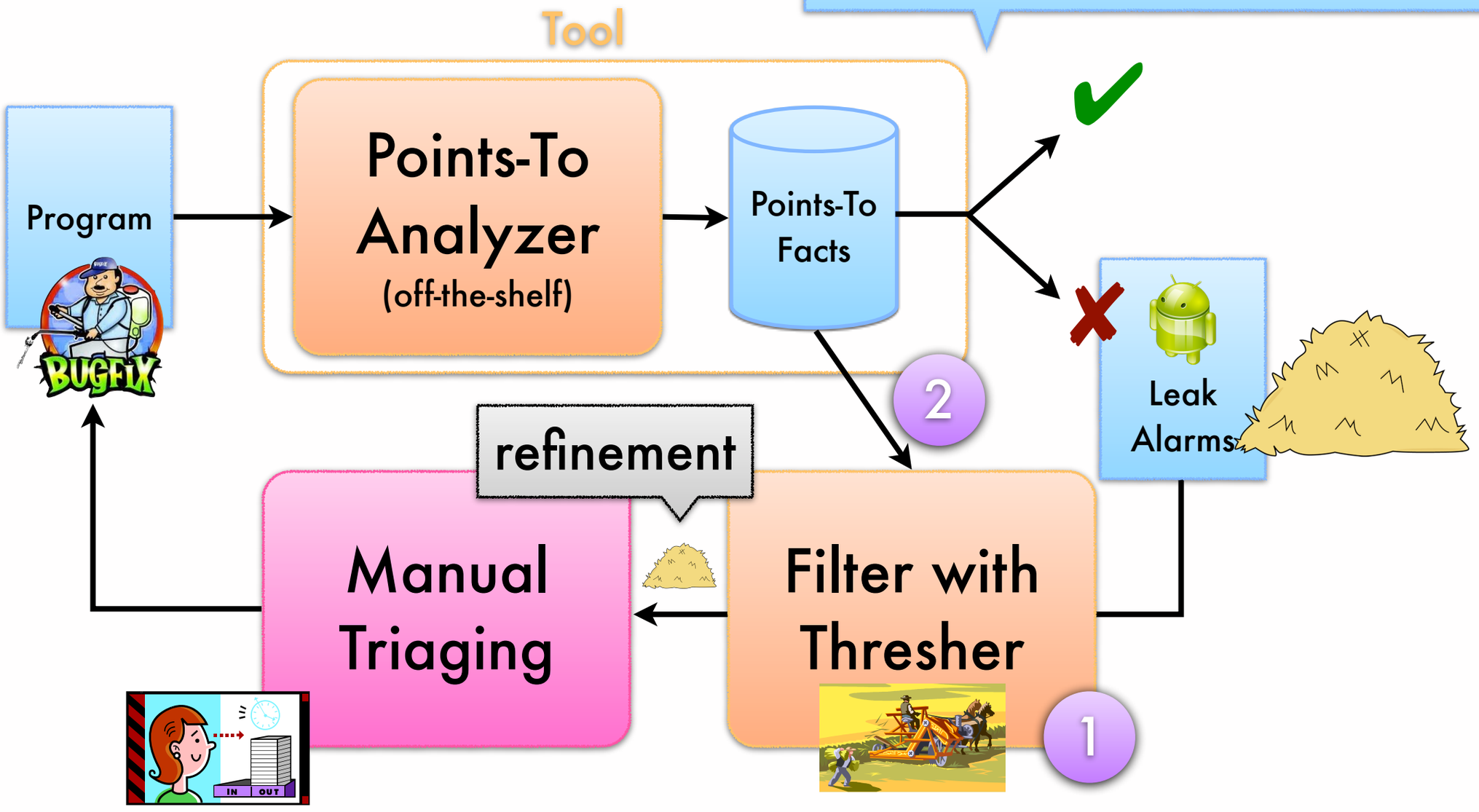


Idea 1: Refute points-to on-demand with second precise "filter" analysis

Idea 2: Leverage the facts from the first analysis in the filter analysis to scale

Roadmap: Thresher filters out false d… refuting them one-by-one.

Related: Staged analysis
[Fink, Yahav, Dor, Ramalingam, Geay (2006)]



Idea 1: Refute points-to on-demand with second precise "filter" analysis

Idea 2: Leverage the facts from the first analysis in the filter analysis to scale

A top-level filter:
Filter leak alarms by refuting points-to edges

Is there a program execution where at some time

`a_static_field`

of type `Activity` ?

A top-level filter:
Filter leak alarms by refuting points-to edges

Is there a program execution where at some time

`a_static_field`

of type `Activity` ?

Select a points-to edge in the path

A top-level filter:
Filter leak alarms by refuting points-to edges

Is there a program execution where at some time

`a_static_field`

↓

⋮

↓

of type `Activity` ?

Select a points-to edge in the path

↓

Try to refute the edge with a symbolic analysis

A top-level filter:
Filter leak alarms by refuting points-to edges

Is there a program execution where at some time

`a_static_field`

⋮

of type `Activity` ?

Select a points-to edge in the path

Try to refute the edge with a symbolic analysis

✔ Refuted

A top-level filter:
Filter leak alarms by refuting points-to edges

Is there a program execution where at some time

`a_static_field`

⋮

of type `Activity` ?

Select a points-to edge in the path

Try to refute the edge with a symbolic analysis

✔ Refuted

False Alarm
Soundly Filtered

A top-level filter:
Filter leak alarms by refuting points-to edges

Is there a program execution where at some time

`a_static_field`

⋮

of type `Activity` ?

Select a points-to edge in the path

↓

Try to refute the edge with a symbolic analysis

✔ Refuted → False Alarm Soundly Filtered

✘ Not Refuted

A top-level filter:
Filter leak alarms by refuting points-to edges

Is there a program execution where at some time

`a_static_field`

⋮

of type `Activity` ?

Select a points-to edge in the path

Try to refute the edge with a symbolic analysis

Repeat

✔ Refuted

False Alarm
Soundly Filtered

✘ Not Refuted

A top-level filter:
Filter leak alarms by refuting points-to edges

Is there a program execution where at some time

`a_static_field`

⋮

of type `Activity` ?

Select a points-to edge in the path

Try to refute the edge with a symbolic analysis

Repeat

✔ Refuted

False Alarm
Soundly Filtered

✘ Not Refuted

Refutation: Derive a contradiction, that a points-to relation can't actually hold

Refuting a points-to edge: What are we up against?


```
class Vec {
  static Object[] EMPTY = new_arr0 Object[1]; ...
  Vec() { this.tbl = EMPTY; capacity initially empty }
}
```

Refuting a points-to edge: What are we up against?

Null object pattern: Should never be written to

```
class Vec
  static Object[] EMPTY = new_arr0 Object[1]; ...
  Vec() { this.tbl = EMPTY; capacity initially empty }
}
```

Refuting a points-to edge: What are we up against?

Null object pattern: Should never be written to

```
class Vec
  static Object[] EMPTY = new_arr0 Object[1]; ...
  Vec() { this.tbl = EMPTY; capacity initially empty }

  void push(Object val) {
    if (need capacity) {
      this.tbl = new_arr1 Object[more capacity];
      copy from old table
    }
    this.tbl[next slot] = val;
  }
}
```

Refuting a points-to edge: What are we up against?

Null object pattern: Should never be written to

```
class Vec
  static Object[] EMPTY = new_arr0 Object[1]; ...
  Vec() { this.tbl = EMPTY; capacity initially empty }

  void push(Object val) {
    if (need capacity) {
      this.tbl = new_arr1 Object[more capacity];
      copy from old table
    }
    this.tbl[next slot] = val;
  }
}
```

Refuting a points-to edge: What are we up against?

Null object pattern: Should never be written to

```
class Vec
  static Object[] EMPTY = new_arr0 Object[1]; ...
  Vec() { this.tbl = EMPTY; capacity initially empty }

  void push(Object val) {
    if (need capacity) {
      this.tbl = new_arr1 Object[more capacity];
      copy from old table
    }
    this.tbl[next slot] = val;
  }
}
```

Refuting a points-to edge: What are we up against?

Null object pattern: Should never be written to

```
class Vec
  static Object[] EMPTY = new_arr0 Object[1]; ...
  Vec() { this.tbl = EMPTY; capacity initially empty }

  void push(Object val) {
    if (need capacity) {
      this.tbl = new_arr1 Object[more capacity];
      copy from old table
    }
    this.tbl[next slot] = val;
  }
}
```

Refuting a points-to edge: What are we up against?


```
class Vec
  static Object[] EMPTY = new_arr0 Object[1]; ...
  Vec() { this.tbl = EMPTY; capacity initially empty }

  void push(Object
    if (need capacity) {
      this.tbl = new_arr1 Object[more capacity];
      copy from old table
    }
    this.tbl[next slot] = val;
  }
}
```

Null object pattern: Should never be written to

Need interprocedural path-sensitivity

Refuting a points-to edge: What are we up against?

Null object pattern: Should never be written to

```
class Vec
  static Object[] EMPTY = new_arr0 Object[1]; ...
  Vec() { this.tbl = EMPTY; capacity initially empty }

  void push(Object
    if (need capacity) {
      this.tbl = new_arr1 Object[more capacity];
      copy from old table
    }
    this.tbl[next slot] = val;
  }
}
```

Need interprocedural path-sensitivity

Refuting a points-to edge: What are we up against?

Null object pattern: Should never be written to

```
class Vec
  static Object[] EMPTY = new_arr0 Object[1]; ...
  Vec() { this.tbl = EMPTY; capacity initially empty }

  void push(Object
    if (need capacity) {
      this.tbl = new_arr1 Object[more capacity];
      copy from old table
    }
    this.tbl[next slot] = val;
  }
}
```

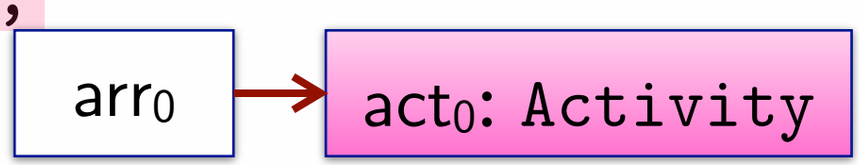
Need interprocedural path-sensitivity

Need strong updates

A refutation analysis: Try to derive a contradiction with a backwards symbolic analysis


```
class Vec {
  static Object[] EMPTY = new_arr0 Object[1]; ...
  Vec() { this.tbl = EMPTY; capacity initially empty }

  void push(Object val) {
    if (need capacity) {
      this.tbl = new_arr1 Object[more capacity];
      copy from old table
    }
    this.tbl[next slot] = val;
  }
}
```



A refutation analysis: Try to derive a contradiction with a backwards symbolic analysis


```
class Vec {
  static Object[] EMPTY = new_arr0 Object[1]; ...
  Vec() { this.tbl = EMPTY; capacity initially empty }

  void push(Object val) {
    if (need capacity) {
      this.tbl = new_arr1 Object[more capacity];
      copy from old table
    }
    this.tbl[next slot] = val;
  }
}
```

$arr_0 \cdot [-] \mapsto act_0 * \text{true}$

A refutation analysis: Try to derive a contradiction with a backwards symbolic analysis


```
class Vec {
  static Object[] EMPTY = new_arr0 Object[1]; ...
  Vec() { this.tbl = EMPTY; capacity initially empty }

  void push(Object val) {
    if (need capacity) {
      this.tbl = new_arr1 Object[more capacity];
      copy from old table
    }
    this.tbl[next slot] = val;
  }
}
```

$arr_0 \cdot [-] \mapsto act_0 * \text{true}$

A refutation analysis: Try to derive a contradiction with a backwards symbolic analysis


```
class Vec {
  static Object[] EMPTY = new Object[1]; ...
  Vec() { this.tbl = EMPTY
  void push(Object val) {
    if (need capacity) {
      this.tbl = new Obj
      copy from old table
    }
    this.tbl[next slot] = val;
  }
}
```

Derive a contradiction along all "backwards" path programs

[Beyer, Henzinger, Majumdar, Rybalchenko (2007)]

arr_o·[-] ↦ act_o * true

A refutation analysis: Try to derive a contradiction with a backwards symbolic analysis

false false false false false false

```
class Vec {
  static Object[] EMPTY = new Object[1]; ...
  Vec() { this.tbl = EMPTY
  void push(Object val) {
    if (need capacity) {
      this.tbl = new Obj
      copy from old table
    }
    this.tbl[next slot] = val;
  }
}
```

Derive a contradiction along all "backwards" path programs

[Beyer, Henzinger, Majumdar, Rybalchenko (2007)]

$arr_o \cdot [-] \mapsto act_o * \text{true}$

A refutation analysis: Try to derive a contradiction with a backwards symbolic analysis

false false false false false false

```
class Vec {
  static Object[] EMPTY = new Object[1]; ...
  Vec() { this.tbl = EMPTY
  void push(Object val) {
    if (need capacity) {
      this.tbl = new Obj
      copy from old table
    }
    this.tbl[next slot] = val;
  }
}
```

Derive a contradiction along all "backwards" path programs

[Beyer, Henzinger, Majumdar, Rybalchenko (2007)]

$arr_o \cdot [-] \mapsto act_o * \text{true}$

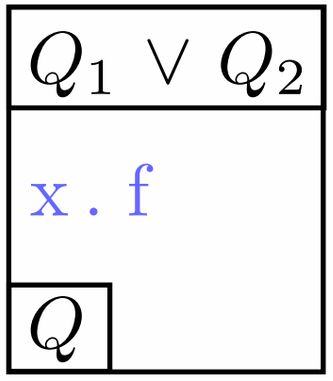
Derive refutations by trying to find witnesses

Roadmap: Precise but with scalability challenges

1

Roadmap: Precise but with scalability challenges

1

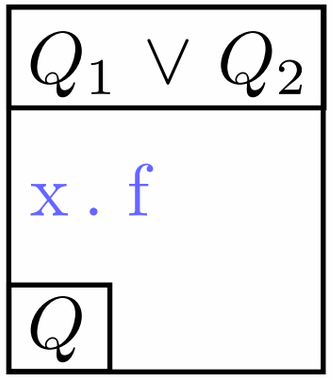


Alias path explosion for strong updates

(On write, case split for each possible alias in Q to maintain separation)

Roadmap: Precise but with scalability challenges

1



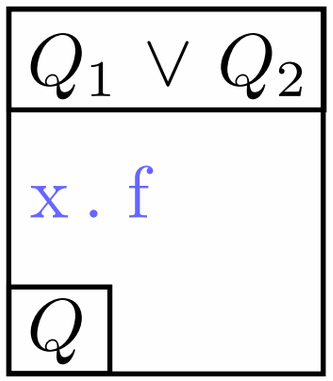
Alias path explosion for strong updates

(On write, case split for each possible alias in Q to maintain separation)

2

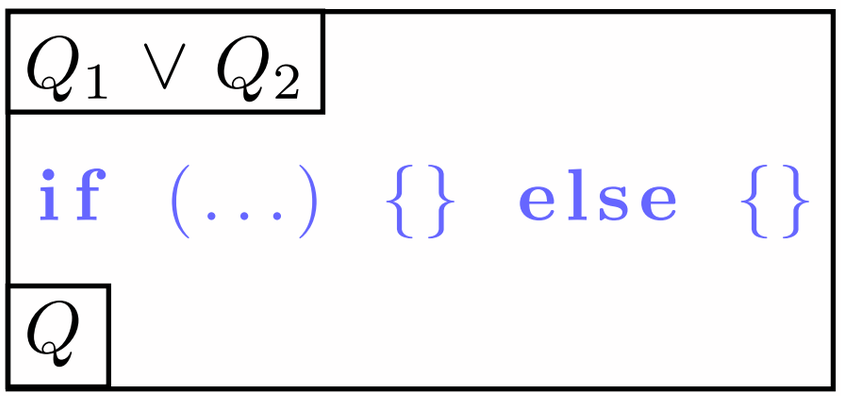
Roadmap: Precise but with scalability challenges

1



Alias path explosion for strong updates
(On write, case split for each possible alias in Q to maintain separation)

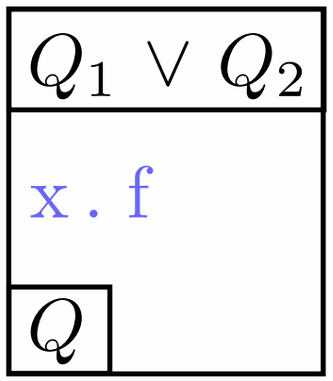
2



Control-flow path explosion:
Ignore for now, reasonable if number of guards relevant to Q is small (e.g., [Das et al. (2002)])

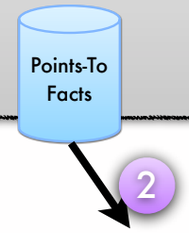
Roadmap: Precise but with scalability challenges

1

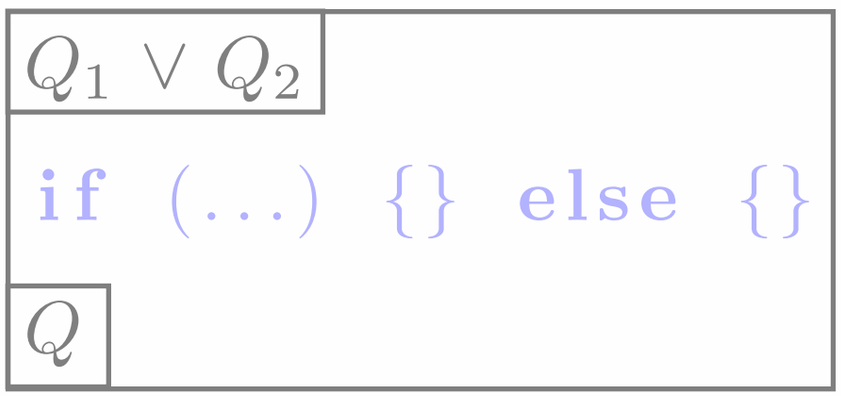


Alias path explosion for strong updates

(On write, case split for each possible alias in Q to maintain separation)



2



Control-flow path explosion:

Ignore for now, reasonable if number of guards relevant to Q is small (e.g., [Das et al. (2002)])

Roadmap: Precise but with scalability challenges

1

$Q_1 \vee Q_2$

x . f

Q

Alias path explosion for strong updates
(On write, case split for each possible alias in Q to maintain separation)

2

$Q_1 \vee Q_2$

if (...) {} else {}

Q

Control-flow path explosion:
Ignore for now, reasonable if number of guards relevant to Q is small (e.g., [Das et al. (2002)])

while Q_{inv} (...) {}

Q

Loops:
Simple loop invariant inference sufficient so far but more sophisticated techniques possible if needed

Roadmap: Precise but with scalability challenges

1

$Q_1 \vee Q_2$

x . f

Q

Alias path explosion for strong updates
(On write, case split for each possible alias in Q to maintain separation)

Points-To Facts

2

$Q_1 \vee Q_2$

if (...) {} else {}

Q

Control-flow path explosion:
Ignore for now, reasonable if number of guards relevant to Q is small (e.g., [Das et al. (2002)])

while Q_{inv} (...) {}

Q

Loops:
Simple loop invariant inference sufficient so far but more sophisticated techniques possible if needed

Roadmap: Precise but with scalability challenges

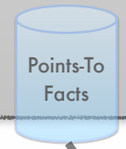
1

$Q_1 \vee Q_2$

x . f

Q

Alias path explosion for strong updates
(On write, case split for each possible alias in Q to maintain separation)



2

$Q_1 \vee Q_2$

if (...) {} else {}

Q

Control-flow path explosion:
Ignore for now, reasonable if number of guards relevant to Q is small (e.g., [Das et al. (2002)])

while Q_{inv} (...) {}

Q

Over-approximate what?

Loops:
Simple loop invariant inference sufficient so far but more sophisticated techniques possible if needed

Soundness Criteria

1

Soundness Criteria

Concrete Evaluation $\langle \sigma, s \rangle \Downarrow \sigma'$

Soundness Criteria

Concrete Evaluation $\langle \sigma, s \rangle \Downarrow \sigma'$

$\sigma \in \mathbf{State} \quad s \in \mathbf{Statement}$

Soundness Criteria

Concrete Evaluation $\langle \sigma, s \rangle \Downarrow \sigma'$

$\sigma \in \mathbf{State} \quad s \in \mathbf{Statement}$

Abstract Analysis $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$

Soundness Criteria

Concrete Evaluation $\langle \sigma, s \rangle \Downarrow \sigma'$ $\sigma \in \mathbf{State}$ $s \in \mathbf{Statement}$

Abstract Analysis $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ $\widehat{\sigma} \in \hat{\mathbf{State}}$ $\gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

Soundness Criteria

Concrete Evaluation $\langle \sigma, s \rangle \Downarrow \sigma'$ $\sigma \in \mathbf{State}$ $s \in \mathbf{Statement}$

Abstract Analysis $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ $\widehat{\sigma} \in \hat{\mathbf{State}}$ $\gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

Standard Total Correctness Soundness Criteria

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma \in \gamma(\widehat{\sigma})$ **,**
then $\langle \sigma, s \rangle \Downarrow \sigma'$ **for some** $\sigma' \in \gamma(\widehat{\sigma}')$ **.**

Soundness Criteria

Concrete Evaluation	$\langle \sigma, s \rangle \Downarrow \sigma'$	$\sigma \in \mathbf{State} \quad s \in \mathbf{Statement}$
Abstract Analysis	$\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$	$\widehat{\sigma} \in \hat{\mathbf{State}} \quad \gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma \in \gamma(\widehat{\sigma})$ **,**
then $\langle \sigma, s \rangle \Downarrow \sigma'$ **for some** $\sigma' \in \gamma(\widehat{\sigma}')$ **.**

Soundness Criteria

Concrete Evaluation $\langle \sigma, s \rangle \Downarrow \sigma'$ $\sigma \in \mathbf{State}$ $s \in \mathbf{Statement}$

Abstract Analysis $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ $\widehat{\sigma} \in \hat{\mathbf{State}}$ $\gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma' \in \gamma(\widehat{\sigma}')$**,**
then $\langle \sigma, s \rangle \Downarrow \sigma'$ **for some** $\sigma \in \gamma(\widehat{\sigma})$ **.**

Soundness Criteria

Concrete Evaluation $\langle \sigma, s \rangle \Downarrow \sigma'$ $\quad \sigma \in \mathbf{State} \quad s \in \mathbf{Statement}$

Abstract Analysis $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ $\quad \widehat{\sigma} \in \hat{\mathbf{State}} \quad \gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

Post: Goal

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma' \in \gamma(\widehat{\sigma}')$**,**
then $\langle \sigma, s \rangle \Downarrow \sigma'$ **for some** $\sigma \in \gamma(\widehat{\sigma})$ **.**

Soundness Criteria

Concrete Evaluation $\langle \sigma, s \rangle \Downarrow \sigma'$ $\sigma \in \mathbf{State}$ $s \in \mathbf{Statement}$

Abstract Analysis $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ $\widehat{\sigma} \in \hat{\mathbf{State}}$ $\gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

"Total" Witness Soundness Criteria

Post: Goal

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma' \in \gamma(\widehat{\sigma}')$**,**
then $\langle \sigma, s \rangle \Downarrow \sigma'$ **for some** $\sigma \in \gamma(\widehat{\sigma})$ **.**

Soundness Criteria

Concrete Evaluation $\langle \sigma, s \rangle \Downarrow \sigma'$ $\quad \sigma \in \mathbf{State} \quad s \in \mathbf{Statement}$

Abstract Analysis $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ $\quad \widehat{\sigma} \in \hat{\mathbf{State}} \quad \gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

"Total" Witness Soundness Criteria

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma' \in \gamma(\widehat{\sigma}')$**,**
then $\langle \sigma, s \rangle \Downarrow \sigma'$ **for some** $\sigma \in \gamma(\widehat{\sigma})$ **.**

Post: Goal

$\widehat{\sigma} = \bot$ **?**

Soundness Criteria

Concrete Evaluation $\langle \sigma, s \rangle \Downarrow \sigma'$ $\sigma \in \mathbf{State}$ $s \in \mathbf{Statement}$

Abstract Analysis $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ $\widehat{\sigma} \in \hat{\mathbf{State}}$ $\gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

"Total" Witness Soundness Criteria

Post: Goal

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma' \in \gamma(\widehat{\sigma}')$**,**
then $\langle \sigma, s \rangle \Downarrow \sigma'$ **for some** $\sigma \in \gamma(\widehat{\sigma})$ **.**

Soundness Criteria

Concrete Evaluation $\langle \sigma, s \rangle \Downarrow \sigma'$ $\sigma \in \mathbf{State}$ $s \in \mathbf{Statement}$

Abstract Analysis $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ $\widehat{\sigma} \in \hat{\mathbf{State}}$ $\gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

"Total" Witness Soundness Criteria

Post: Goal

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma' \in \gamma(\widehat{\sigma}')$**,**
then $\langle \sigma, s \rangle \Downarrow \sigma'$ **for some** $\sigma \in \gamma(\widehat{\sigma})$ **.**

Ball, Kupferman, and Yorsh (2005)

Soundness Criteria

Concrete Evaluation $\langle \sigma, s \rangle \Downarrow \sigma'$ $\sigma \in \mathbf{State}$ $s \in \mathbf{Statement}$

Abstract Analysis $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ $\widehat{\sigma} \in \hat{\mathbf{State}}$ $\gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

"Total" Witness Soundness Criteria

Post: Goal

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma' \in \gamma(\widehat{\sigma}')$**,**
then $\langle \sigma, s \rangle \Downarrow \sigma'$ **for some** $\sigma \in \gamma(\widehat{\sigma})$ **.**

Ball, Kupferman, and Yorsh (2005)

Snugglebug, Alter, DART, … are under-approximate

Soundness Criteria

Concrete Evaluation $\langle \sigma, s \rangle \Downarrow \sigma'$ $\sigma \in \mathbf{State}$ $s \in \mathbf{Statement}$

Abstract Analysis $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ $\widehat{\sigma} \in \hat{\mathbf{State}}$ $\gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma' \in \gamma(\widehat{\sigma}')$**,**
then $\langle \sigma, s \rangle \Downarrow \sigma'$ **for some** $\sigma \in \gamma(\widehat{\sigma})$ **.**

Soundness Criteria

Concrete Evaluation	$\langle \sigma, s \rangle \Downarrow \sigma'$	$\sigma \in \mathbf{State} \quad s \in \mathbf{Statement}$
Abstract Analysis	$\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$	$\widehat{\sigma} \in \hat{\mathbf{State}} \quad \gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma' \in \gamma(\widehat{\sigma}')$ **and** $\langle \sigma, s \rangle \Downarrow \sigma'$**, then** $\sigma \in \gamma(\widehat{\sigma})$**.**

Soundness Criteria

Concrete Evaluation	$\langle \sigma, s \rangle \Downarrow \sigma'$	$\sigma \in \mathbf{State} \quad s \in \mathbf{Statement}$
Abstract Analysis	$\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$	$\widehat{\sigma} \in \hat{\mathbf{State}} \quad \gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

Refutation Soundness Criteria

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma' \in \gamma(\widehat{\sigma}')$ **and** $\langle \sigma, s \rangle \Downarrow \sigma'$**, then** $\sigma \in \gamma(\widehat{\sigma})$**.**

Soundness Criteria

If a loop may "produce" a conjunct of the query, we can "assume it does" (weaken the query) only at the cost of precision.

Refutation Soundness Criteria

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma' \in \gamma(\widehat{\sigma}')$ **and** $\langle \sigma, s \rangle \Downarrow \sigma'$**, then** $\sigma \in \gamma(\widehat{\sigma})$**.**

Soundness Criteria

If a loop may "produce" a conjunct of the query, we can "assume it does" (weaken the query) only at the cost of precision.

Refutation Soundness Criteria

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma' \in \gamma(\widehat{\sigma}')$ **and** $\langle \sigma, s \rangle \Downarrow \sigma'$**, then** $\sigma \in \gamma(\widehat{\sigma})$.

Refutations: Prove alarms false with "partial" witnesses, an "easier condition" for loops

Soundness Criteria

Concrete Evaluation	$\langle \sigma, s \rangle \Downarrow \sigma'$	$\sigma \in \mathbf{State} \quad s \in \mathbf{Statement}$
Abstract Analysis	$\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$	$\widehat{\sigma} \in \hat{\mathbf{State}} \quad \gamma : \hat{\mathbf{State}} \to \wp(\mathbf{State})$

If $\vdash \{\widehat{\sigma}\}\ s\ \{\widehat{\sigma}'\}$ **such that** $\sigma' \in \gamma(\widehat{\sigma}')$ **and** $\langle \sigma, s \rangle \Downarrow \sigma'$**, then** $\sigma \in \gamma(\widehat{\sigma})$**.**

Roadmap: Precise but with scalability challenges

1

$Q_1 \vee Q_2$

x . f

Q

Alias path explosion for strong updates
(On write, case split for each possible alias in Q to maintain separation)

2

while Q_{inv} $(\ldots)$ {}

Q

Loops:
Simple loop invariant inference sufficient so far but more sophisticated techniques possible if needed

Roadmap: Precise but with scalability challenges

1

$Q_1 \vee Q_2$

x . f

Q

Alias path explosion for strong updates
(On write, case split for each possible alias in Q to maintain separation)

2

$Q_1 \vee Q_2$

if (...) {} else {}

Q

Control-flow path explosion:
Ignore for now, reasonable if number of guards relevant to Q is small (e.g., [Das et al. (2002)])

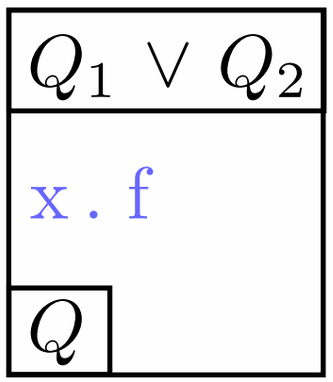
while Q_{inv} (...) {}

Q

Loops:
Simple loop invariant inference sufficient so far but more sophisticated techniques possible if needed

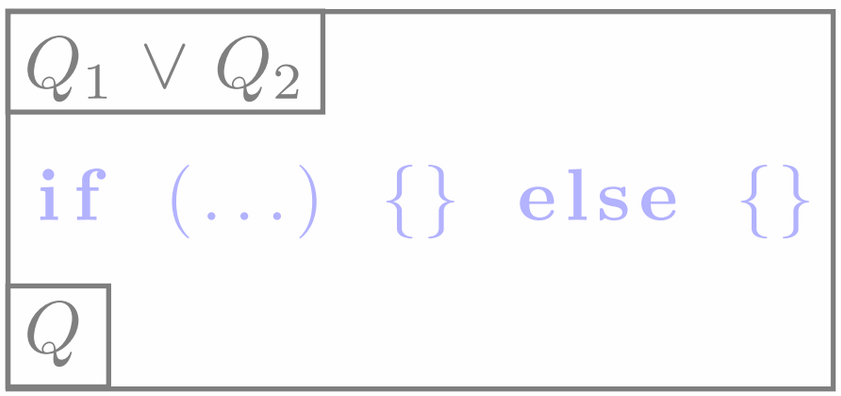
Roadmap: Precise but with scalability challenges

1

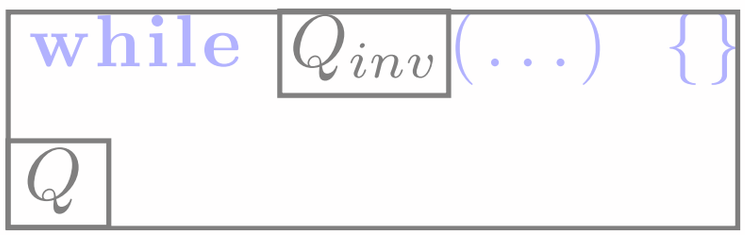


Alias path explosion for strong updates
(On write, case split for each possible alias in Q to maintain separation)

2

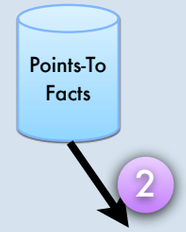


Control-flow path explosion:
Ignore for now, reasonable if number of guards relevant to Q is small (e.g., [Das et al. (2002)])

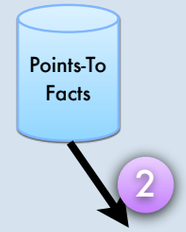


Loops:
Simple loop invariant inference sufficient so far but more sophisticated techniques possible if needed

from constraints: Reducing separation constraints with points-to facts

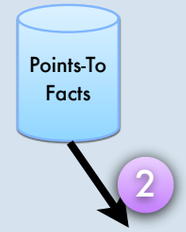


from constraints: Reducing separation constraints with points-to facts



o from { …, a_i ,… }

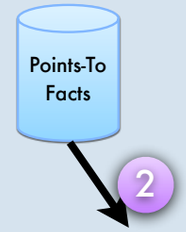
from constraints: Reducing separation constraints with points-to facts



o from { …, a_i ,… }

symbolic object instance (an address)

from constraints: Reducing separation constraints with points-to facts

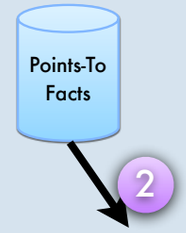


o from { …, a_i ,… }

symbolic object instance (an address)

abstract loc in points-to (set of addresses)

from constraints: Reducing separation constraints with points-to facts



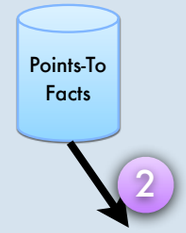
o from { …, a_i ,… }

symbolic object instance (an address)

abstract loc in points-to (set of addresses)

Refute (derive false) if:

from constraints: Reducing separation constraints with points-to facts



o from { …, a_i ,… }

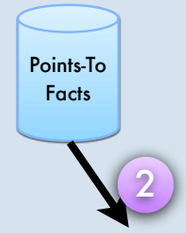
symbolic object instance (an address)

abstract loc in points-to (set of addresses)

Refute (derive false) if:

`i > j ∧ i < j`

from constraints: Reducing separation constraints with points-to facts



o from { …, a_i ,… }

symbolic object instance (an address)

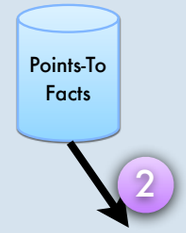
abstract loc in points-to (set of addresses)

Refute (derive false) if:

`i > j ∧ i < j`

or `o·f↦p * o·f↦q ∧ p≠q`

from constraints: Reducing separation constraints with points-to facts



o from { …, a_i ,… }

symbolic object instance (an address)

abstract loc in points-to (set of addresses)

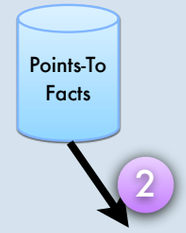
Refute (derive false) if:

`i > j ∧ i < j`

or `o·f↦p * o·f↦q ∧ p≠q`

or `o` from ∅

from constraints: Reducing separation constraints with points-to facts



o from { …, a_i ,… }

symbolic object instance (an address)

abstract loc in points-to (set of addresses)

Refute (derive false) if:

`i > j ∧ i < j`

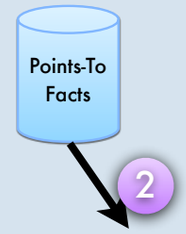
or `o·f↦p * o·f↦q ∧ p≠q`

or `o` from ∅

`x.f = p`

$\boxed{\mathtt{y \cdot f \mapsto p}}$

from constraints: Reducing separation constraints with points-to facts



o from { …, a_i ,… }

symbolic object instance (an address)

abstract loc in points-to (set of addresses)

Refute (derive false) if:

`i > j ∧ i < j`

or `o·f↦p * o·f↦q ∧ p≠q`

or `o` from ∅

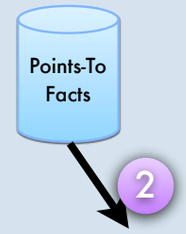
$\boxed{\mathtt{y} \text{ from } \mathrm{pt}(\mathtt{x}) \cap \mathrm{pt}(\mathtt{y}) \wedge \mathtt{x} = \mathtt{y}}$

$\vee \boxed{\mathtt{y} \cdot \mathtt{f} \mapsto \mathtt{p} \wedge \mathtt{x} \neq \mathtt{y}}$

`x.f = p`

$\boxed{\mathtt{y} \cdot \mathtt{f} \mapsto \mathtt{p}}$

from constraints: Reducing separation constraints with points-to facts



o from { …, a_i ,… }

symbolic object instance (an address)

abstract loc in points-to (set of addresses)

Refute (derive false) if:

`i > j ∧ i < j`

or `o·f↦p * o·f↦q ∧ p≠q`

or `o` from ∅

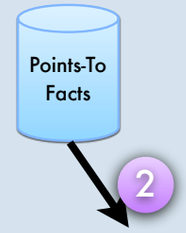
$\boxed{\mathtt{y} \text{ from } \mathrm{pt}(\mathtt{x}) \cap \mathrm{pt}(\mathtt{y}) \wedge \mathtt{x} = \mathtt{y}}$

$\vee \boxed{\mathtt{y} \cdot \mathtt{f} \mapsto \mathtt{p} \wedge \mathtt{x} \neq \mathtt{y}}$

```
x.f = p
```

$\boxed{\mathtt{y} \cdot \mathtt{f} \mapsto \mathtt{p}}$

from constraints: Reducing separation constraints with points-to facts



o from { …, a_i ,… }

symbolic object instance (an address)

abstract loc in points-to (set of addresses)

Refute (derive false) if:

`i > j ∧ i < j`

or `o·f↦p * o·f↦q ∧ p≠q`

or `o` from ∅

`y` from $\mathrm{pt}(\mathtt{x}) \cap \mathrm{pt}(\mathtt{y})$ $\wedge\ \mathtt{x} = \mathtt{y}$

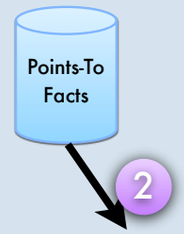
$\vee$ $\mathtt{y} \cdot \mathtt{f} \mapsto \mathtt{p} \wedge \mathtt{x} \neq \mathtt{y}$

`x.f = p`

$\mathtt{y} \cdot \mathtt{f} \mapsto \mathtt{p}$

Generalized disalias check:
$\mathrm{pt}(\mathrm{x}) \cap \mathrm{pt}(\mathrm{y}) = \varnothing$

from constraints: Reducing separation constraints with points-to facts



o from { …, a_i ,… }

symbolic object instance (an address)

abstract loc in points-to (set of addresses)

Restriction on possible abstract locations based on flow in the backwards analysis

Refute (derive false) if:

`i > j ∧ i < j`

or `o·f↦p * o·f↦q ∧ p≠q`

or `o` from ∅

`y` from $\mathrm{pt}(\mathtt{x}) \cap \mathrm{pt}(\mathtt{y}) \wedge \mathtt{x} = \mathtt{y}$

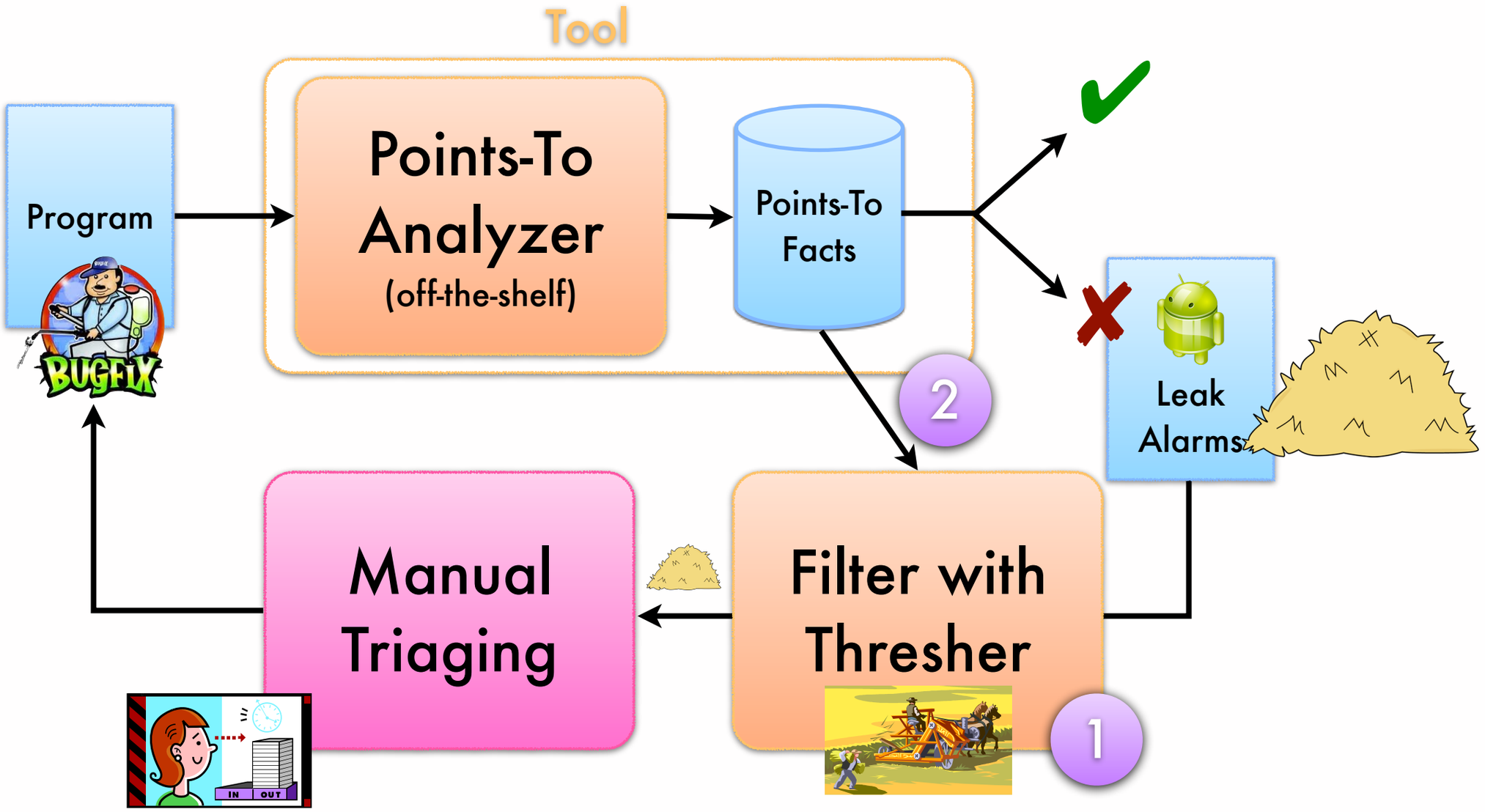
$\vee$ $\mathtt{y} \cdot \mathtt{f} \mapsto \mathtt{p} \wedge \mathtt{x} \neq \mathtt{y}$

`x.f = p`

$\mathtt{y} \cdot \mathtt{f} \mapsto \mathtt{p}$

Generalized disalias check:
$\mathrm{pt}(\mathrm{x}) \cap \mathrm{pt}(\mathrm{y}) = \varnothing$

Roadmap: Thresher filters out false alarms by refuting them one-by-one.



Idea 1: Refute points-to on-demand with second precise "filter" analysis

Idea 2: Leverage the facts from the first analysis in the filter analysis to scale

Is Thresher effective at filtering?

Thresher analyzes Java VM bytecode

7 Android app benchmarks

2,000 to 40,000 source lines of code

+ 880,000 sources lines of Android framework code

Off-the-shelf, state-of-the-art points-to analysis from WALA

Is Thresher effective at filtering?

1

Program	LOC	Points-To Alarms	Thresher Refuted	True Bugs	Thresher Time (s)	False Alarm %	Filtered %
PulsePoint	unknown	16	8	8	95	0	100
StandupTimer	2K	25	15	0	1068	100	60
DroidLife	3K	3	0	3	1	0	-
SMSPopUp	7K	5	1	4	46	0	100
aMetro	20K	54	18	36	18	0	100
K9Mail	40K	208	130	64	374	18	90
Total	**72K**	**311**	**172**	**115**	**1602**	**17**	**88**

Is Thresher effective at filtering?

1

Program	LOC	Points-To Alarms	Thresher Refuted	True Bugs	Thresher Time (s)	False Alarm %	Filtered %
PulsePoint	unknown	16	8	8	95	0	100
StandupTimer	2K	25	15	0	1068	100	60
DroidLife	3K	3	0	3	1	0	-
SMSPopUp	7K	5	1	4	46	0	100
aMetro	20K	54	18	36	18	0	100
K9Mail	40K	208	130	64	374	18	90
Total	**72K**	**311**	**172**	**115**	**1602**	**17**	**88**

staticfield-Activity pairs

Is Thresher effective at filtering?

1

Program	LOC	Points-To Alarms	Thresher Refuted
PulsePoint	unknown	16	8
StandupTimer	2K	25	15
DroidLife	3K	3	0
SMSPopUp	7K	5	1
aMetro	20K	54	18
K9Mail	40K	208	130
Total	**72K**	**311**	**172**

`staticfield-Activity` pairs

Filtered

Is Thresher effective at filtering?

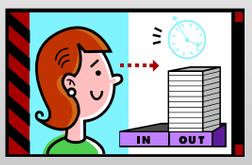
1

Program	LOC	Points-To Alarms	Thresher Refuted	True Bugs
PulsePoint	unknown	16	8	8
StandupTimer	2K	25	15	0
DroidLife	3K	3	0	3
SMSPopUp	7K	5	1	4
aMetro	20K	54	18	36
K9Mail	40K	208	130	64
Total	**72K**	**311**	**172**	**115**

`staticfield-Activity` pairs

Filtered

Manual



Is Thresher effective at filtering?

1

Program	LOC	Points-To Alarms	Thresher Refuted	True Bugs
PulsePoint	unknown	16	8	8
StandupTimer	2K	25	15	0
DroidLife	3K	3	0	3
SMSPopUp	7K	5	1	4
aMetro	20K	54	18	36
K9Mail	40K	208	130	64
Total	**72K**	**311**	**172**	**115**

Is Thresher effective at filtering?

1

Program	LOC	Points-To Alarms	Thresher Refuted	True Bugs	Thresher Time (s)
PulsePoint	unknown	16	8	8	95
StandupTimer	2K	25	15	0	1068
DroidLife	3K	3	0	3	1
SMSPopUp	7K	5	1	4	46
aMetro	20K	54	18	36	18
K9Mail	40K	208	130	64	374
Total	**72K**	**311**	**172**	**115**	**1602**

Is Thresher effective at filtering?

1

Program	LOC	Points-To Alarms	Thresher Refuted	True Bugs	Thresher Time (s)
PulsePoint	unknown	16	8	8	95
StandupTimer	2K	25	15	0	1068
DroidLife	3K	3	0	3	1
SMSPopUp	7K	5	1	4	46
aMetro	20K	54	18	36	18
K9Mail	40K	208	130	64	374
Total	**72K**	**311**	**172**	**115**	**1602**

< ~coffee to lunch break

Is Thresher effective at filtering?

1

Program	LOC	Points-To Alarms	Thresher Refuted	True Bugs	Thresher Time (s)
PulsePoint	unknown	16	8	8	95
StandupTimer	2K	25	15	0	1068
DroidLife	3K	3	0	3	1
SMSPopUp	7K	5	1	4	46
aMetro	20K	54	18	36	18
K9Mail	40K	208	130	64	374
Total	**72K**	**311**	**172**	**115**	**1602**

Is Thresher effective at filtering?

1

Program	LOC	Points-To Alarms	Thresher Refuted	True Bugs	Thresher Time (s)	False Alarm %
PulsePoint	unknown	16	8	8	95	0
StandupTimer	2K	25	15	0	1068	100
DroidLife	3K	3	0	3	1	0
SMSPopUp	7K	5	1	4	46	0
aMetro	20K	54	18	36	18	0
K9Mail	40K	208	130	64	374	18
Total	**72K**	**311**	**172**	**115**	**1602**	**17**

% after filtering

Is Thresher effective at filtering?

1

Program	LOC	Points-To Alarms	Thresher Refuted	True Bugs	Thresher Time (s)	False Alarm %	Filtered %
PulsePoint	unknown	16	8	8	95	0	100
StandupTimer	2K	25	15	0	1068	100	60
DroidLife	3K	3	0	3	1	0	-
SMSPopUp	7K	5	1	4	46	0	100
aMetro	20K	54	18	36	18	0	100
K9Mail	40K	208	130	64	374	18	90
Total	**72K**	**311**	**172**	**115**	**1602**	**17**	**88**

% after filtering

Is Thresher effective at filtering?

1

Program	LOC	Points-To Alarms	Thresher Refuted	True Bugs	Thresher Time (s)	False Alarm %	Filtered %
PulsePoint	unknown	16	8	8	95	0	100
StandupTimer	2K	25	15	0	1068	100	60
DroidLife	3K	3	0	3	1	0	-
SMSPopUp	7K	5	1	4	46	0	100
aMetro	20K	54	18	36	18	0	100
K9Mail	40K	208	130	64	374	18	90
Total	**72K**	**311**	**172**	**115**	**1602**	**17**	**88**

Is Thresher effective at filtering?

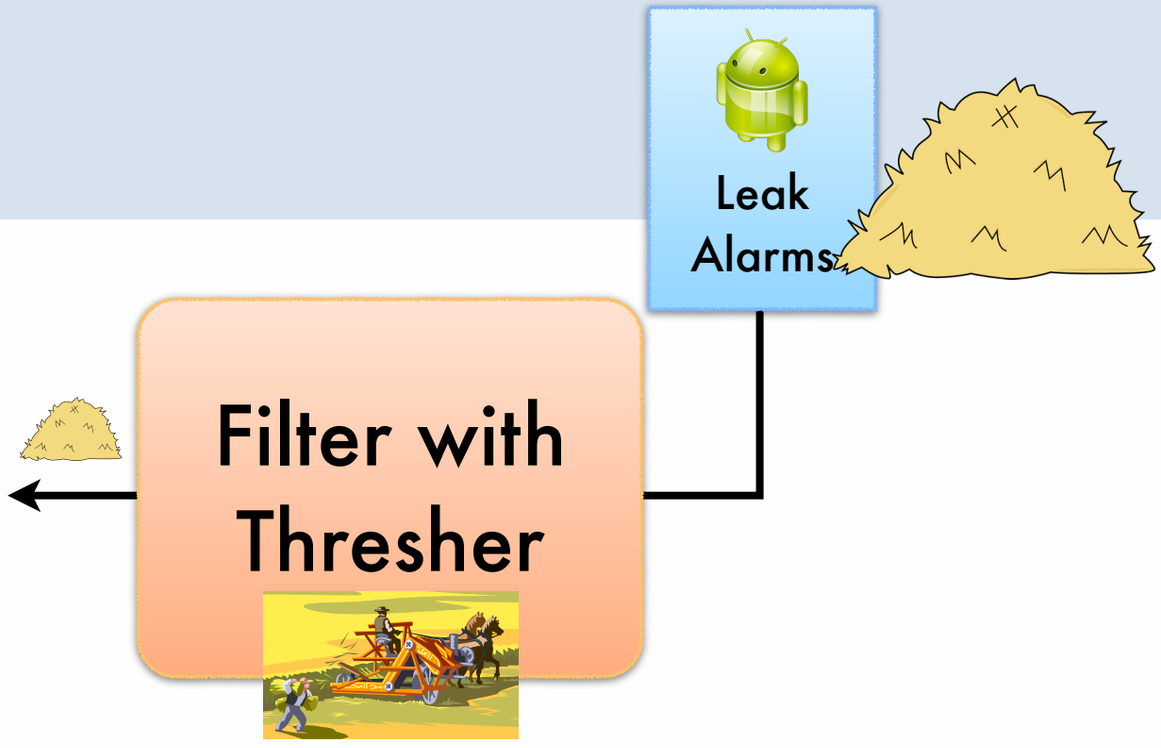
1

Program	LOC	Points-To Alarms	Thresher Refuted	True Bugs	Thresher Time (s)	False Alarm %	Filtered %
PulsePoint	unknown	16	8	8	95	0	100
StandupTimer	2K	25	15	0	1068	100	60
DroidLife	3K	3	0	3	1	0	-
SMSPopUp	7K	5	1	4	46	0	100
aMetro	20K	54	18	36	18	0	100
K9Mail	40K	208	130	64	374	18	90
Total	**72K**	**311**	**172**	**115**	**1602**	**17**	**88**

False alarms down to 17% from 63% (points-to analysis only)

Thresher filters 88% of false alarms from points-to analysis

Some Highlights



Thresher: Precise Refutations for Heap Reachability

Assist in triage of queries about heap relations

- Assume alarms false, prove them so (refute) automatically with a "partial" witness search
- Reduced separation constraints with points-to facts
- Filters out ~90% of false alarms to expose true bugs
- Application: Find memory leaks and eliminate crashes in Android

Roadmap: Precise but with scalability challenges

1

$Q_1 \vee Q_2$

x . f

Q

Alias path explosion for strong updates
(On write, case split for each possible alias in Q to maintain separation)

2

$Q_1 \vee Q_2$

if (...) {} else {}

Q

Control-flow path explosion:
Ignore for now, reasonable if number of guards relevant to Q is small (e.g., [Das et al. (2002)])

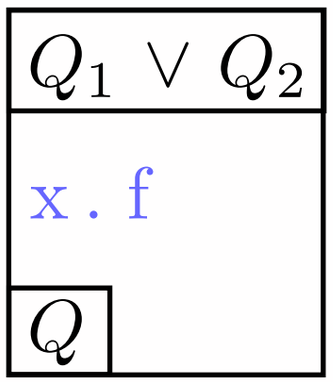
while Q_{inv} (...) {}

Q

Loops:
Simple loop invariant inference sufficient so far but more sophisticated techniques possible if needed

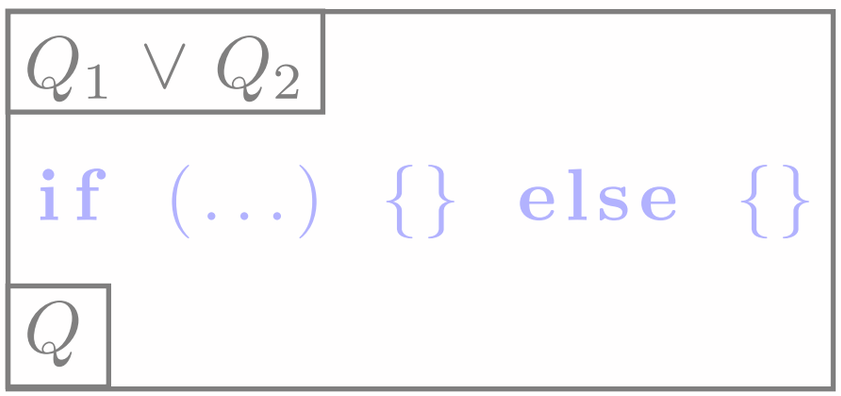
Roadmap: Precise but with scalability challenges

1

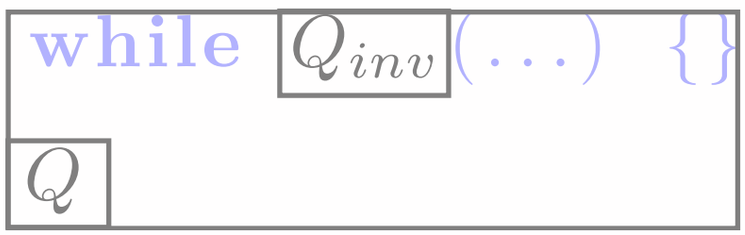


Alias path explosion for strong updates
(On write, case split for each possible alias in Q to maintain separation)

2



Control-flow path explosion:
Ignore for now, reasonable if number of guards relevant to Q is small (e.g., [Das et al. (2002)])



Loops:
Simple loop invariant inference sufficient so far but more sophisticated techniques possible if needed

Roadmap: Precise but with scalability challenges

1

$Q_1 \vee Q_2$

x . f

Q

Alias path explosion for strong updates
(On write, case split for each possible alias in Q to maintain separation)

2

$Q_1 \vee Q_2$

if (...) {} else

Q

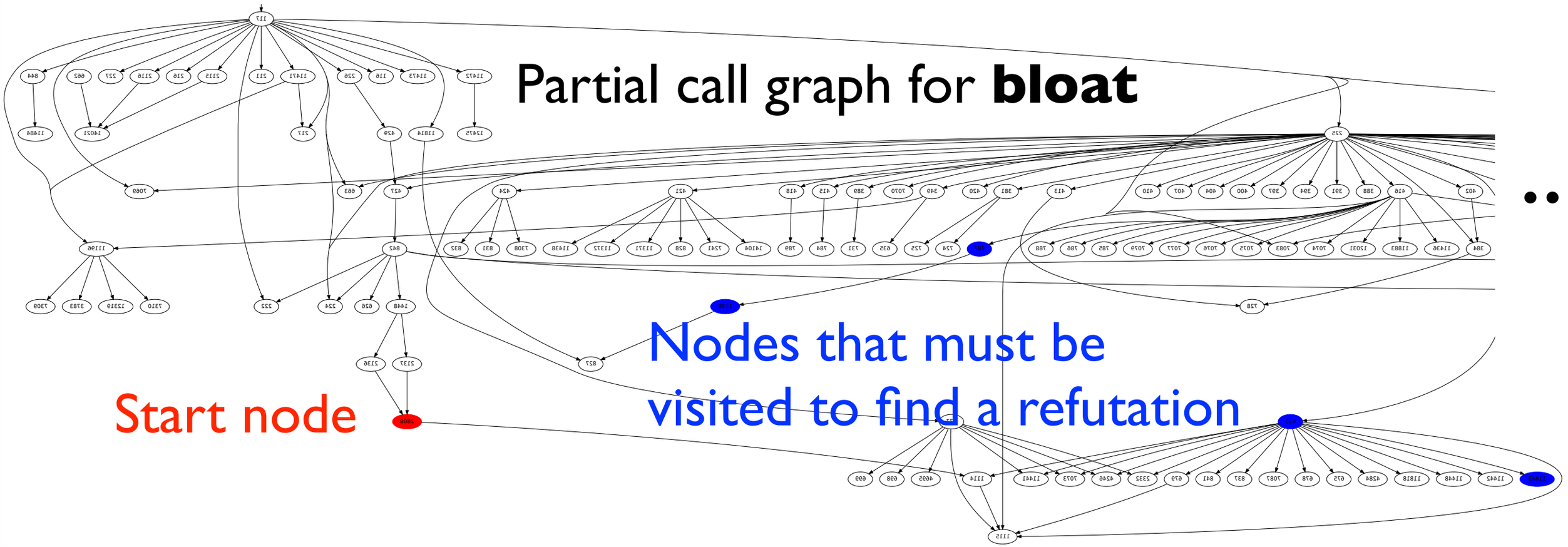
Control-flow path explosion:
Ignore for now, reasonable if number of guards relevant to Q is small (e.g., [Das et al. (2002)])

while Q_{inv} (...) {}

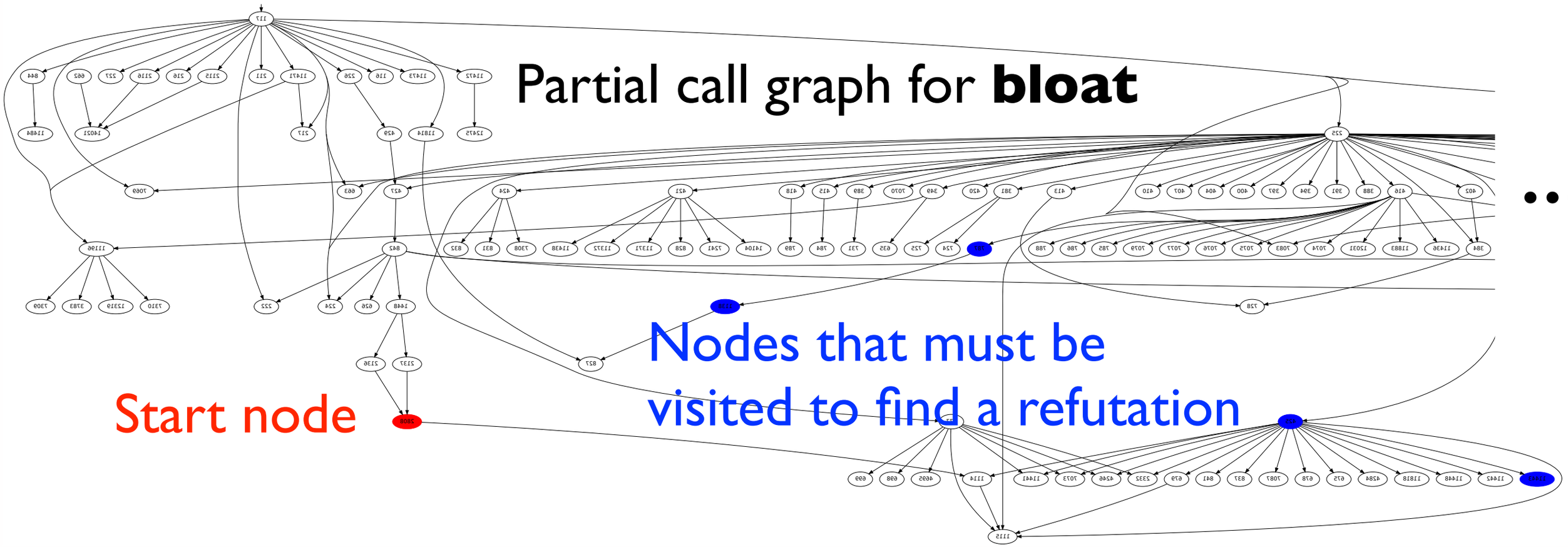
Q

Loops:
Simple loop invariant inference sufficient so far but more sophisticated techniques possible if needed

Teaser: Most transitions are irrelevant

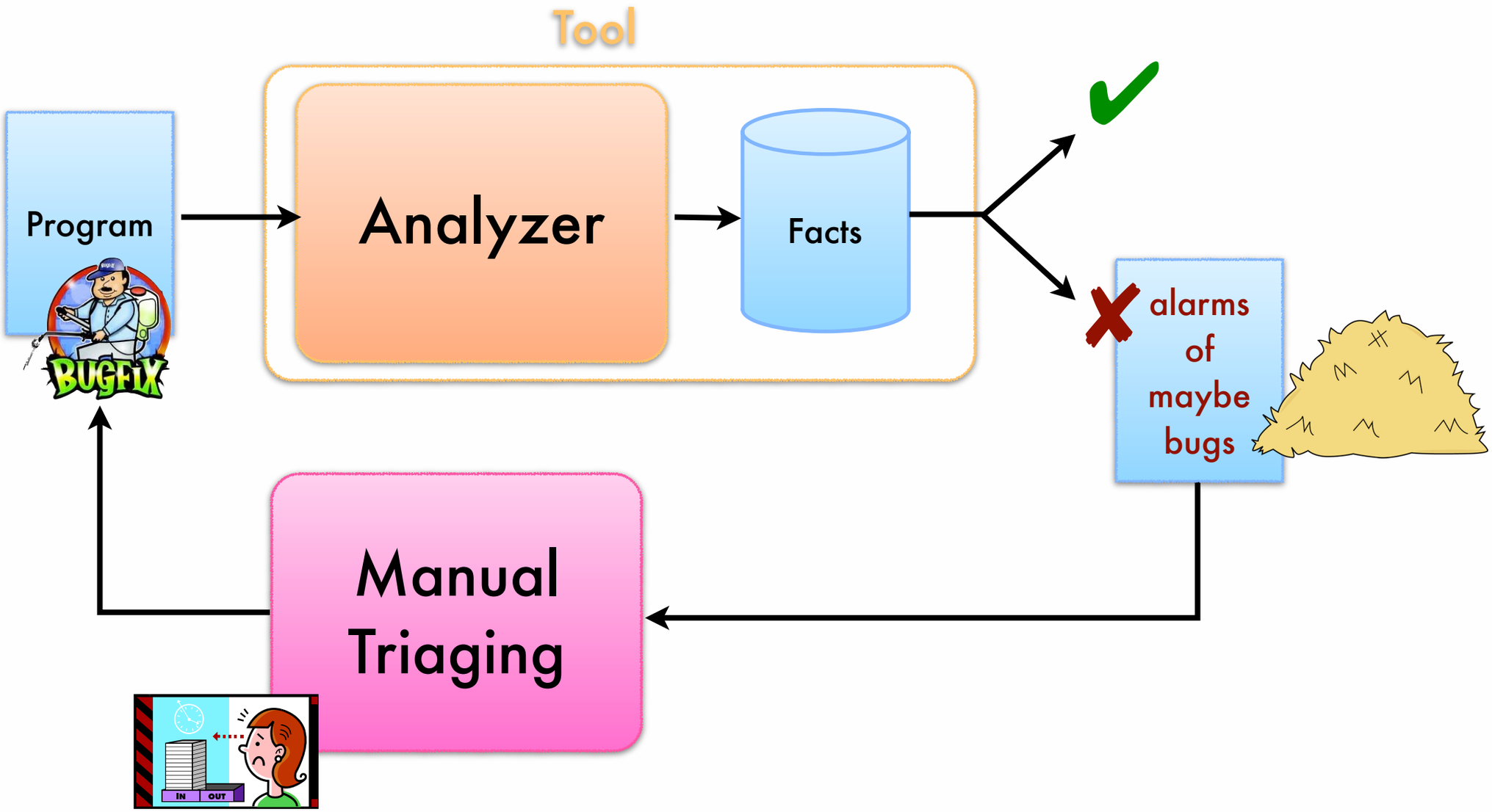


Teaser: Most transitions are irrelevant

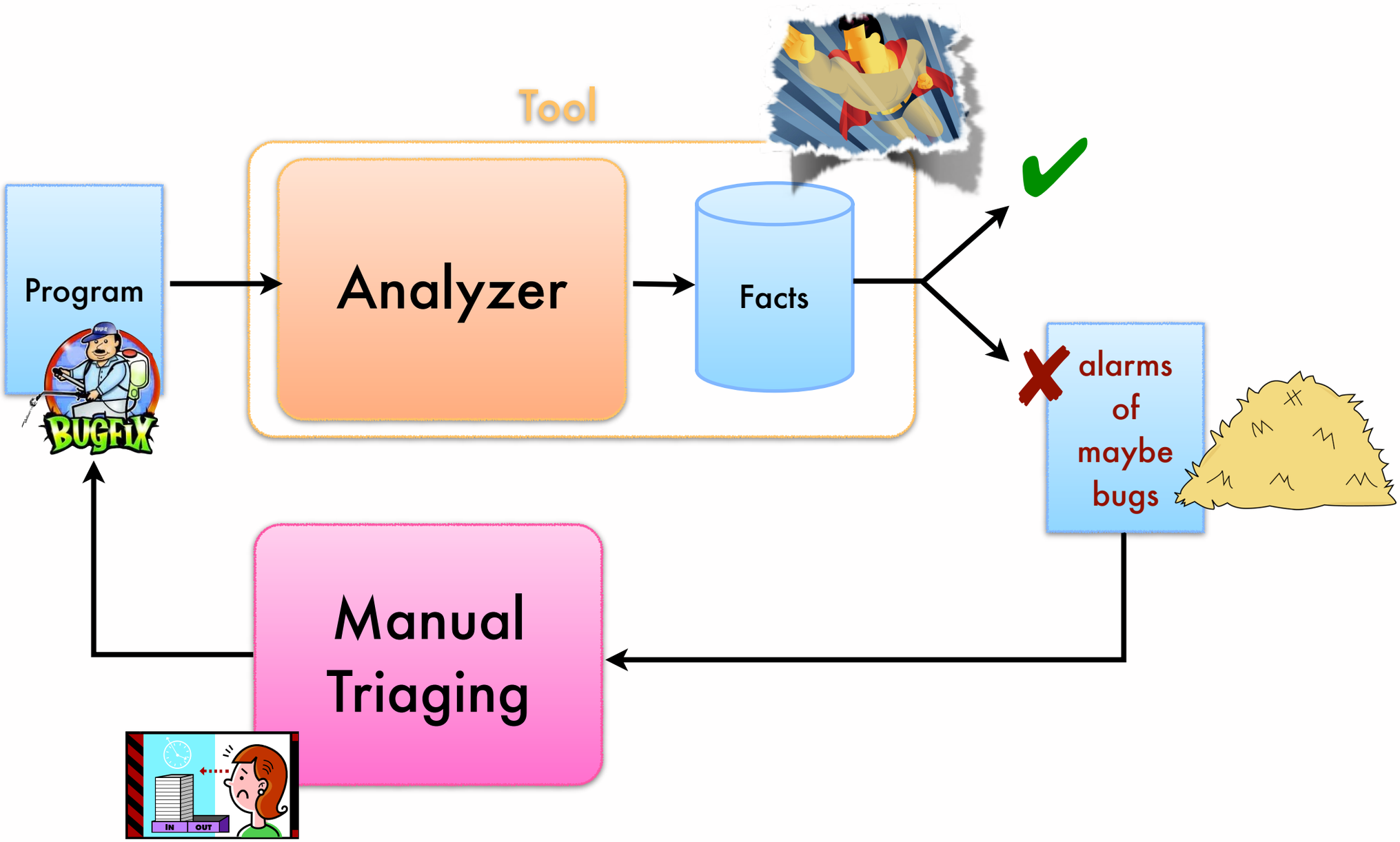


How do we soundly "jump" to the transitions that are **relevant** for deriving the **refutation**?

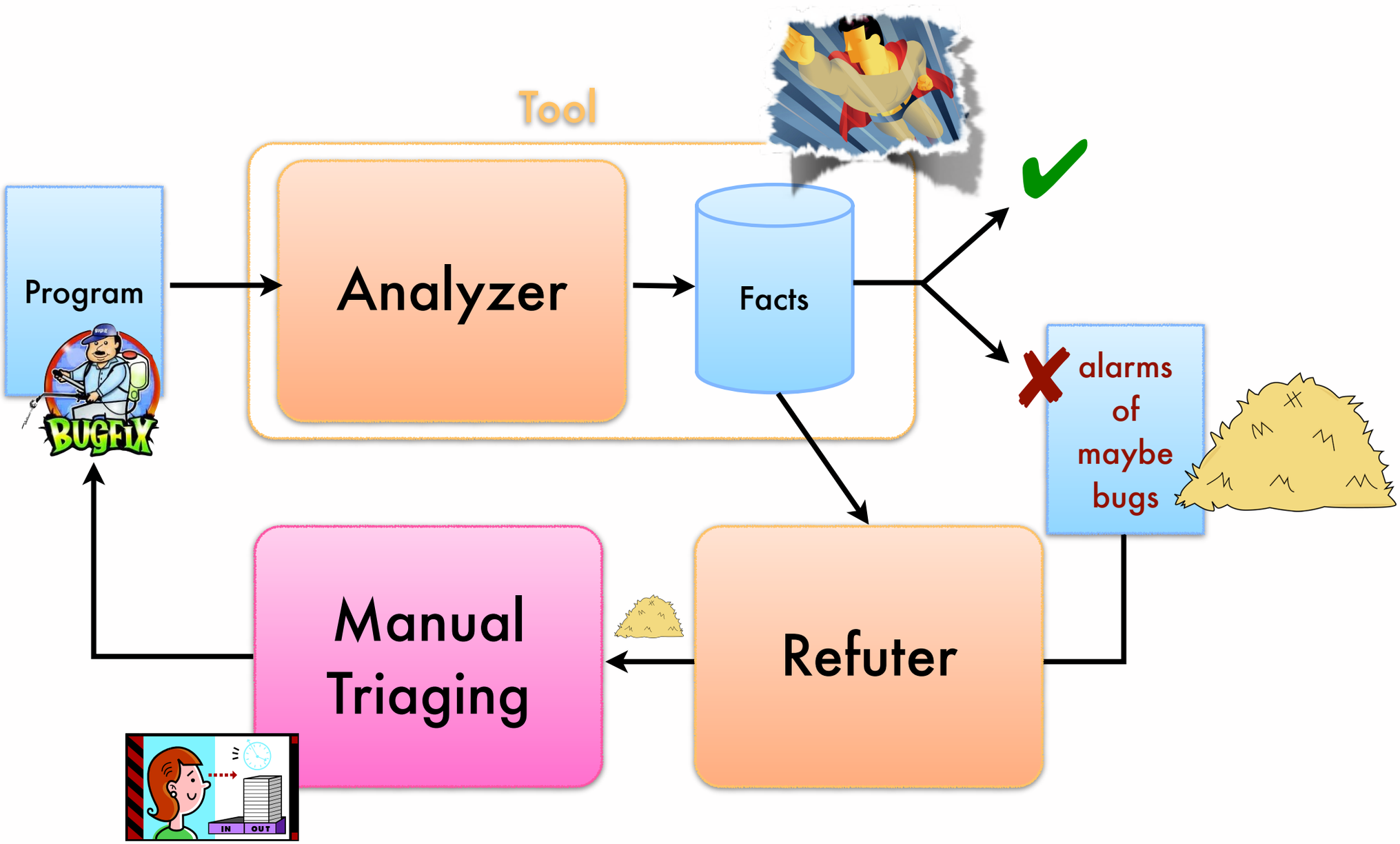
Final Commentary: Design and apply analyses to the whole bug mitigation process!



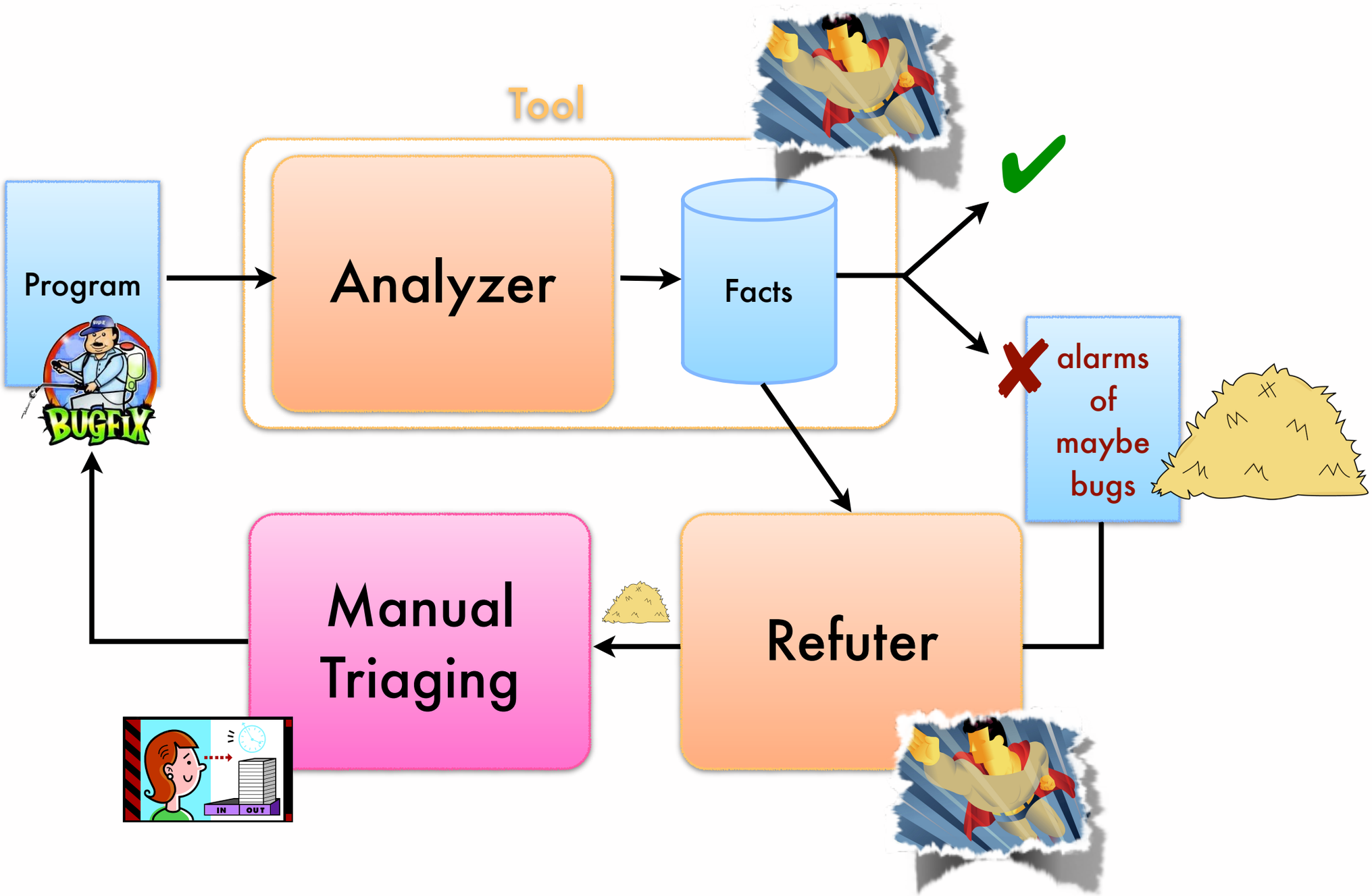
Final Commentary: Design and apply analyses to the whole bug mitigation process!



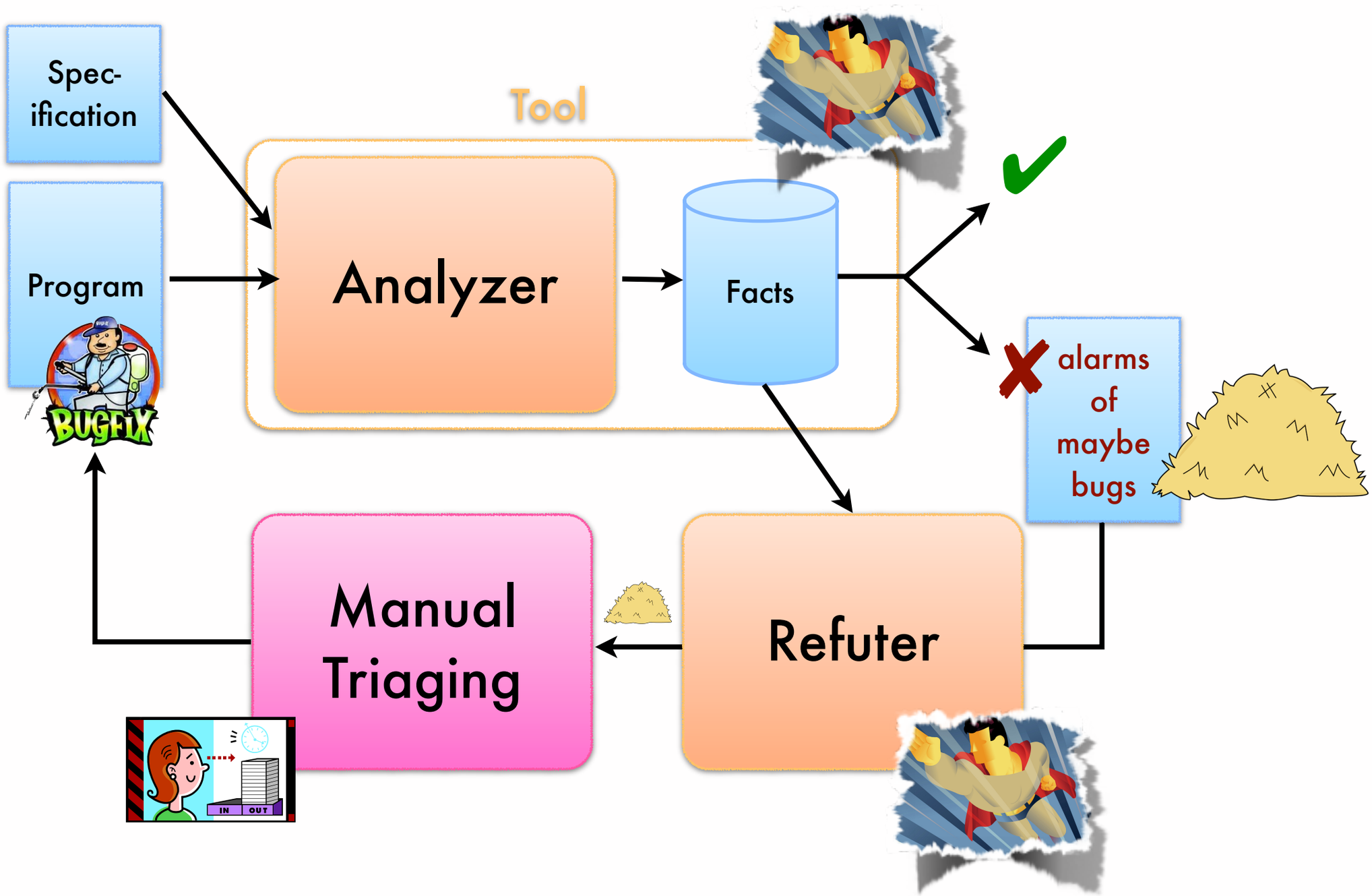
Final Commentary: Design and apply analyses to the whole bug mitigation process!



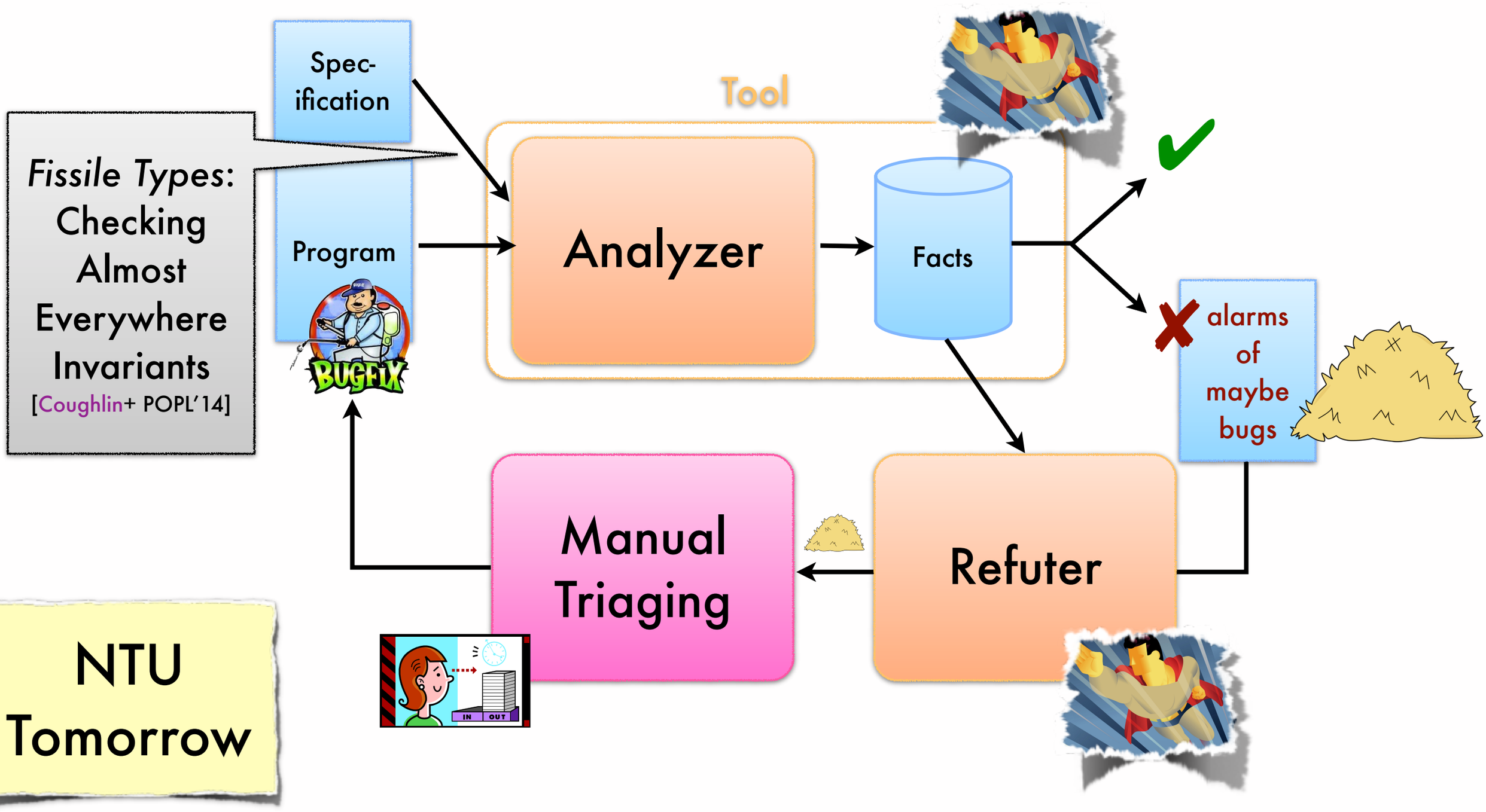
Final Commentary: Design and apply analyses to the whole bug mitigation process!



Final Commentary: Design and apply analyses to the whole bug mitigation process!



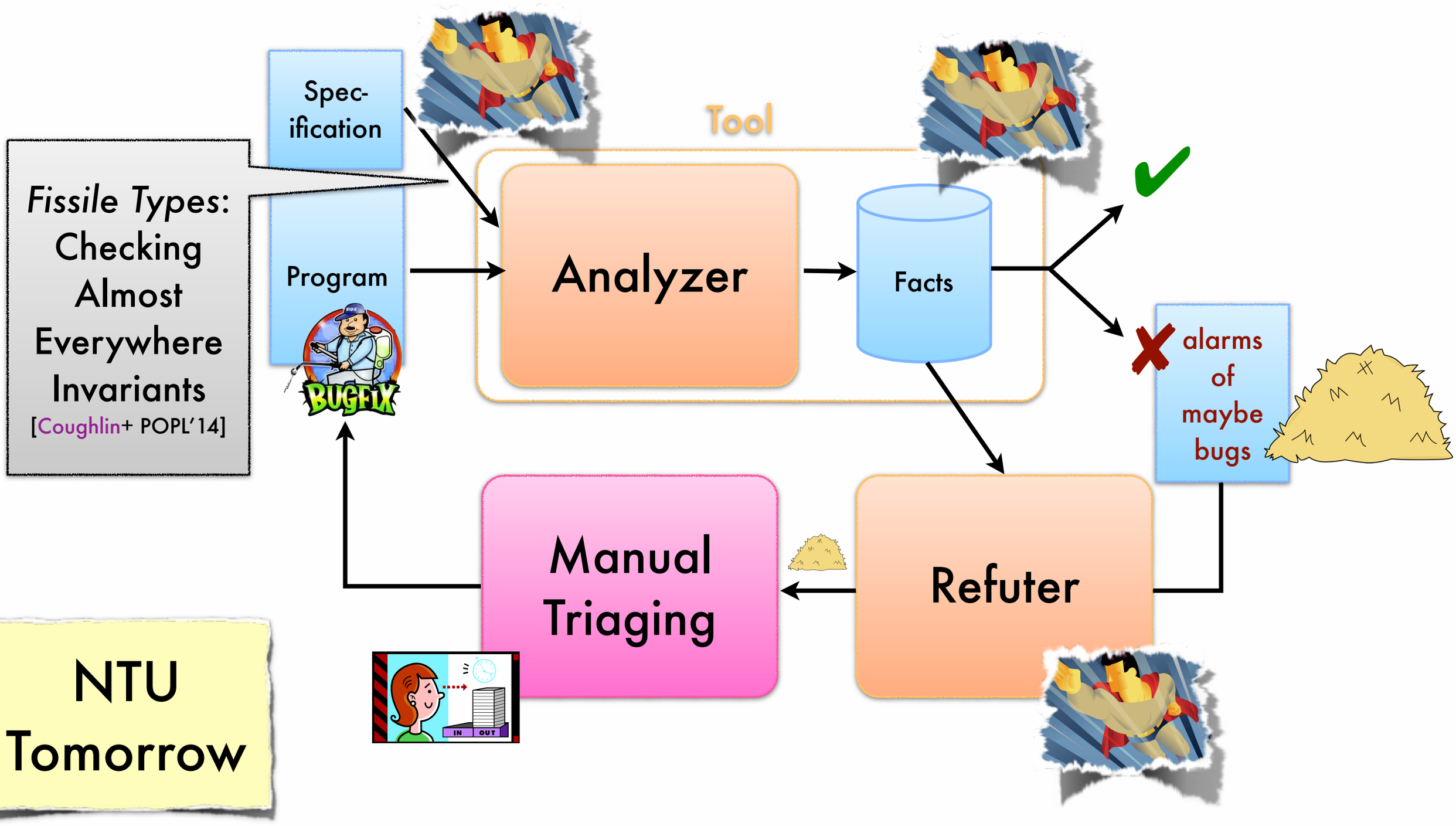
Final Commentary: Design and apply analyses to the whole bug mitigation process!



Fissile Types: Checking Almost Everywhere Invariants
[Coughlin+ POPL'14]

NTU Tomorrow

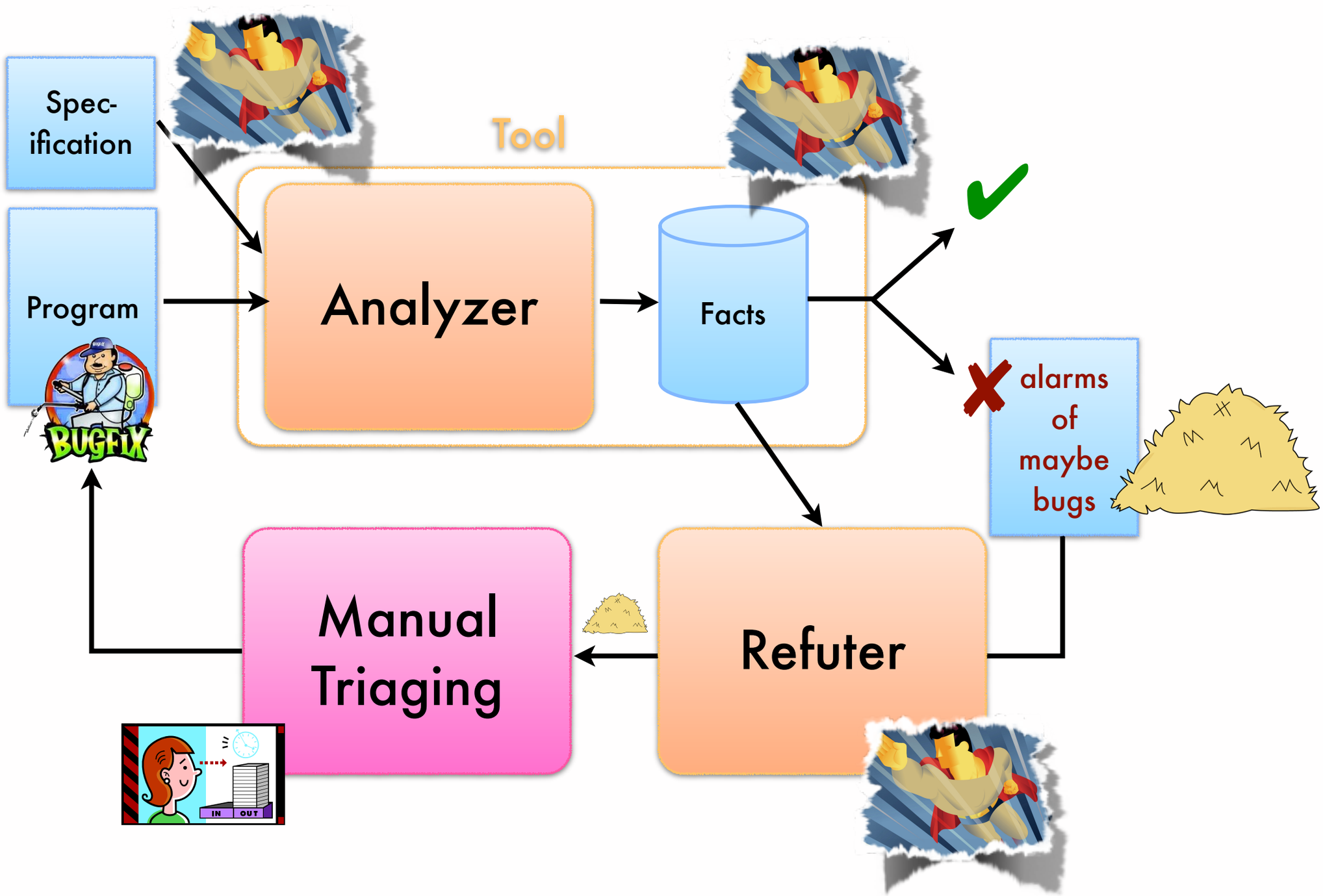
Final Commentary: Design and apply analyses to the whole bug mitigation process!



Fissile Types: Checking Almost Everywhere Invariants
[Coughlin+ POPL'14]

NTU Tomorrow

Final Commentary: Design and apply analyses to the whole bug mitigation process!



Thank You!

PLV

www.cs.colorado.edu/~bec

pl.cs.colorado.edu

Thank You!

PLV

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


Cerny


Chang


Sankaranaryananan


Somenzi

Android
OS

... in the process of finding leaks in apps

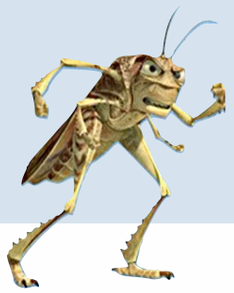
Find Android's HashMap bug ...

```
class HashMap {
  static Object[] EMPTY = new Object[2]; ...
  HashMap() { this.tbl = EMPTY; capacity initially empty }

  void put(Object key, Object val) {
    if (need capacity) {
      this.tbl = new Object[more capacity];
      copy from old table
    }
    this.tbl[bucket using hash of key] = val;
  }

  HashMap(Map m) {
    if (m.size() < 1) { this.tbl = EMPTY; }
    else { this.tbl = new Object[at least m.size()]; }
    copy from m
  }
}
```

Find Android's HashMap bug ...



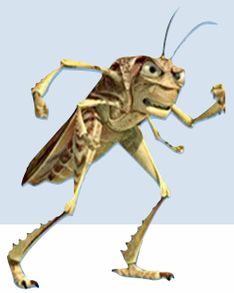
Null object pattern: Should never be written to

```
class HashM
  static Object[] EMPTY = new Object[2]; ...
  HashMap() { this.tbl = EMPTY; capacity initially empty }

  void put(Object key, Object val) {
    if (need capacity) {
      this.tbl = new Object[more capacity];
      copy from old table
    }
    this.tbl[bucket using hash of key] = val;
  }

  HashMap(Map m) {
    if (m.size() < 1) { this.tbl = EMPTY; }
    else { this.tbl = new Object[at least m.size()]; }
    copy from m
  }
}
```

Find Android's HashMap bug ...



Null object pattern: Should never be written to

```
class HashM
  static Object[] EMPTY = new Object[2]; ...
  HashMap() { this.tbl = EMPTY; capacity initially empty }

  void put(Object key, Object val) {
    if (need capacity) {
      this.tbl = new Object[more capacity];
      copy from old table
    }
    this.tbl[bucket using hash of key] = val;
  }

  HashMap(Map m) {
    if (m.size() < 1) { this.tbl = EMPTY; }
    else { this.tbl = new Object[at least m.size()]; }
    copy from m
  }
}
```

allocate new backing array on first write

Find Android's HashMap bug ...

Null object pattern: Should never be written to

```
class HashM
  static Object[] EMPTY = new Object[2]; ...
  HashMap() { this.tbl = EMPTY; capacity initially empty }

  void put(Object key, Object val) {
    if (need capacity) {
      this.tbl = new Object[more capacity];
      copy from old table
    }
    this.tbl[bucket using hash of key] = val;
  }

  HashMap(Map m) {
    if (m.size() < 1) { this.tbl = EMPTY; }
    else { this.tbl = new Object[at least m.size()]; }
    copy from m
  }
}
```

allocate new backing array on first write

Find Android's HashMap bug ...

Null object pattern: Should never be written to

```
class HashM
  static Object[] EMPTY = new Object[2]; ...
  HashMap() { this.tbl = EMPTY; capacity initially empty }

  void put(Object key, Object val) {
    if (need capacity) {
      this.tbl = new Object[more capacity];
      copy from old table
    }
    this.tbl[bucket using hash of key] = val;
  }

  HashMap(Map m) {
    if (m.size() < 1) { this.tbl = EMPTY; }
    else { this.tbl = new Object[at least m.size()]; }
    copy from m
  }
}
```

allocate new backing array on first write

An "evil" implementation of the `Map` interface can corrupt `EMPTY`. Then, all `HashMap`s created in the future will be corrupted.

Find Android's HashMap bug ...

Null object pattern: Should never be written to

```
class HashM
  static Object[] EMPTY = new Object[2]; ...
  HashMap() { this.tbl = EMPTY; capacity initially empty }

  void put(Object key, Object val) {
    if (need capacity) {
      this.tbl = new Object[more capacity];
      copy from old table
    }
    this.tbl[bucket using hash of key] = val;
  }

  HashMap(Map m) {
    if (m.size() < 1) { this.tbl = EMPTY; }
    else { this.tbl = new Object[at least m.size()]; }
    copy from m
  }
}
```

allocate new backing array on first write

return 0

return "evil" content

An "evil" implementation of the `Map` interface can corrupt `EMPTY`. Then, all `HashMap`s created in the future will be corrupted.

Find Android's HashMap bug ...

Null object pattern: Should never be written to

```
class HashM
  static Object[] EMPTY = new Object[2]; ...
  HashMap() { this.tbl = EMPTY; capacity initially empty }

  void put(Object key, Object val) {
    if (need capacity) {
      this.tbl = new Object[more capacity];
      copy from old table
    }
    this.
  }
```

allocate new backing array on first write

We reported this, Google fixed it

https://android-review.googlesource.com/#/c/52183/

```
  HashMap(Map m) {
    if (m.size() < 1) { this.tbl = EMPTY; }
    else { this.tbl = new Object[at least m.size()]; }
    copy from m
  }
}
```

return 0

return "evil" content

An "evil" implementation of the `Map` interface can corrupt `EMPTY`. Then, all `HashMaps` created in the future will be corrupted.