Designing code analyses
for Large Software Systems (DECA)

Eric Bodden

Fraunhofer SIT

TECHNISCHE UNIVERSITÄT DARMSTADT

SECURE SOFTWARE ENGINEERING GROUP

EC SPRIDE

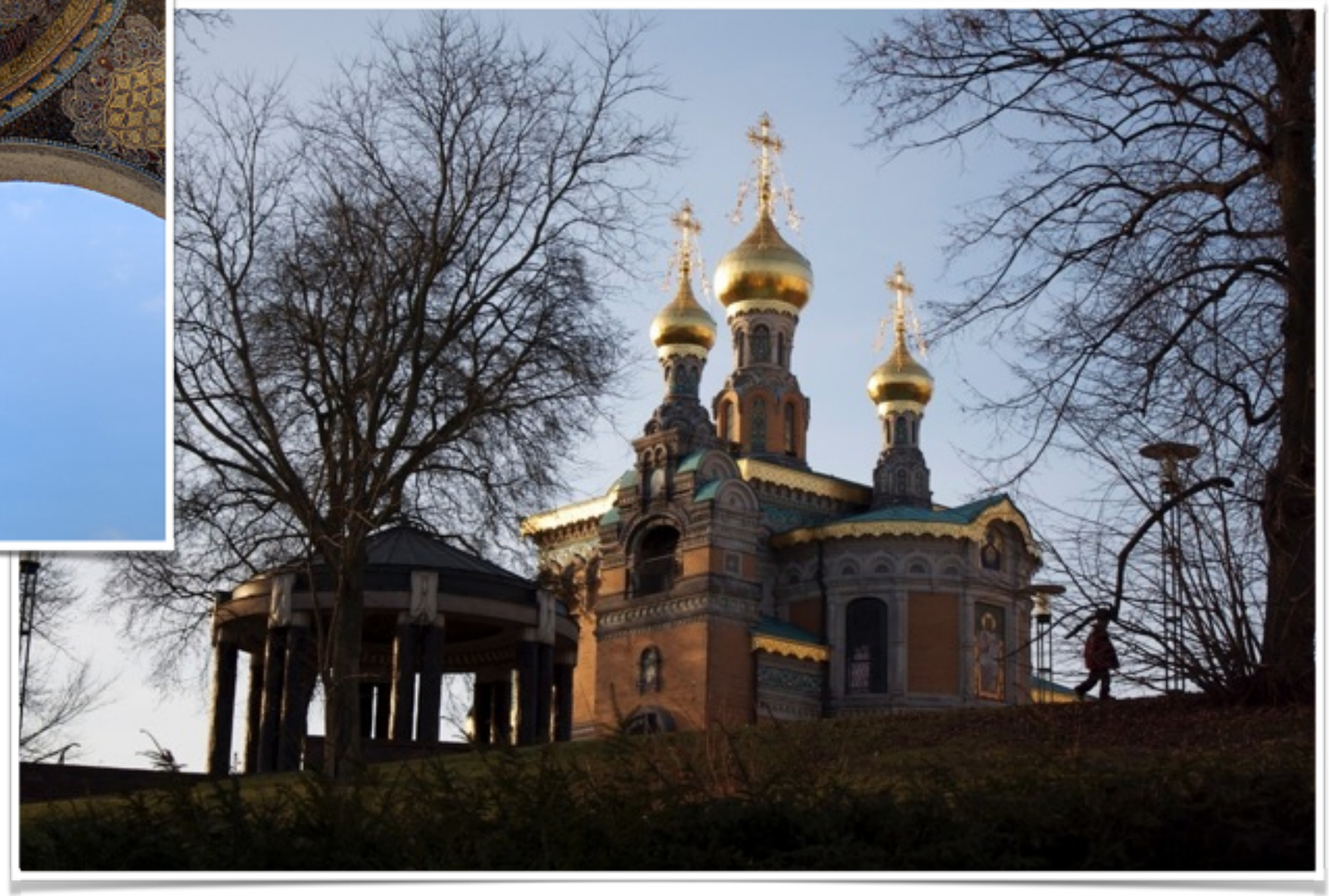
http://sseblog.ec-spride.de/deca/

Studied in Aachen

ERASMUS in Canterbury, UK

Ph.D. in Montréal, Québec

In Darmstadt since 2009



Since 2009: CASED

Since 2011: EC SPRIDE

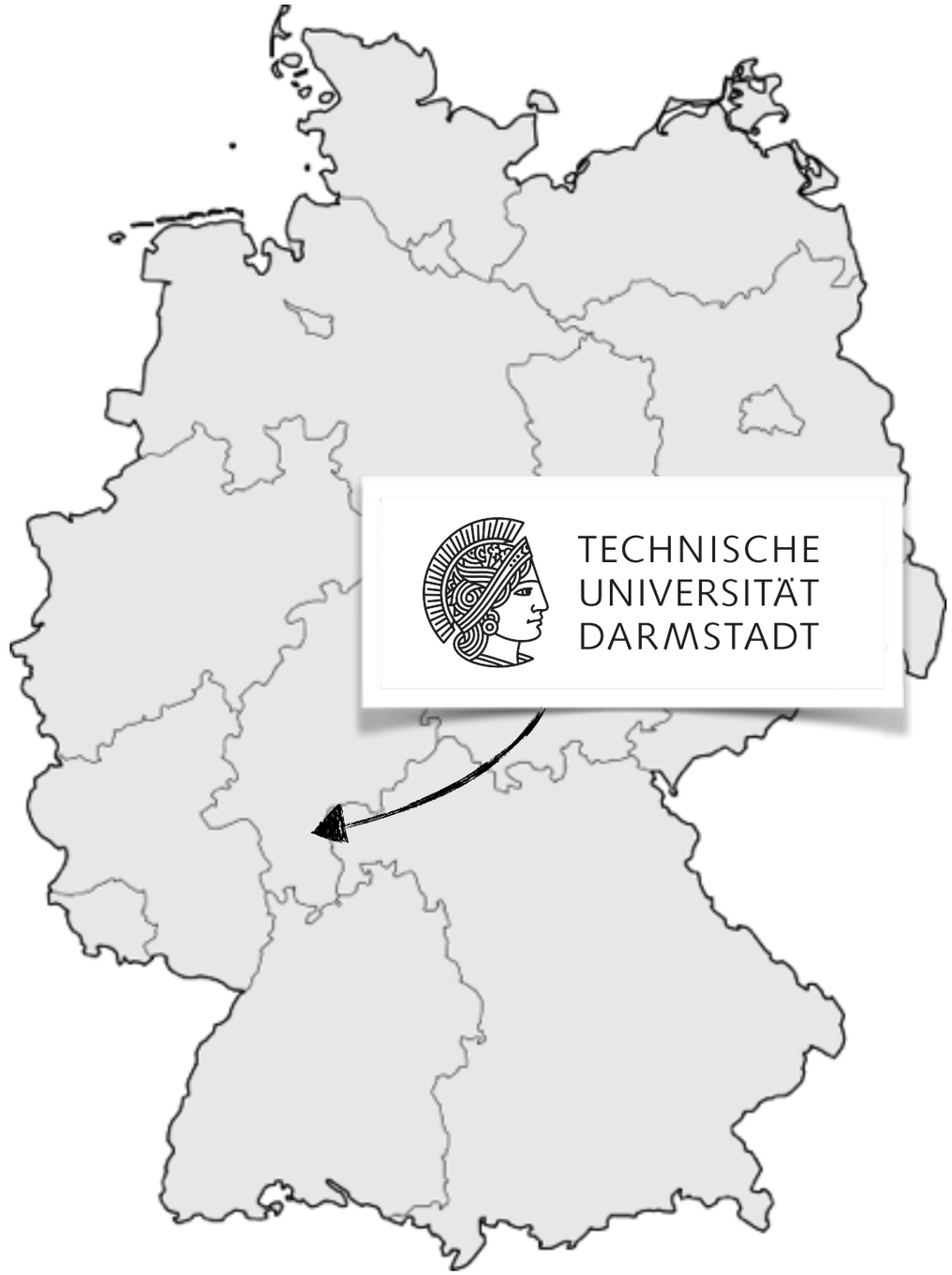
GEFÖRDERT VOM

Bundesministerium
für Bildung
und Forschung

Since 2009: CASED

Since 2011: EC SPRIDE

Since 2012: Emmy Noether-Programm, Deutsche Forschungsgemeinschaft DFG

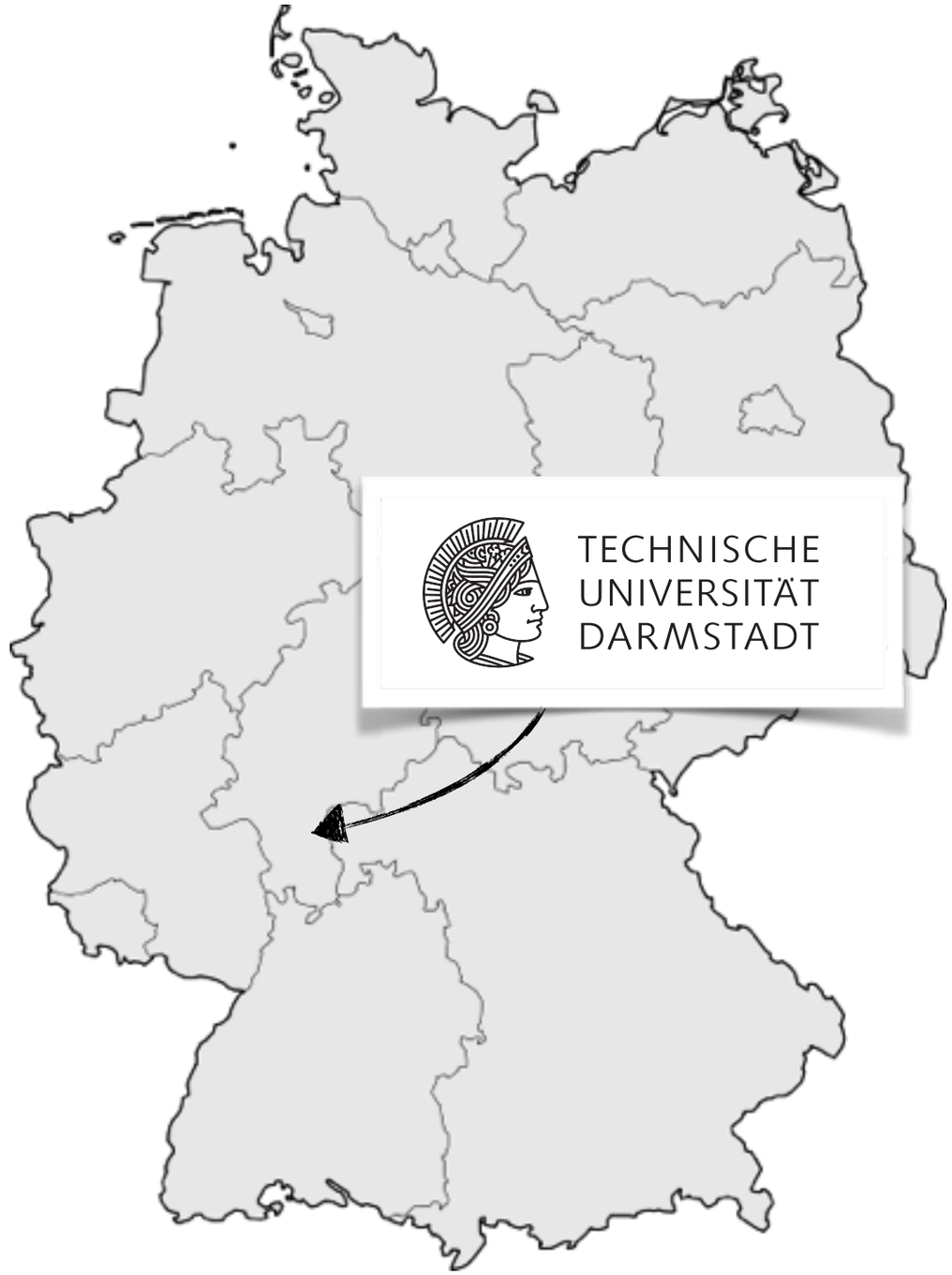


Since 2009: CASED

Since 2011: EC SPRIDE

Since 2012: Emmy Noether-Programm, Deutsche Forschungsgemeinschaft DFG

Since 2013: Fraunhofer SIT



Since 2009: CASED

Since 2011: EC SPRIDE

Since 2012: Emmy Noether-Programm, Deutsche Forschungsgemeinschaft DFG

Since 2013: Fraunhofer SIT

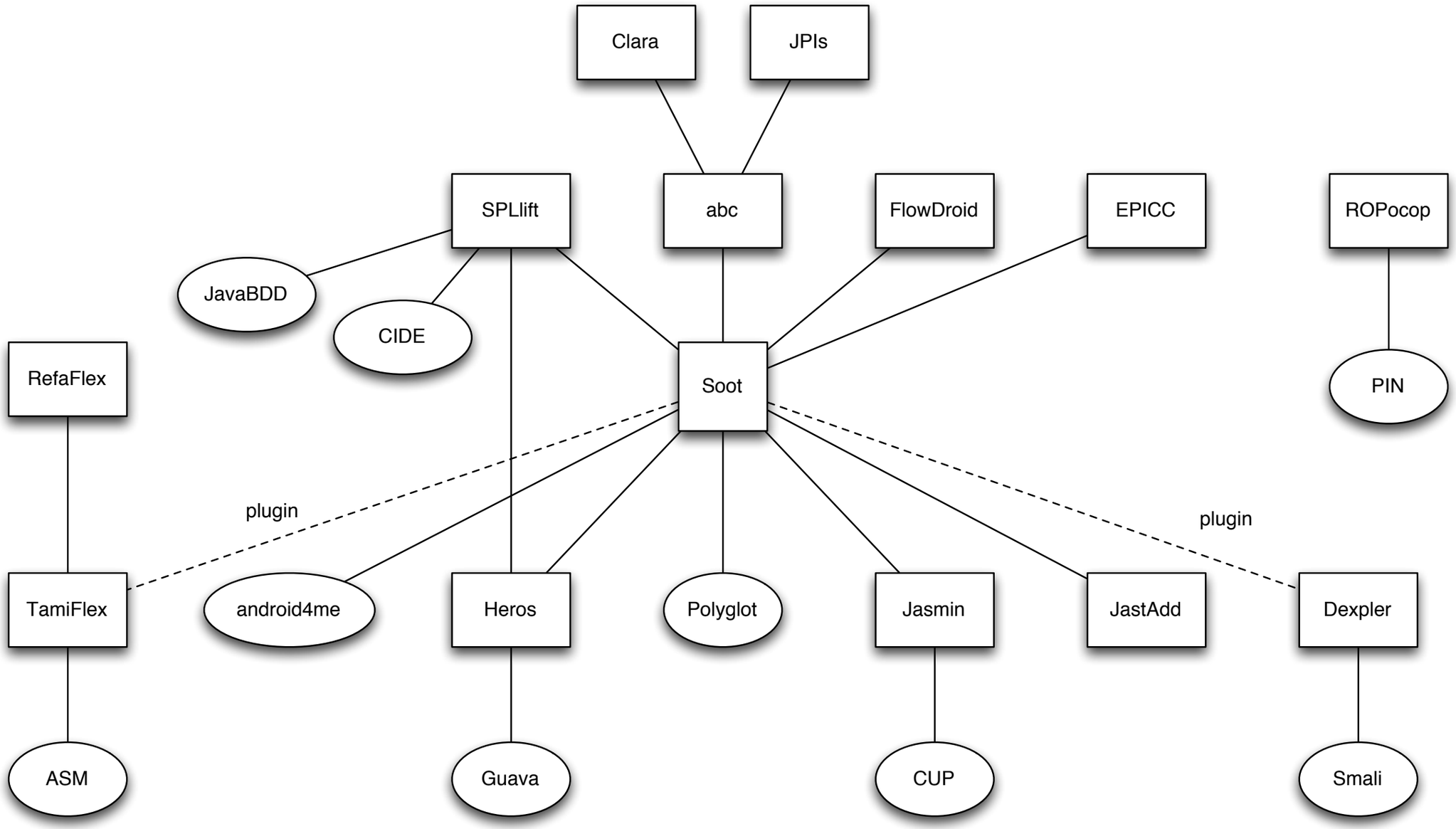
Two institutions, one group

TECHNISCHE UNIVERSITÄT DARMSTADT

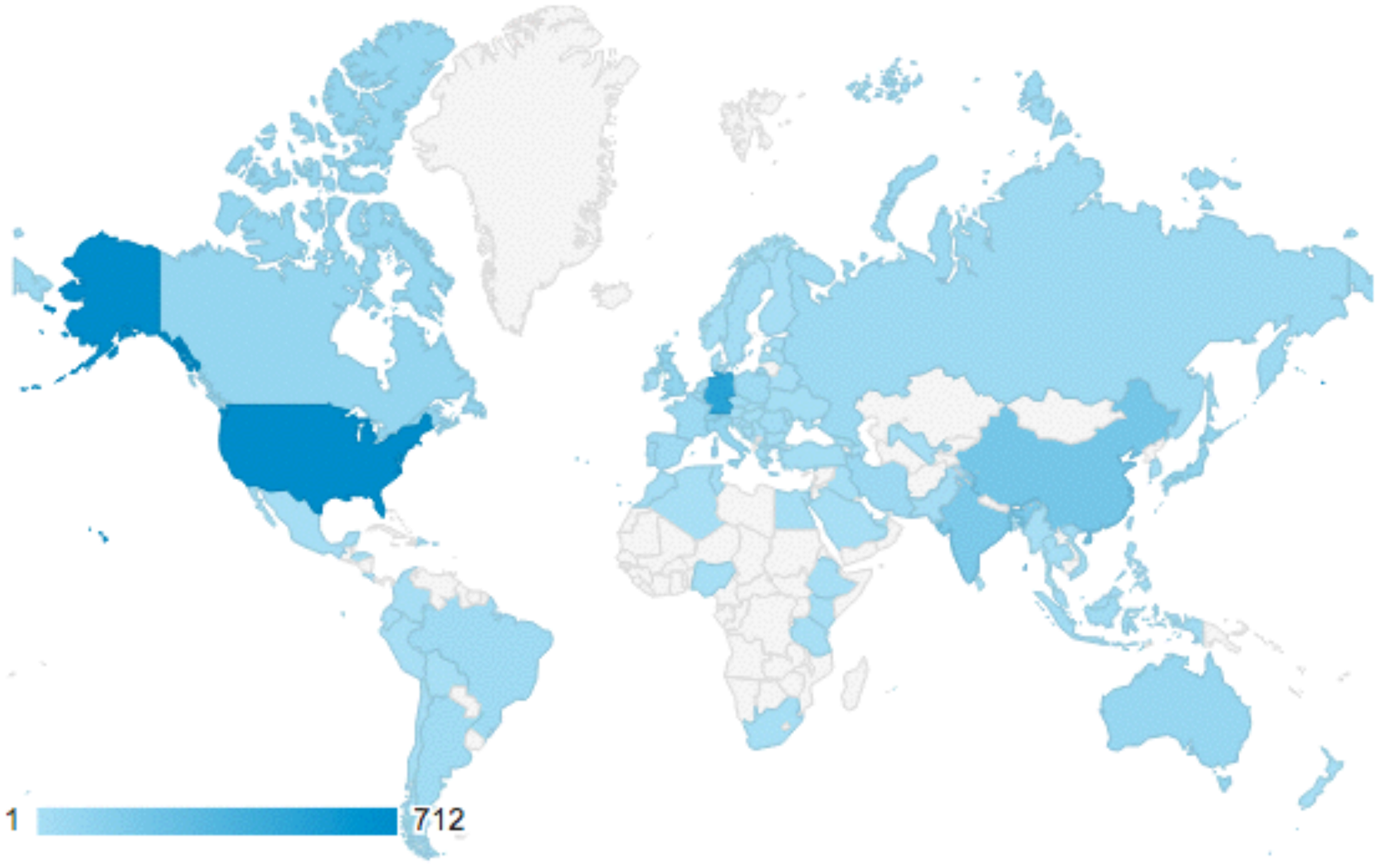
SECURE SOFTWARE ENGINEERING GROUP

Fraunhofer SIT

Develop popular tools



Develop popular tools



Current industrial partners

Where to find us

Fraunhofer SIT

5th floor

Why should I care about code analyses?

Recent Java security issues

Oracle patches Java 7 vulnerability

Breaking its quarterly update schedule, Oracle has released a new Java runtime that addres recent security flaws.

News - Aug 30, 2012, 2:50 PM | By Topher Kessler

source: cnet.com

Recent Java security issues

Oracle patches Java 7 vulnerability

Breaking its quarterly update schedule, Oracle has released a new Java runtime that addres recent security flaws.

News - Aug 30, 2012, 2:50 PM | By Topher Kessler

New vulnerabilities found in latest Java update

Following its latest updates, more vulnerabilities have been uncovered in Oracle's Java 7 runtime.

News - Aug 31, 2012, 3:50 PM | By Topher Kessler

source: cnet.com

Recent Java security issues

Oracle patches Java 7 vulnerability

Breaking its quarterly update schedule, Oracle has released a new Java runtime that addres recent security flaws.

News - Aug 30, 2012, 2:50 PM | By Topher Kessler

New vulnerabilities found in latest Java update

Following its latest updates, more vulnerabilities have been uncovered in Oracle's Java 7 runtime.

News - Aug 31, 2012, 3:50 PM | By Topher Kessler

Java flaw draws Web attacks, reports say

A vulnerability in the Java software has the potential to affect a wide swath of computer users, and researchers warn that it's already being exploited "in the wild."

News - Jan 10, 2013, 1:22 PM | By Jonathan Skillings

source: cnet.com

Recent Java security issues

Oracle patches Java 7 vulnerability

Breaking its quarterly update schedule, Oracle has released a new Java runtime that addres recent security flaws.

News - Aug 30, 2012, 2:50 PM | By Topher Kessler

New vulnerabilities found in latest Java update

Following its latest updates, more vulnerabilities have been uncovered in Oracle's Java 7 runtime.

News - Aug 31, 2012, 3:50 PM | By Topher Kessler

Java flaw draws Web attacks, reports say

A vulnerability in the Java software has the potential to affect a wide swath of computer users, and researchers warn that it's already being exploited "in the wild."

News - Jan 10, 2013, 1:22 PM | By Jonathan Skillings

Oracle releases software update to fix Java vulnerability

Emergency software update repairs vulnerability that could allow remote attackers to execute arbitrary code.

News - Jan 13, 2013, 3:43 PM | By Steven Musil

source: cnet.com

Recent Java security issues

Oracle patches Java 7 vulnerability

Breaking its quarterly update schedule, Oracle has released a new Java runtime that addres recent security flaws.

News - Aug 30, 2012, 2:50 PM | By Topher Kessler

New vulnerabilities found in latest Java update

Following its latest updates, more vulnerabilities have been uncovered in Oracle's Java 7 runtime.

News - Aug 31, 2012, 3:50 PM | By Topher Kessler

Java flaw draws Web attacks, reports say

A vulnerability in the Java software has the potential to affect a wide swath of computer users, and researchers warn that it's already being exploited "in the wild."

News - Jan 10, 2013, 1:22 PM | By Jonathan Skillings

Oracle releases software update to fix Java vulnerability

Emergency software update repairs vulnerability that could allow remote attackers to execute arbitrary code.

News - Jan 13, 2013, 3:43 PM | By Steven Musil

Homeland Security still advises disabling Java, even after update

DHS says an unpatched vulnerability may still put Web browsers using the plugin at risk of remote attack.

News - Jan 14, 2013, 4:37 PM | By Steven Musil

source: cnet.com

Recent Java security issues

Oracle patches Java 7 vulnerability

Breaking its quarterly update schedule, Oracle has released a new Java runtime that addres recent security flaws.

News - Aug 30, 2012, 2:50 PM | By Topher Kessler

New vulnerabilities found in latest Java update

Following its latest updates, more vulnerabilities have been uncovered in Oracle's Java 7 runtime.

News - Aug 31, 2012, 3:50 PM | By Topher Kessler

Java flaw draws Web attacks, reports say

A vulnerability in the Java software has the potential to affect a wide swath of computer users, and researchers warn that it's already being exploited "in the wild."

News - Jan 10, 2013, 1:22 PM | By Jonathan Skillings

Oracle releases software update to fix Java vulnerability

Emergency software update repairs vulnerability that could allow remote attackers to execute arbitrary code.

News - Jan 13, 2013, 3:43 PM | By Steven Musil

Homeland Security still advises disabling Java, even after update

DHS says an unpatched vulnerability may still put Web browsers using the plugin at risk of remote attack.

News - Jan 14, 2013, 4:37 PM | By Steven Musil

Oracle pushes out new Java update to patch security holes

Released Friday, the latest critical patch update contains fixes for 50 different security flaws.

News - Feb 04, 2013, 5:44 AM | By Lance Whitney

source: cnet.com

Recent Java security issues

Oracle patches Java 7 vulnerability

Breaking its quarterly update schedule, Oracle has released a new Java runtime that addres recent security flaws.

News - Aug 30, 2012, 2:50 PM | By Topher Kessler

New vulnerabilities found in latest Java update

Following its latest updates, more vulnerabilities have been uncovered in Oracle's Java 7 runtime.

News - Aug 31, 2012, 3:50 PM | By Topher Kessler

Java flaw draws Web attacks, reports say

A vulnerability in the Java software has the potential to affect a wide swath of computer users, and researchers warn that it's already being exploited "in the wild."

News - Jan 10, 2013, 1:22 PM | By Jonathan Skillings

Oracle releases software update to fix Java vulnerability

Emergency software update repairs vulnerability that could allow remote attackers to execute arbitrary code.

News - Jan 13, 2013, 3:43 PM | By Steven Musil

Homeland Security still advises disabling Java, even after update

DHS says an unpatched vulnerability may still put Web browsers using the plugin at risk of remote attack.

News - Jan 14, 2013, 4:37 PM | By Steven Musil

Oracle pushes out new Java update to patch security holes

Released Friday, the latest critical patch update contains fixes for 50 different security flaws.

News - Feb 04, 2013, 5:44 AM | By Lance Whitney

Oracle issues emergency Java update to patch vulnerabilities

After hackers attack a new flaw in Java, "Oracle decided to release a fix for this vulnerability and another closely related bug as soon as possible."

News - Mar 04, 2013, 7:22 PM | By Dara Kerr

New Java flaw could hit 1 billion users

A new Java vulnerability has surfaced that apparently affects all Java runtimes and therefore puts close to a billion users at risk.

News - Sep 26, 2012, 8:56 AM | By Topher Kessler

source: cnet.com

… problems can be found through code analysis


the
PONIE
AWARDS

goto fail:

```
static OSStatus
SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
                                 uint8_t *signature, UInt16 signatureLen)
{
    OSStatus        err;
    ...

    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;

    // code ommitted for brevity...

    err = sslRawVerify(ctx,
                       ctx->peerPubKey,
                       dataToSign,              /* plaintext */
                       dataToSignLen,           /* plaintext length */
                       signature,
                       signatureLen);
    if(err) {
        sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                    "returned %d\n", (int)err);
        goto fail;
    }

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;

}
```

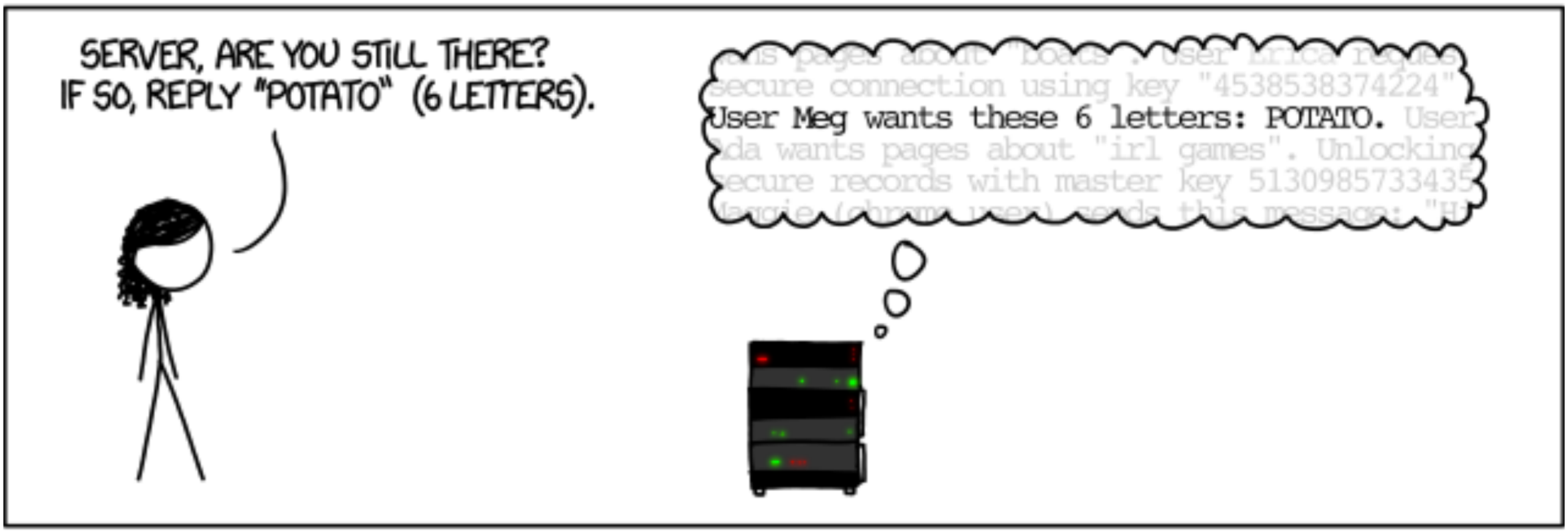
Oops...

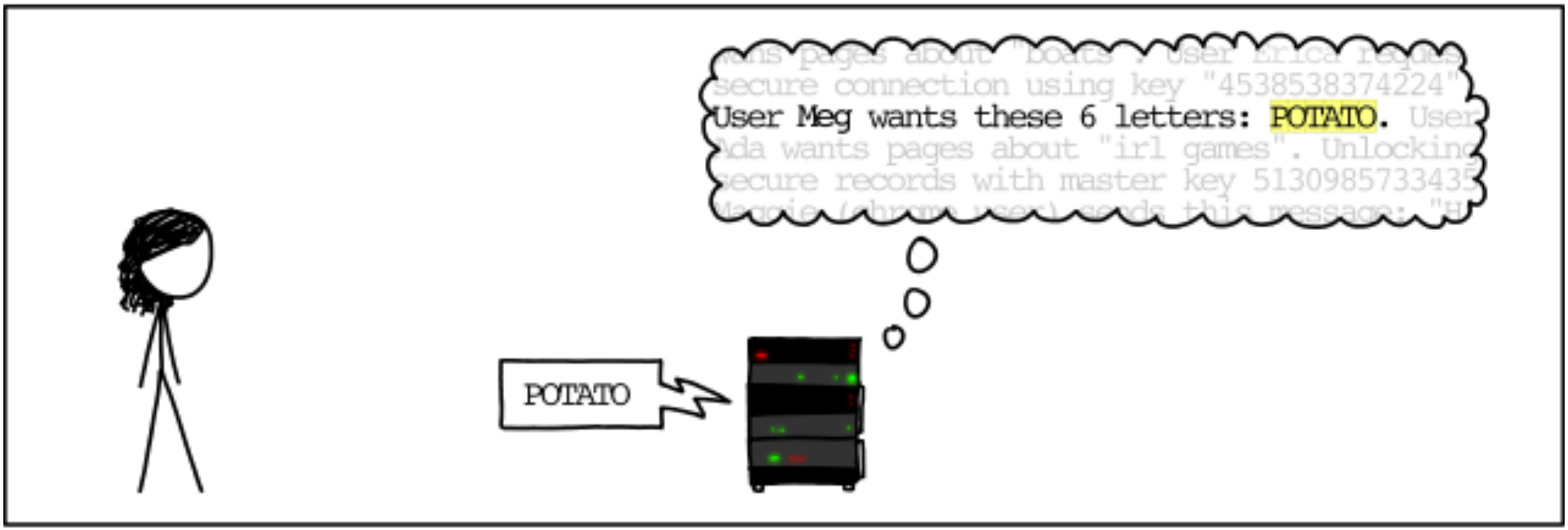
Never gets called
(but needed to be)...

Despite the name, always
returns "it's OK!!!"

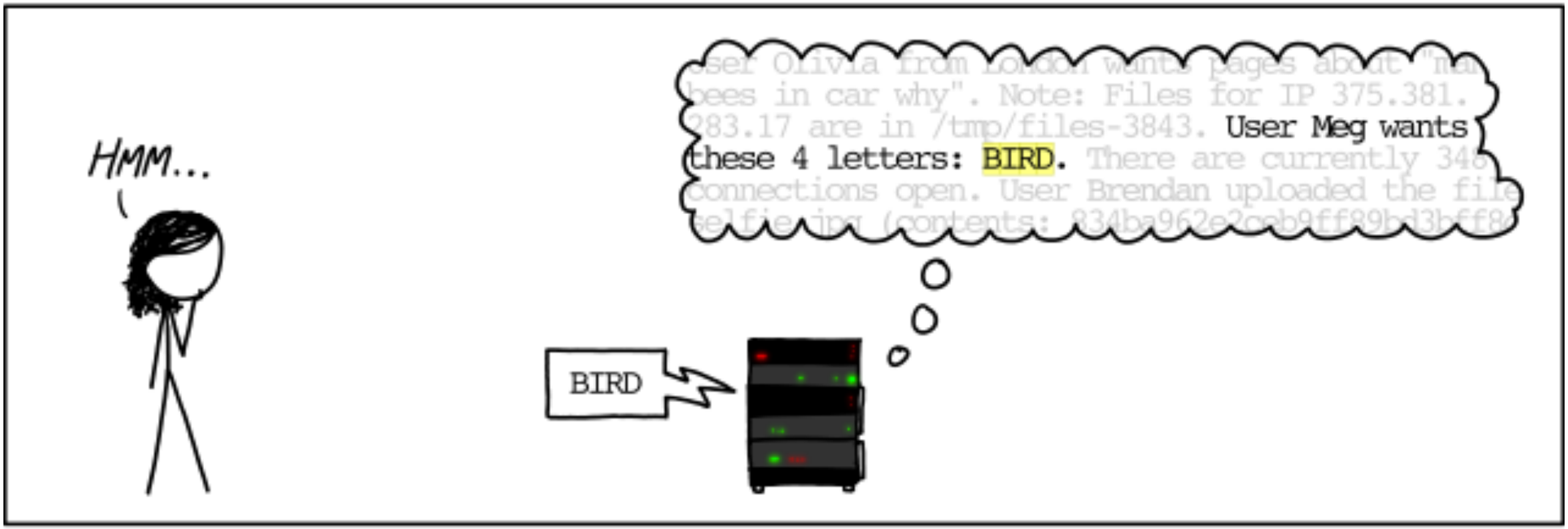
… could have been *easily* found through code analysis

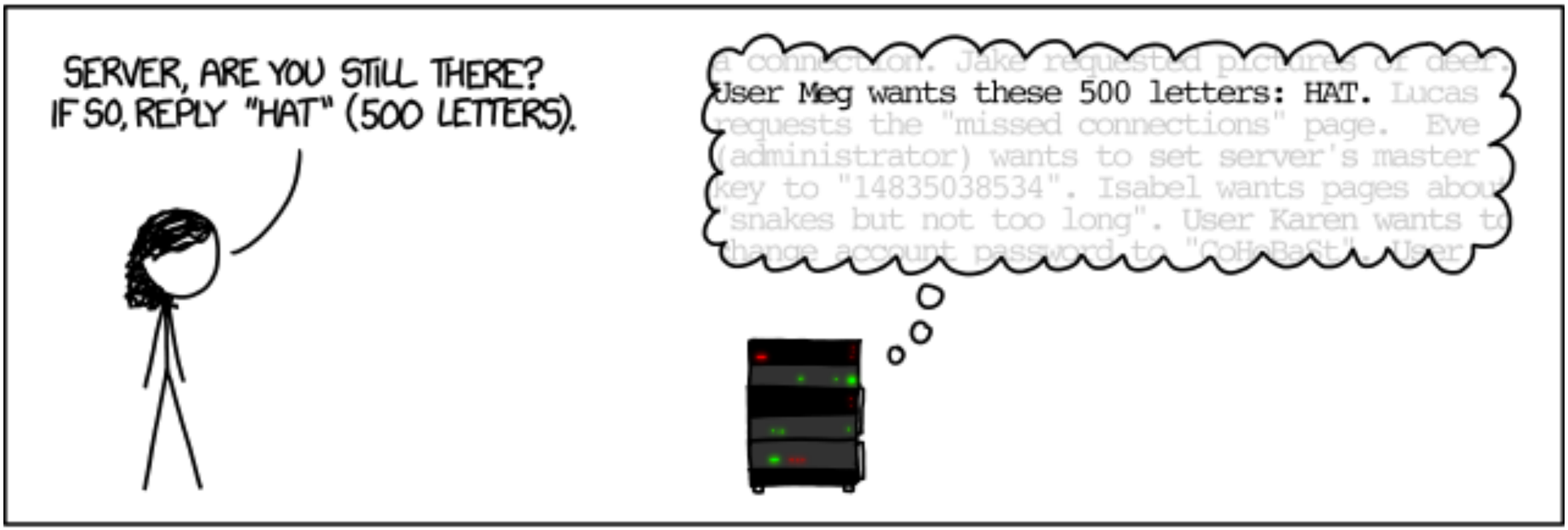
HOW THE HEARTBLEED BUG WORKS:

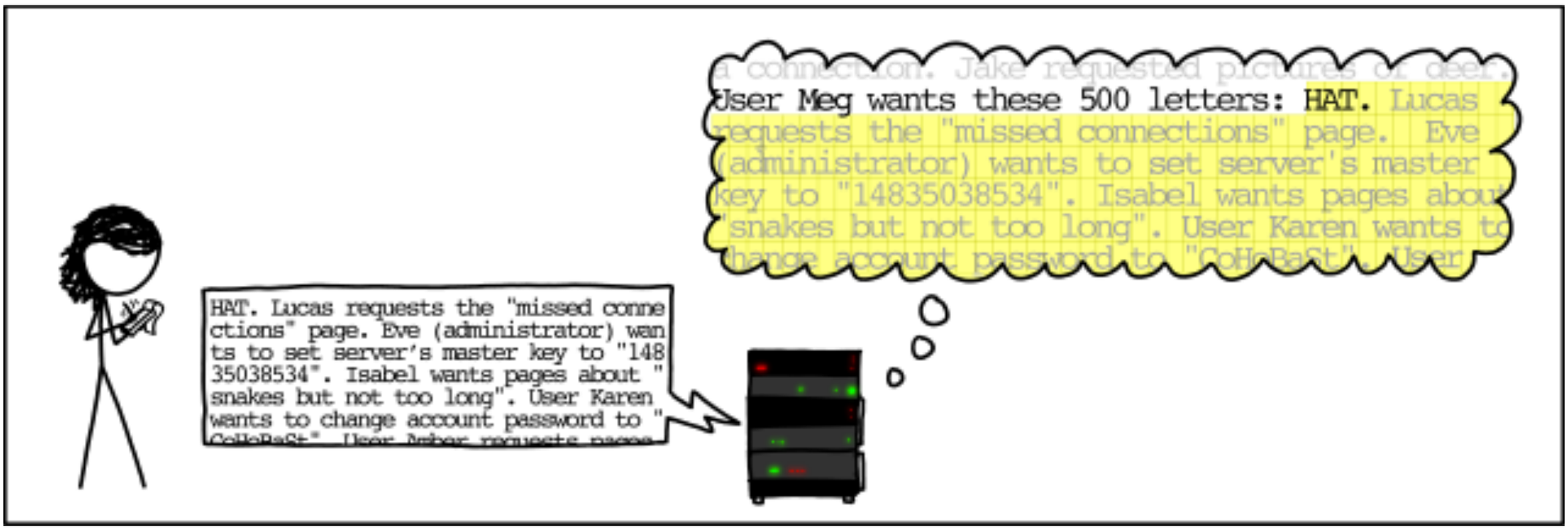



User Meg wants these 6 letters: POTATO.
POTATO


SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).
User Olivia from London wants pages about "ma bees in car why". Note: Files for IP 375.381. 283.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 346 connections open. User Brendan uploaded the file selfie.jpg (contents: 834ba962e2ceb9ff89bd3bff8


HMM...
User Olivia from London wants pages about "man bees in car why". Note: Files for IP 375.381. 283.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 346 connections open. User Brendan uploaded the file selfie.jpg (contents: 834ba962e2cab9ff89bd3bff84
BIRD


SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).
User Meg wants these 500 letters: HAT.


User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoPaSt". User
HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "

wasn't found by code analysis (but by fuzzing) but could have been!


Photo
Graph
Timer
Speed
Value
Doctor
Info

Contact

```
void main() {
        a = new A();
        b = a.g;
        foo(a);
        sink(b.f);
}
```

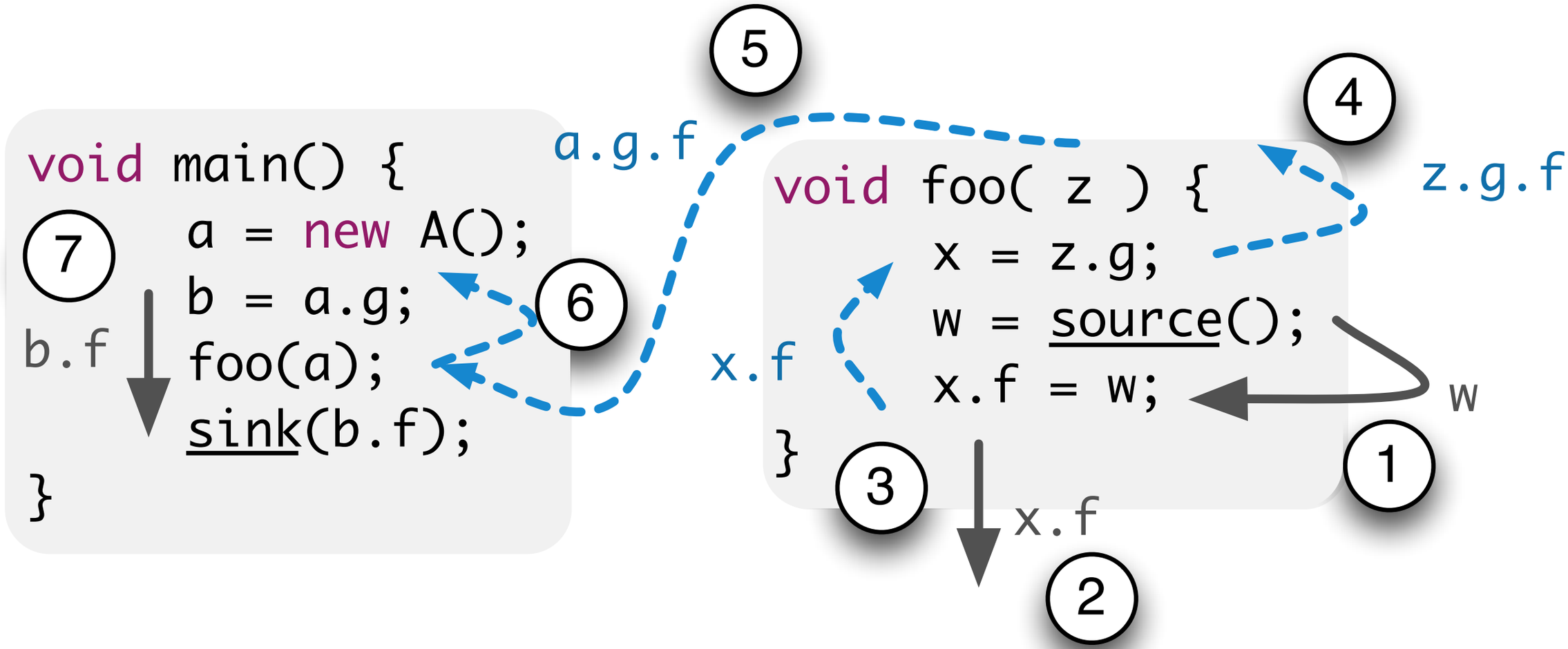
```
void foo( z ) {
        x = z.g;
        w = source();
        x.f = w;
}
```

Email

```
void main() {
        a = new A();
        b = a.g;
        foo(a);
        sink(b.f);
}
```

```
void foo( z ) {
        x = z.g;
        w = source();
        x.f = w;
}
```

Will it leak?


void main() {
a = new A();
b = a.g;
foo(a);
sink(b.f);
}
void foo(z) {
x = z.g;
w = source();
x.f = w;
}
a.g.f
z.g.f
x.f
w
b.f
x.f
1
2
3
4
5
6
7

Will it leak?

Topic of this course

- Static code analysis:
  - We typically don't run the code.
  - We assume that we know the code but nothing else. (maybe a few configuration files)
- Will teach you how to analyze real, large-scale programs:
  - 1,000,000+ lines of code
  - loops, procedures, recursion, reflection, you name it…

DECA vs. ICA

- DECA (this lecture): focus on analysis <u>design</u>
  - Foundational concepts
  - Algorithms and frameworks
  - What works when? How does it work and why?

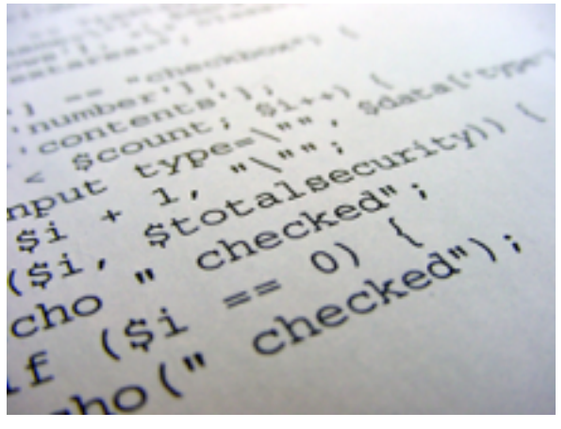
- ICA: focus on analysis <u>implementation</u>
  - Experiment with concrete analysis implementations
  - Implement your own analysis
  - Learning by doing

DECA vs. ICA

- DECA (this lecture): focus on analysis <u>design</u>
  - Foundational concepts
  - Algorithms and frameworks
  - What works when? How does it work and why?

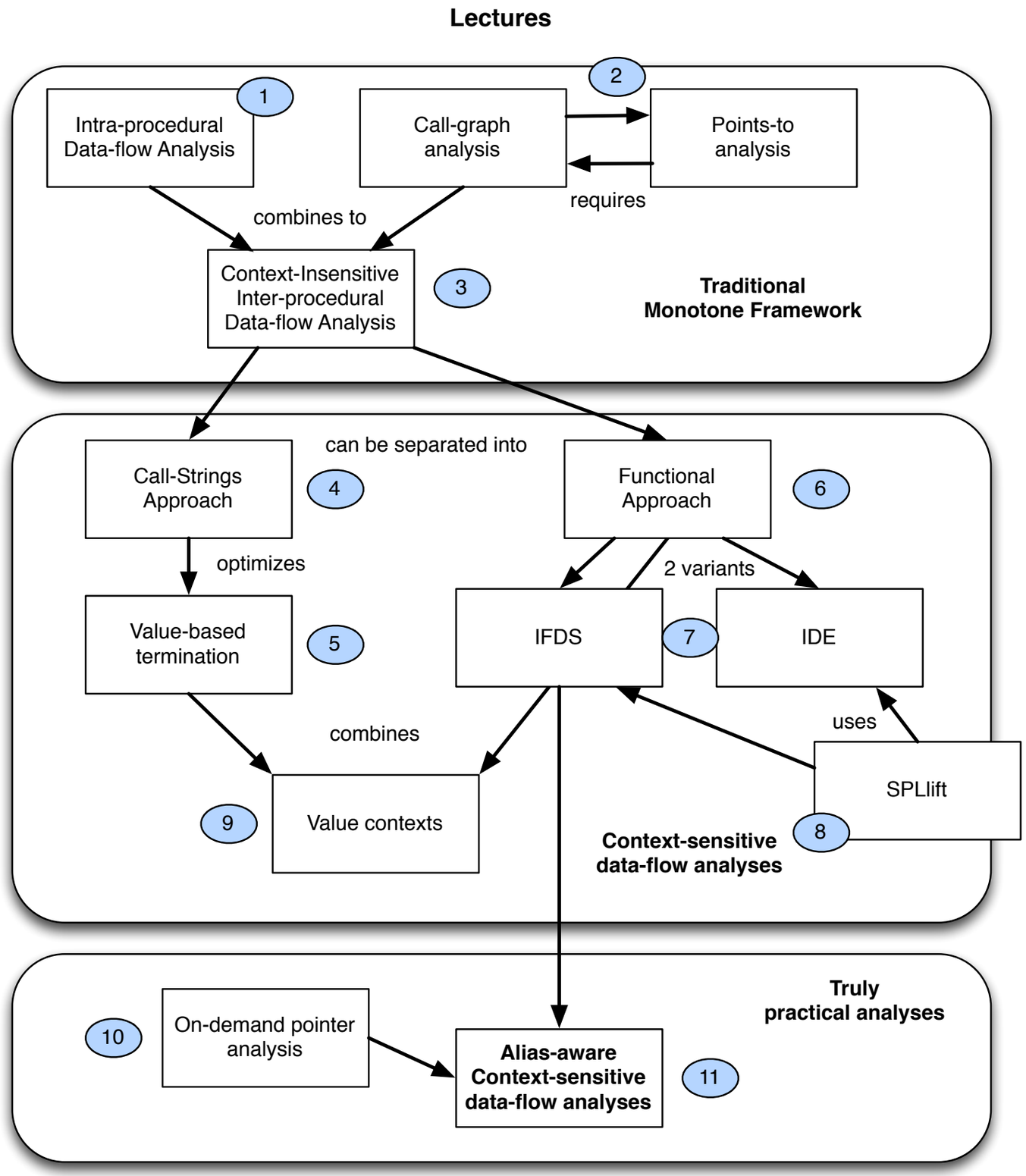
depends on

- ICA: focus on analysis <u>implementation</u>
  - Experiment with concrete analysis implementations
  - Implement your own analysis
  - Learning by doing



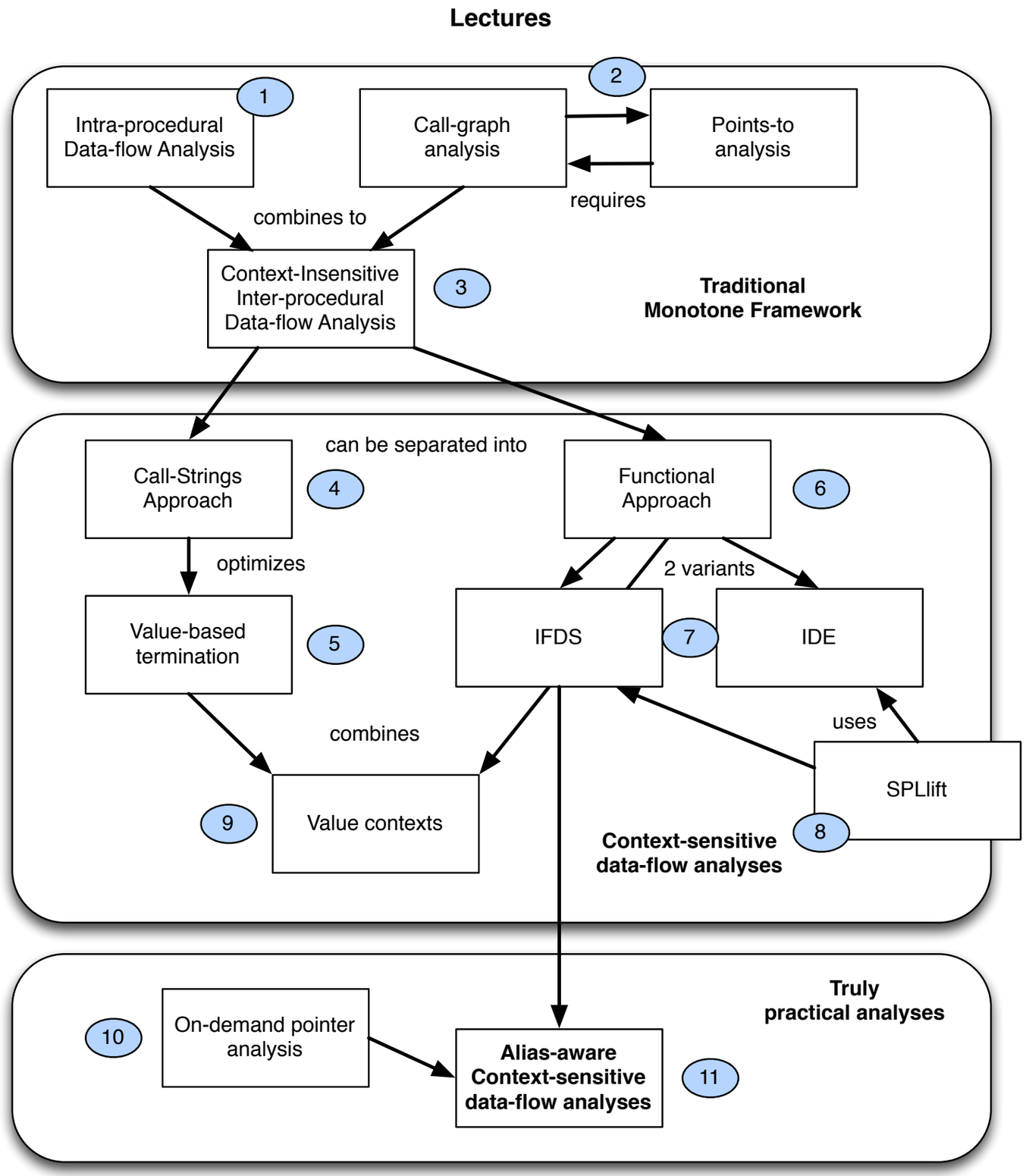
Lecture outline

Preliminary Outline



Preliminary Outline

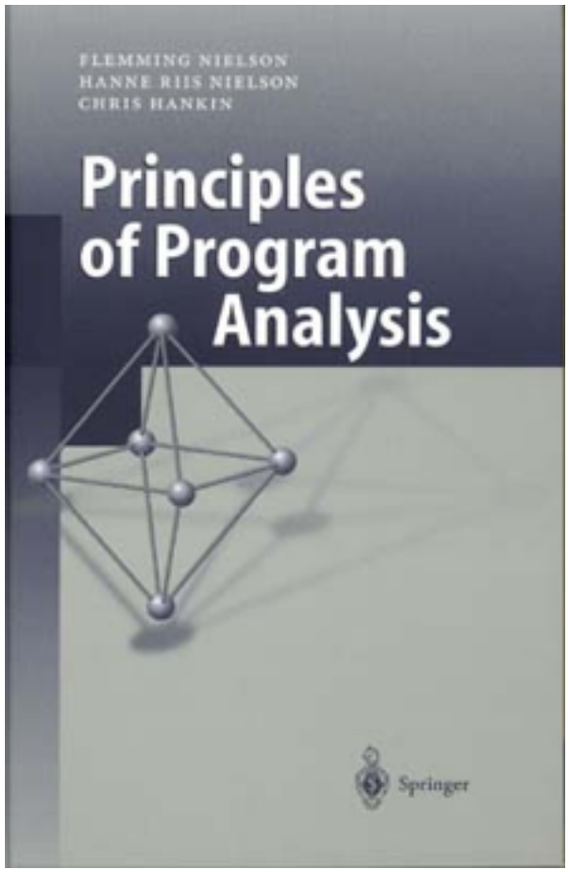
+ special topics



This will be a practical lecture

- Many examples
- Important algorithms and ideas behind them
  - Why do they work? When do they work best?
- Motivated by examples
- Proofs only where they aid understanding

For further reading...

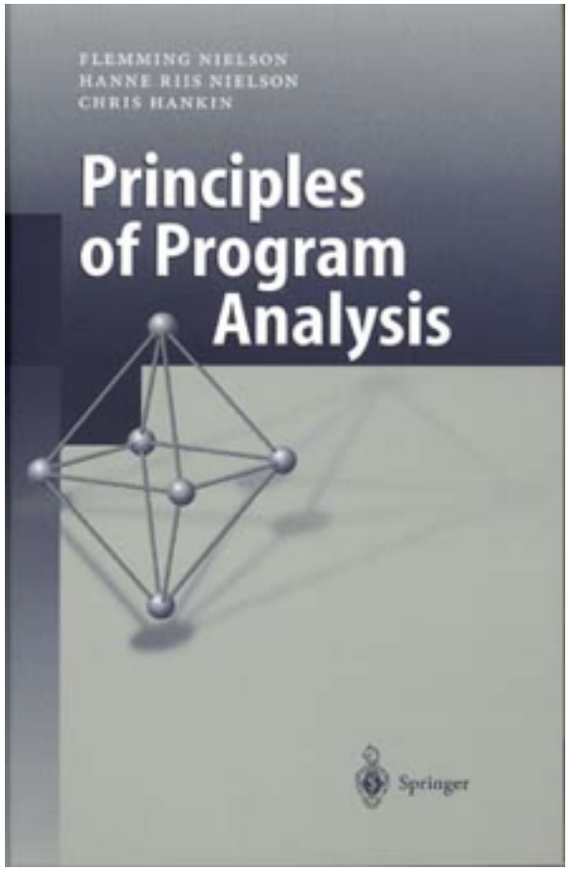


Quite formal

Focuses on

“call-strings approach”

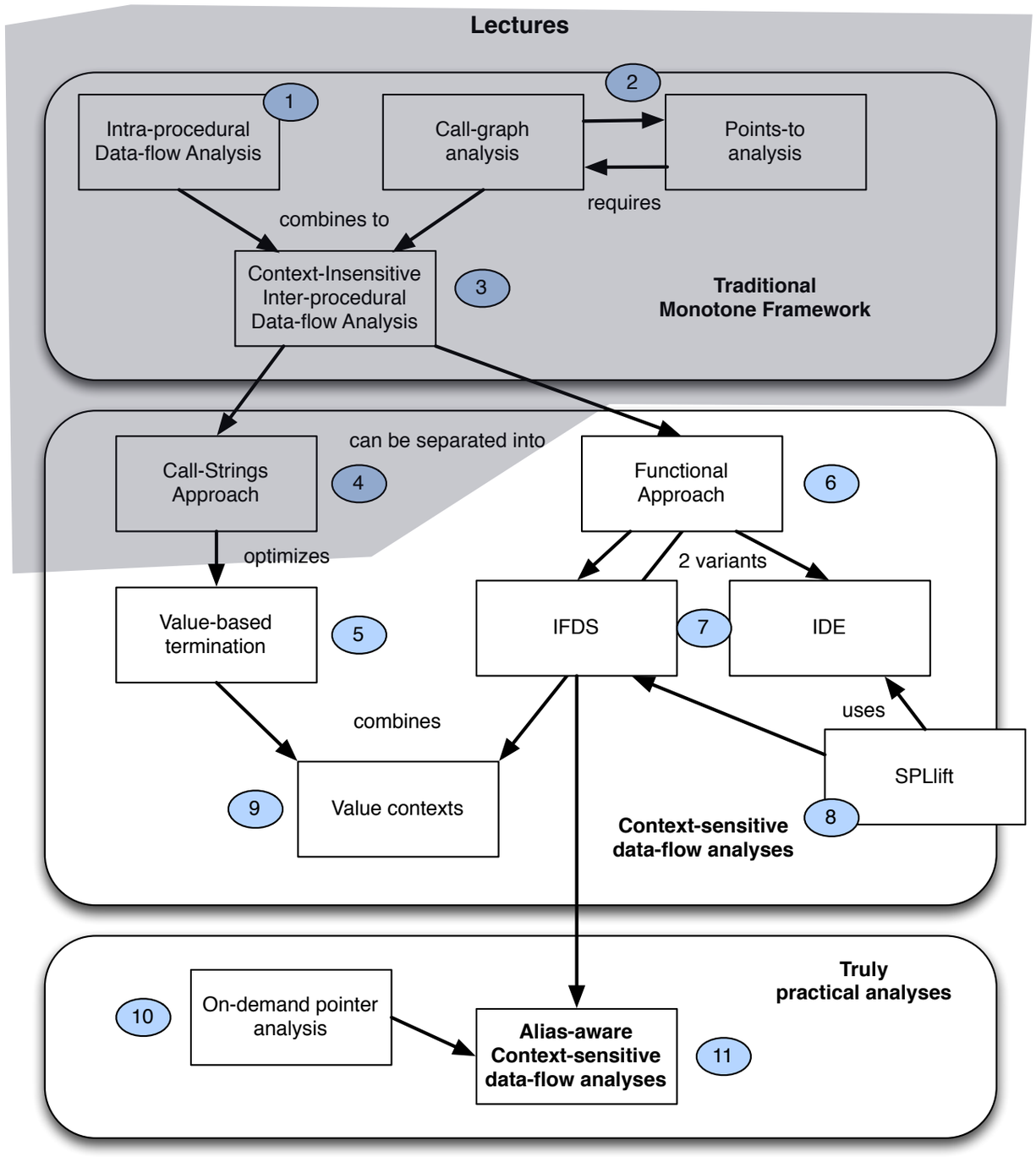
For further reading...



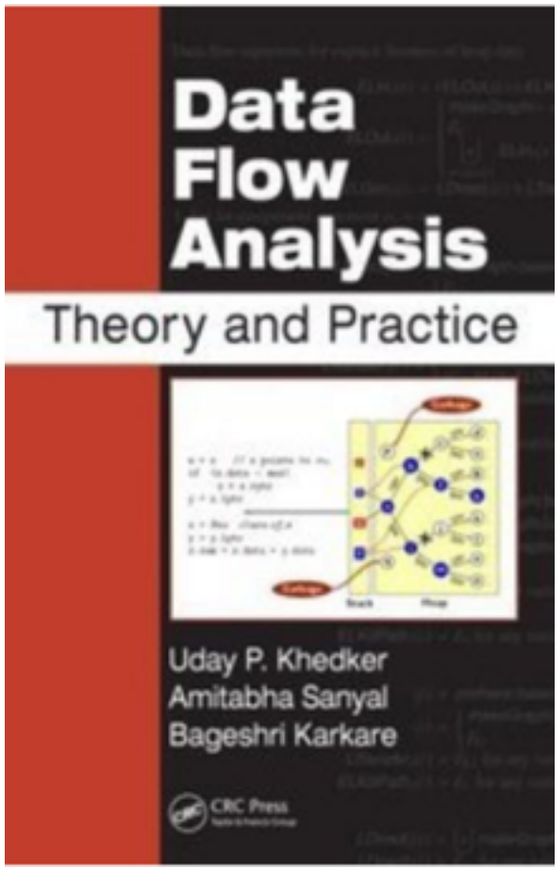
Quite formal

Focuses on

“call-strings approach”



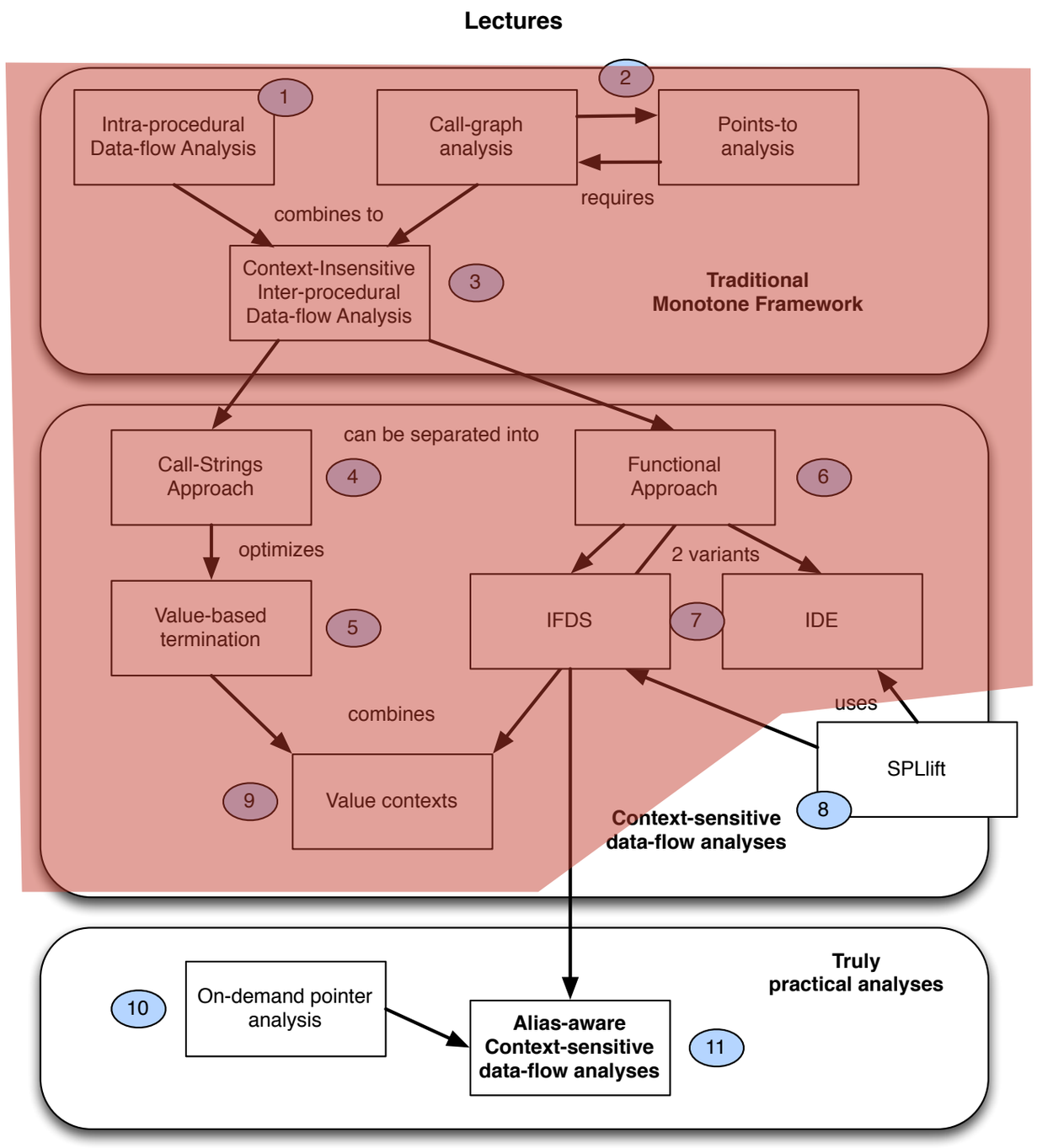
For further reading...



More applied

Focuses on

“functional approach”



Course Setup

- About 90 minutes each Wednesday
- Regular exercise sheets
- When exercises are due:
  - 20-30min discussion of exercises
- Final exam

Signing up

- In TUCaN, sign up both for the module and for the lecture
- Module: 20-00-0771
- Lecture: 20-00-0771-iv

Course SVN repository

- https://repository.st.informatik.tu-darmstadt.de/sse/deca/2014
  - public/
    - exercise-sheets/
    - slides/
    - students/ - your private space for submission of solutions

NOTE: SVN can only be reached from TU network!
Use VPN as required.

Exercises

- There will be 4 exercise sheets
- Each sheet up to 15 points each
  => Maximum of 60 points
- Bonus = (total points awarded) / 60
  - rounded up to next grade level
- Maximal bonus: 1.0
- Final grade = exam grade + bonus

Exercises

- Bonus cannot be used to pass exam!
- Exercises usually due on Monday before next sheet is given out (23:59).
  - Due date is stated on sheet
- Complete exercises in groups of 4 people
- Hand in using Version Control, not Email!

Exercises - SVN Setup

- Find group partners. (Here or using forum.)
  - Forum available at www.d120.de
  - There should be four people per group.
- Email the following data to steven.arzt@cased.de
  - your directory name, your names, your Student ID numbers, your RBG login names (if different), and your email addresses (!)
- We will create a secured directory for you at: `https://repository.st.informatik.tu-darmstadt.de/sse/deca/2014/students/`
- We will then secure your directory and email you back.

Exercises - SVN Setup

- Find group partners. (Here or using forum.)
  - Forum available at www.d120.de
  - There should be four people per group.
- Email the following data to steven.arzt@cased.de
  - your directory name, your names, your Student ID numbers, your RBG login names (if different), and your email addresses (!)
- We will create a secured directory for you at: `https://repository.st.informatik.tu-darmstadt.de/sse/deca/2014/students/`
- We will then secure your directory and email you back.

DEADLINE: Fri, Nov 7th

Exercises - Handing in

- Check in certain files stated on exercise sheets
  - often only `solution.pdf`
- No need to email us, just check in by the deadline!
- We will push your grade (pass/fail) and comments into your group directory.

Exercises - Discussion

- For questions please use the forums.
- May also ask questions after each lecture.
- I will try to discuss the solution to each sheet on the day the next sheet is given out.

Course Notes

- There will be no fully-fledged script.
- I will provide, though...
  - all slides, and
  - essential notes, e.g. of algorithms, and links to background reading
- Material will be in SVN, password protected

Contacting us

- Please use the forum!
- No office hours:
  use email or make appointment

Questions?

What we will cover today

Not quite source code, not quite bytecode:

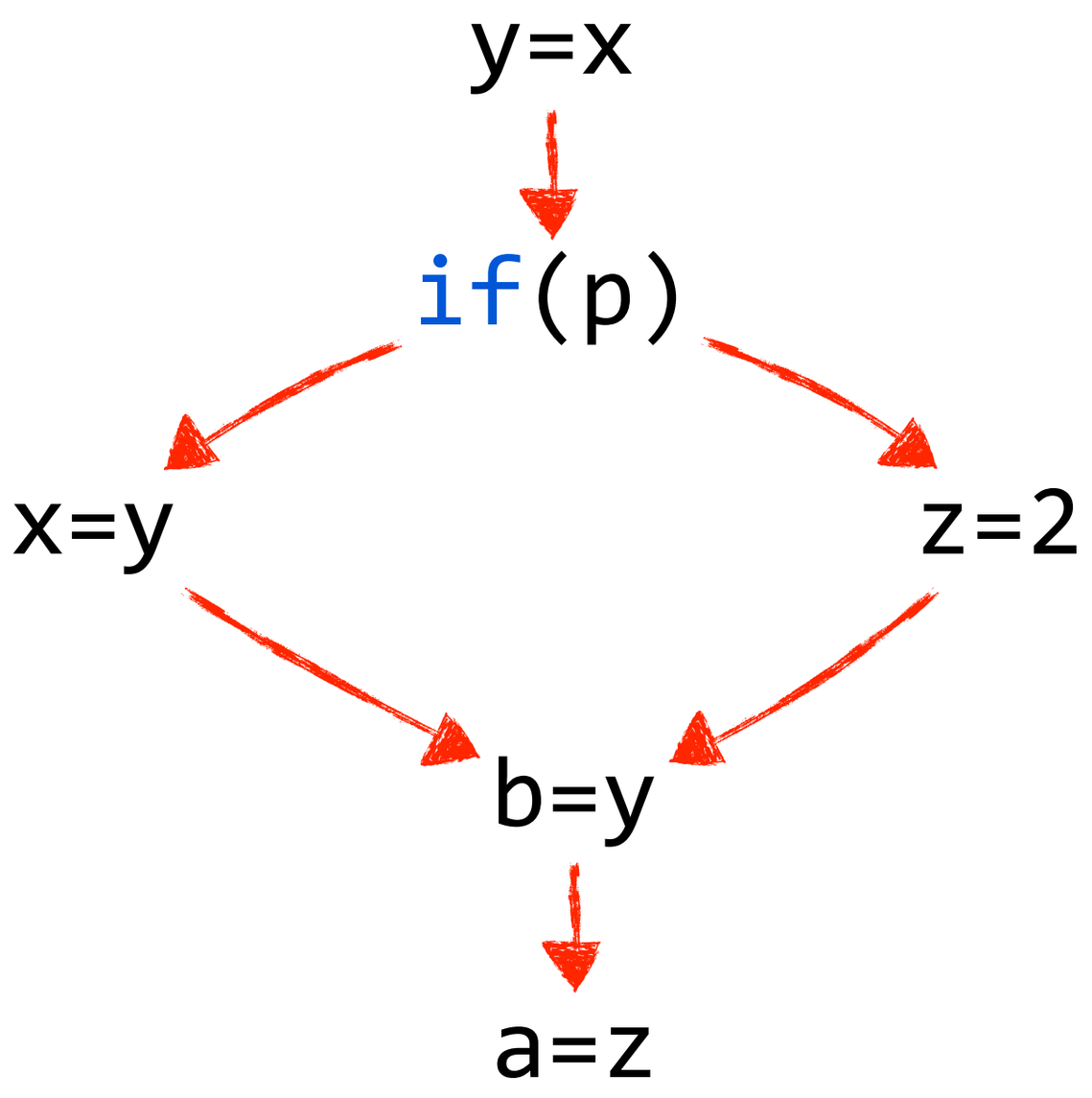
intermediate representations
for static analysis

General Workflow

- Parse method (as source code or bytecode) and convert into control-flow graph (CFG)
  - Nodes: Simplified statements
  - Edges: Possible control flows between such statements

Example

```
y=x;
if(p) x=y;
else z=2;

b=y;
a=z;
```

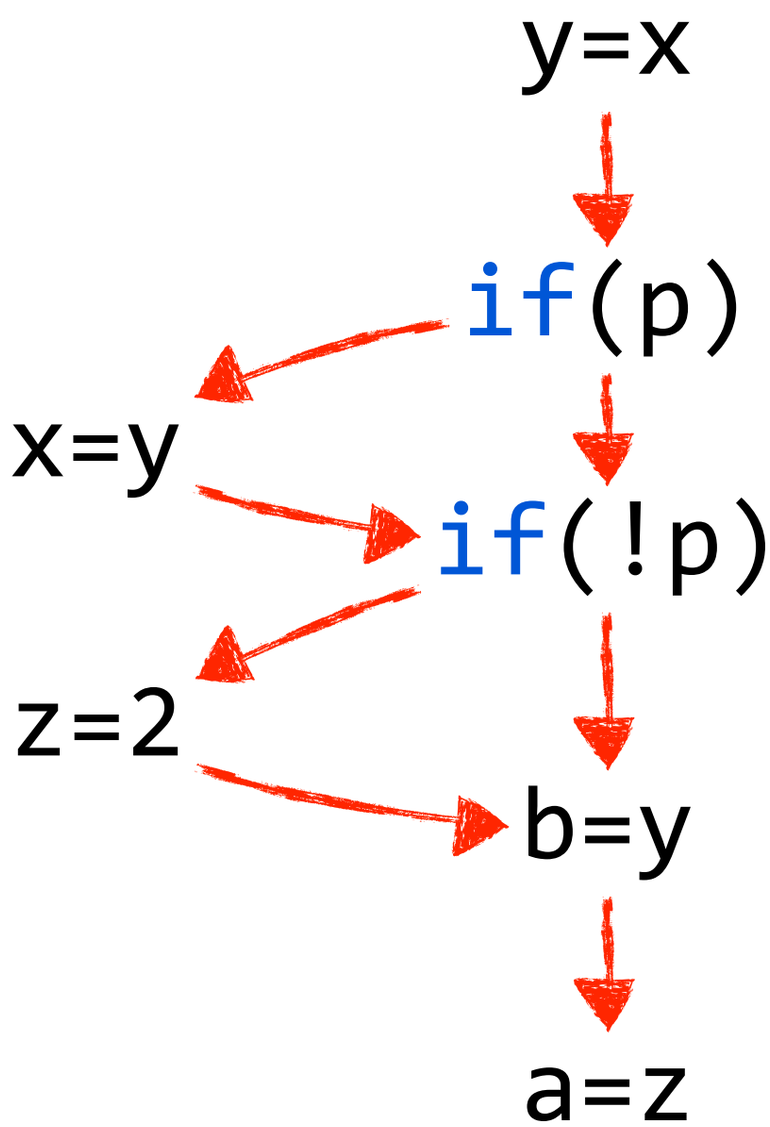


In general, CFG is an over-approximation

```
y=x;
if(p)  x=y;
if(!p) z=2;
b=y;
a=z;
```

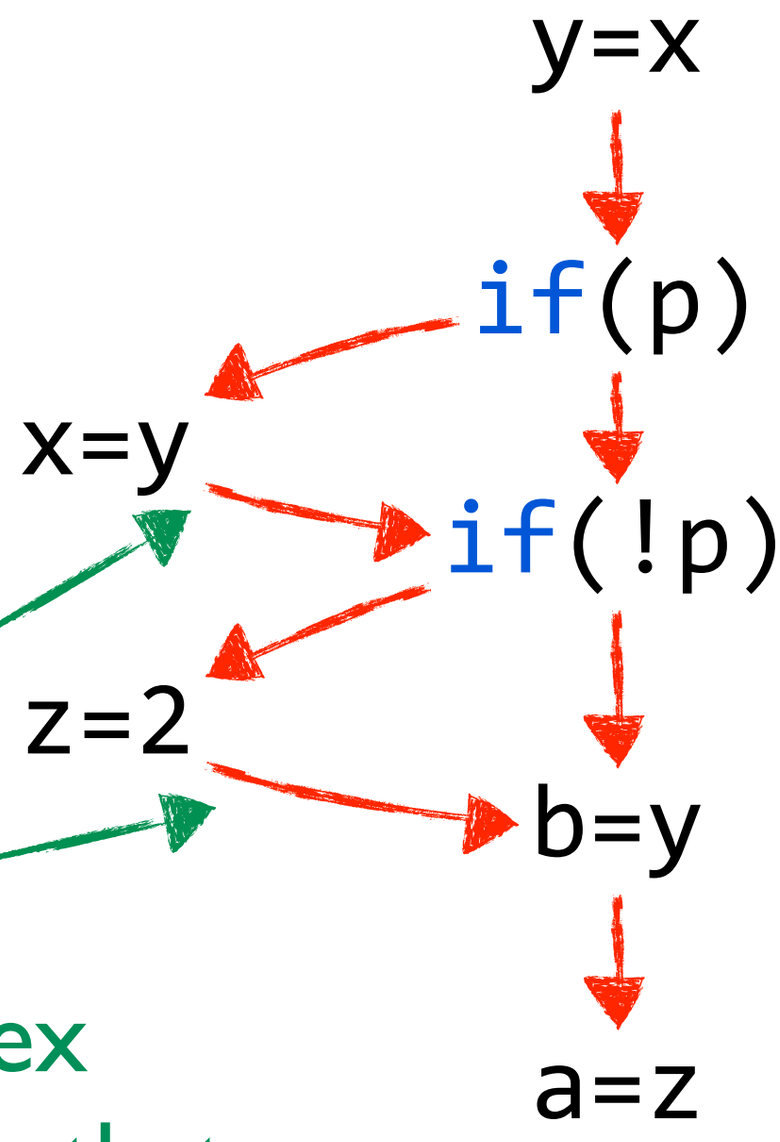
In general, CFG is an over-approximation

```
y=x;
if(p)   x=y;
if(!p) z=2;
b=y;
a=z;
```



In general, CFG is an over-approximation

```
y=x;
if(p)   x=y;
if(!p) z=2;
b=y;
a=z;
```

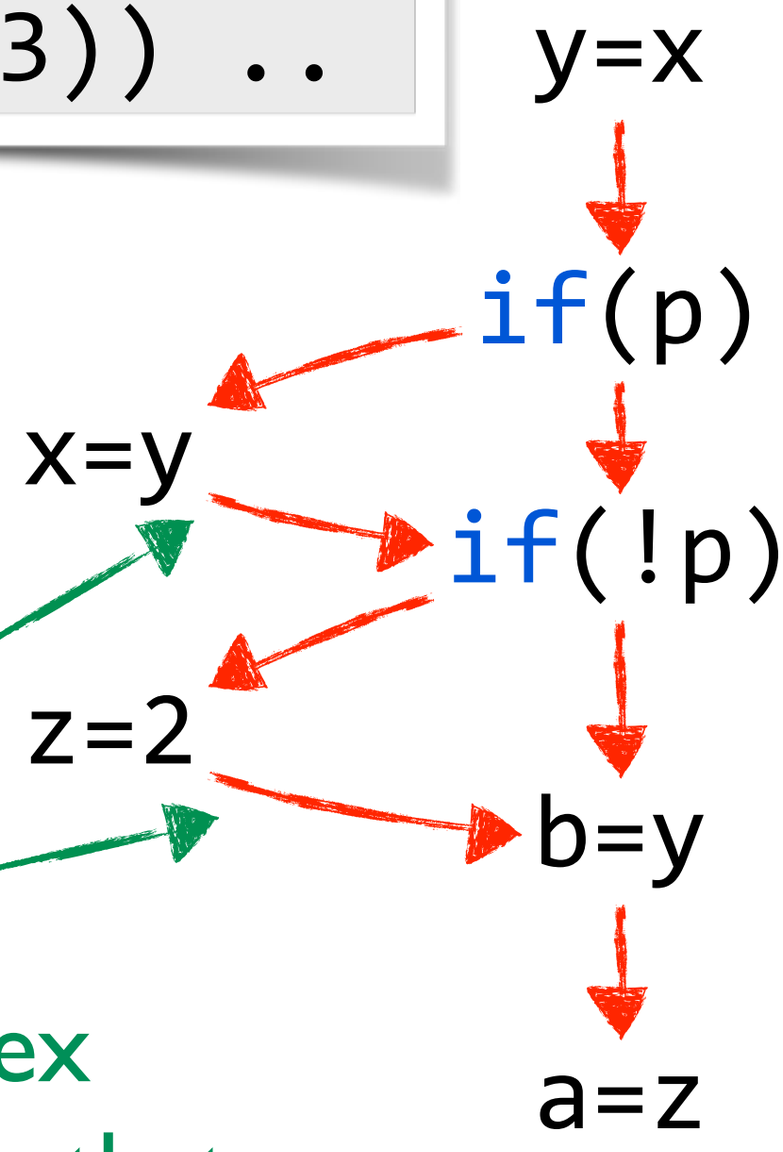


depending on how complex predicate p is, cannot infer that branches are mutually exclusive

In general, CFG is an over-approximation

```
if(isPrime(2312321)) ..
if(isPrime(2312323)) ..
```

```
y=x;
if(p)  x=y;
if(!p) z=2;
b=y;
a=z;
```



depending on how complex predicate p is, cannot infer that branches are mutually exclusive

Lesson learned

- Almost always, control-flow graphs are conservative:
  - if control may flow from statement a to statement b then there is an edge from a to b
  - opposite is not true!
  - this problem cannot be solved (undecidable)
- Real-life CFGs will even contain edges for exceptional control flow (otherwise unsound)

Important design decision

What statements/nodes to allow or not?

One extreme: Java source code

```
void bar(int i) {
  int x, y;
  if(i>0) {
    x = 0;
  } else {
    y = 0;
  }
  System.out.println(x);
}
```

Eclipse:
value of x undefined

One extreme: Java source code

Problem: statements (and classes) can be nested...

```
for(..) {
 for(..) {
   }
}
```

One extreme: Java source code

Problem: statements (and classes) can be nested...

```
for(..) {
 for(..) {
    new Comparator() {
       public int compareTo(..) {
            ... and so on
       }
    }
 }
}
```

Other extreme: Java bytecode

- Advantages:
  - no nesting; one statement follows the other; looping/ branches through jumps (goto)
  - nested classes are "flattened" into normal classes
- Disadvantages:
  - No local variables: operations performed on operand stack
  - More than 200 possible bytecodes!

Other extreme: Java bytecode

```
void foo() {
  double d1 = 3.0;
  double d2 = 2.0;
  int i1 = (int) (d1*d2);
  bar(this,i1);
}
```

Other extreme: Java bytecode

```
ldc2_w #15; //double 3.0d
dstore_1
ldc2_w #17; //double 2.0d
dstore_3
dload_1
dload_3
dmul
d2i
istore 5
aload_0
aload_0
iload 5
invokespecial   #19; //Method bar:
(LMain;I)V
return
```

```
void foo() {
  double d1 = 3.0;
  double d2 = 2.0;
  int i1 = (int) (d1*d2);
  bar(this,i1);
}
```

Other extreme: Java bytecode

```
ldc2_w #15; //double 3.0d
dstore_1
ldc2_w #17; //double 2.0d
dstore_3
dload_1
dload_3
dmul
d2i
istore 5
aload_0
aload_0
iload 5
invokespecial #19; //Method bar:
(LMain;I)V
return
```

```
void foo() {
  double d1 = 3.0;
  double d2 = 2.0;
  int i1 = (int) (d1*d2);
  bar(this,i1);
}
```

pop and multiply two top operands on the stack;
place result on stack again

Other extreme: Java bytecode

```
ldc2_w #15; //double 3.0d
dstore_1
ldc2_w #17; //double 2.0d
dstore_3
dload_1
dload_3
dmul
d2i
istore 5
aload_0
aload_0
iload 5
invokespecial   #19; //Method bar:
(LMain;I)V
return
```

```
void foo() {
  double d1 = 3.0;
  double d2 = 2.0;
  int i1 = (int) (d1*d2);
  bar(this,i1);
}
```

many overloaded versions of essentially the same operation

Android Bytecode

- Similar to Java bytecode but...
- Logical registers instead of operand stack
- Some values are untyped
  - example: the type of numerical constants is not known before their first use
  - `0` can mean `0` or `0f` or `null`
- Roughly 250 bytecodes
  - including Optimized DEX (ODEX)

Android Bytecode

- Similar to Java bytecode but...
- Logical registers instead of operand stack
- Some values are untyped
  - example: the type of numerical constants is not known before their first use
  - `0` can mean `0` or `0f` or `null`
- Roughly 250 bytecodes
  - including Optimized DEX (ODEX)

→First exercise sheet

Example Intermediate Representation: Jimple

- Jimple = "like **J**ava, but s**imple**"
- Combines the best of both worlds
  - Local variables, like in source code
  - no stack operations
  - Special variables for "this" and parameters
  - Only simple statements, never nested

Golden mean: Jimple IR

```
void foo()
{
    Main this;
    double d1, d2, temp$0;
    int i1;

    this := @this: Main;
    d1 = 3.0;
    d2 = 2.0;
    temp$0 = d1 * d2;
    i1 = (int) temp$0;
    virtualinvoke this.<Main: void bar(Main,int)>(this, i1);
    return;
}
```

```
void foo() {
  double d1 = 3.0;
  double d2 = 2.0;
  int i1 = (int) (d1*d2);
  bar(this,i1);
}
```

Golden mean: Jimple IR

```
void foo()
{
    Main this;
    double d1, d2, temp$0;
    int i1;

    this := @this: Main;
    d1 = 3.0;
    d2 = 2.0;
    temp$0 = d1 * d2;
    i1 = (int) temp$0;
    virtualinvoke this.<Main: void bar(Main,int)>(this, i1);
    return;
}
```

```
void foo() {
  double d1 = 3.0;
  double d2 = 2.0;
  int i1 = (int) (d1*d2);
  bar(this,i1);
}
```

all variables explicitly declared, even “this”

Golden mean: Jimple IR

```
void foo()
{
    Main this;
    double d1, d2, temp$0;
    int i1;

    this := @this: Main;
    d1 = 3.0;
    d2 = 2.0;
    temp$0 = d1 * d2;
    i1 = (int) temp$0;
    virtualinvoke this.<Main: void bar(Main,int)>(this, i1);
    return;
}
```

```
void foo() {
  double d1 = 3.0;
  double d2 = 2.0;
  int i1 = (int) (d1*d2);
  bar(this,i1);
}
```

special references for “this” and parameters

Golden mean: Jimple IR

```
void foo()
{
    Main this;
    double d1, d2, temp$0;
    int i1;

    this := @this: Main;
    d1 = 3.0;
    d2 = 2.0;
    temp$0 = d1 * d2;
    i1 = (int) temp$0;
    virtualinvoke this.<Main: void bar(Main,int)>(this, i1);
    return;
}
```

```
void foo() {
  double d1 = 3.0;
  double d2 = 2.0;
  int i1 = (int) (d1*d2);
  bar(this,i1);
}
```

no stack operations; instead assignments

Golden mean: Jimple IR

```
void foo()
{
    Main this;
    double d1, d2, temp$0;
    int i1;

    this := @this: Main;
    d1 = 3.0;
    d2 = 2.0;
    temp$0 = d1 * d2;
    i1 = (int) temp$0;
    virtualinvoke this.<Main: void bar(Main,int)>(this, i1);
    return;
}
```

```
void foo() {
  double d1 = 3.0;
  double d2 = 2.0;
  int i1 = (int) (d1*d2);
  bar(this,i1);
}
```

1:n

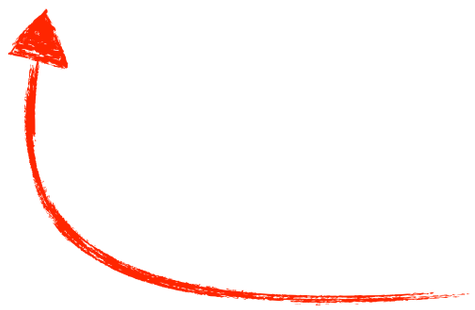
"complex" statements broken down

at most one reference on left-hand side,
at most two references on right-hand side
=> "three-address code"

Golden mean: Jimple IR

```
void foo()
{
    Main this;
    double d1, d2, temp$0;
    int i1;

    this := @this: Main;
    d1 = 3.0;
    d2 = 2.0;
    temp$0 = d1 * d2;
    i1 = (int) temp$0;
    virtualinvoke this.<Main: void bar(Main,int)>(this, i1);
    return;
}
```

```
void foo() {
  double d1 = 3.0;
  double d2 = 2.0;
  int i1 = (int) (d1*d2);
  bar(this,i1);
}
```



method calls fully resolved,
explicit “this” reference

Java Bytecode vs. Jimple

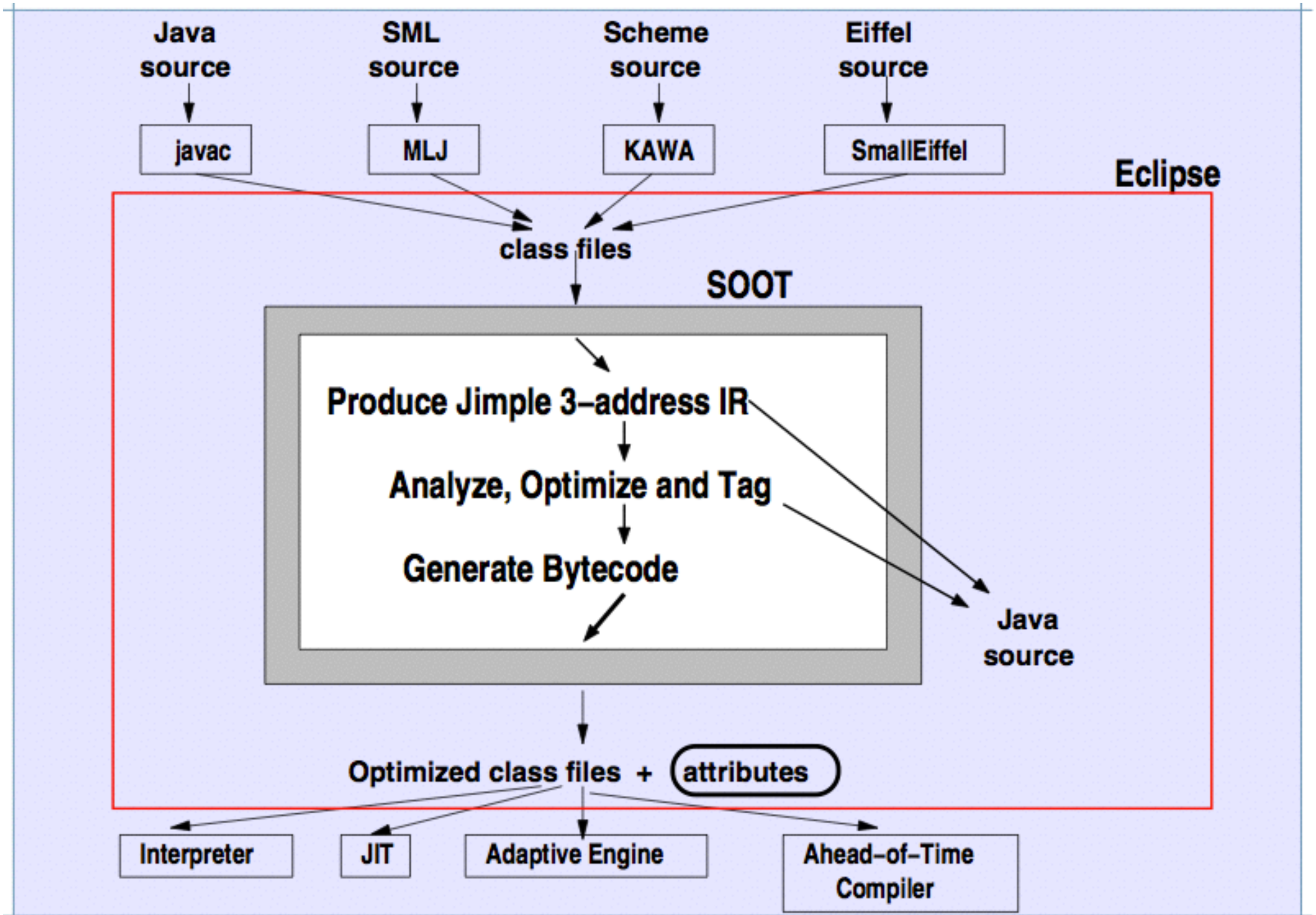
Bytecode	Jimple
each instr. has implicit effect on stack	each stmt. acts explicitly on named variables
no types for stack locations	types for local variables
>200 kinds of instructions	15 types of statements

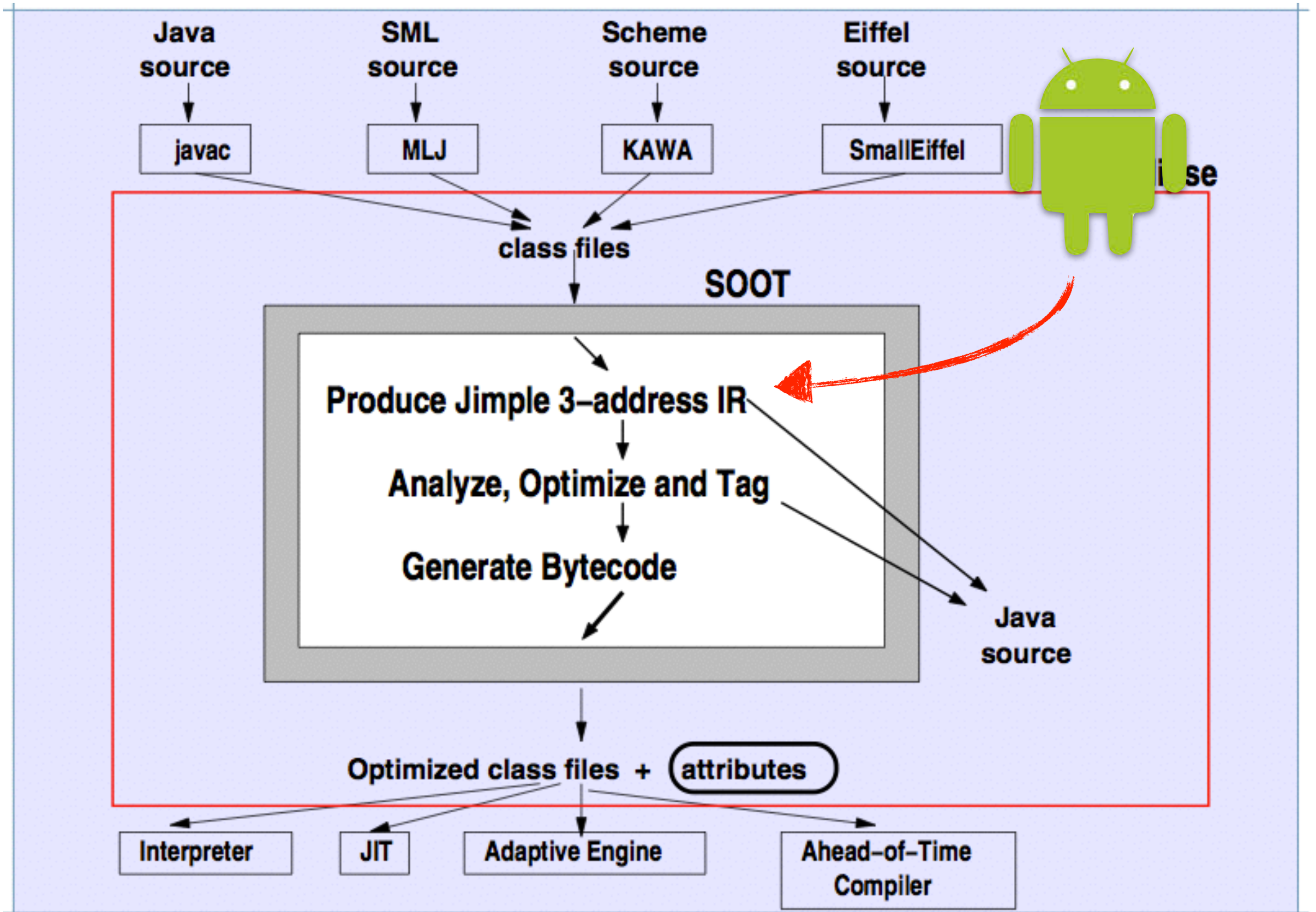
Jimple is part of Soot

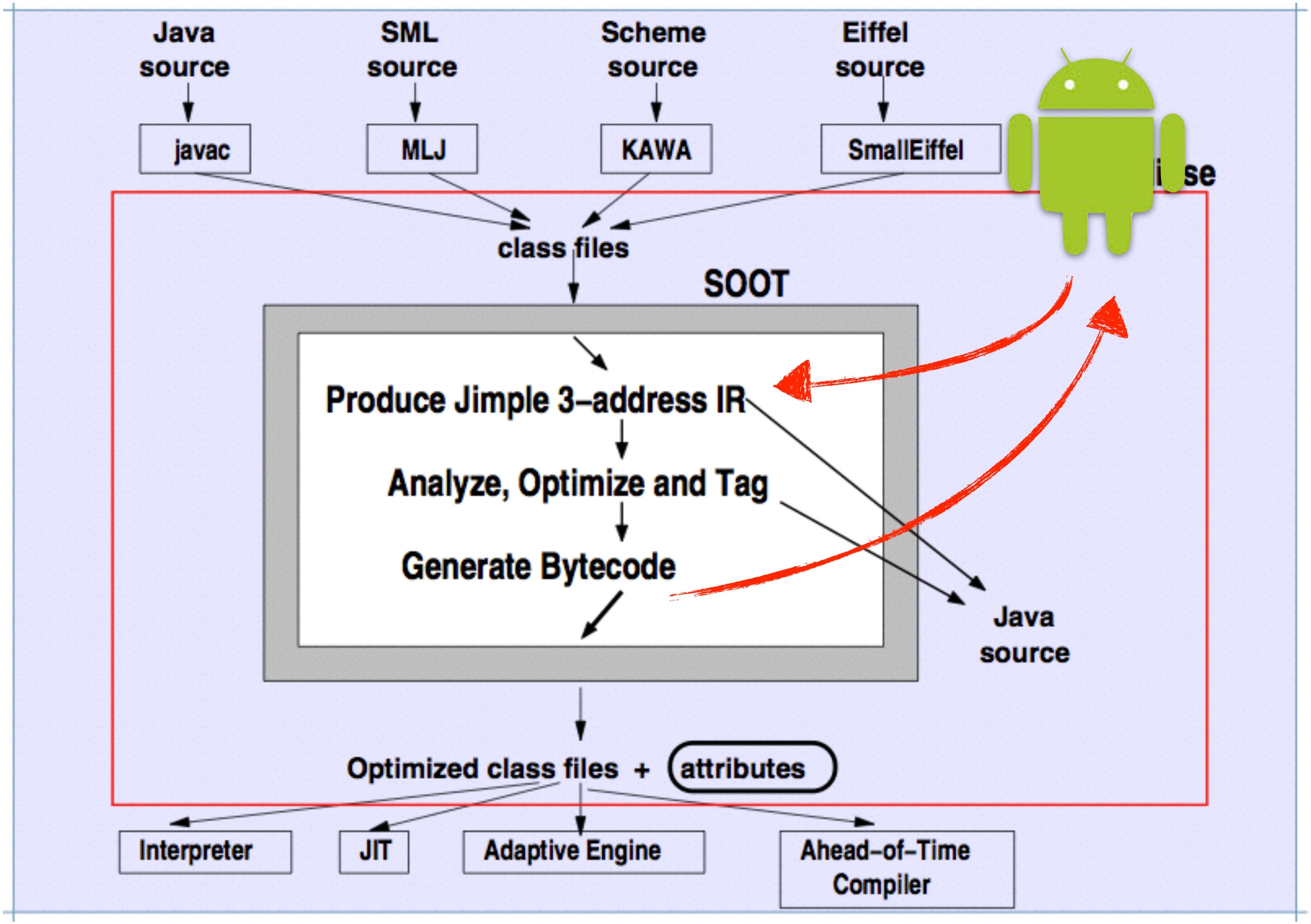
- a free compiler infrastructure, written in Java
  - under LGPL; active open-source community
- was originally designed to analyze and transform Java bytecode
- original motivation was to provide a common infrastructure with which researchers could compare analyses (points-to analyses)
- has been extended to include decompilation and visualization

Soot

- Soot has many potential applications:
  - used as a stand-alone tool (command line or Eclipse plugin)
  - extended to include new IRs, analyses, transformations and visualizations
  - as the basis of building new special-purpose tools


Java source
SML source
Scheme source
Eiffel source
javac
MLJ
KAWA
SmallEiffel
Eclipse
class files
SOOT
Produce Jimple 3-address IR
Analyze, Optimize and Tag
Generate Bytecode
Java source
Optimized class files + attributes
Interpreter
JIT
Adaptive Engine
Ahead-of-Time Compiler


Java source
SML source
Scheme source
Eiffel source
javac
MLJ
KAWA
SmallEiffel
class files
SOOT
Produce Jimple 3-address IR
Analyze, Optimize and Tag
Generate Bytecode
Java source
Optimized class files + attributes
Interpreter
JIT
Adaptive Engine
Ahead-of-Time Compiler


Java source
SML source
Scheme source
Eiffel source
javac
MLJ
KAWA
SmallEiffel
class files
SOOT
Produce Jimple 3-address IR
Analyze, Optimize and Tag
Generate Bytecode
Java source
Optimized class files + attributes
Interpreter
JIT
Adaptive Engine
Ahead-of-Time Compiler

Kinds of Jimple Stmts

- Core statements:
  `NopStmt`
  `DefinitionStmt: IdentityStmt, AssignStmt`

- Intraprocedural control-flow:
  `IfStmt, GotoStmt, TableSwitchStmt, LookupSwitchStmt`

- Interprocedural control-flow:
  `InvokeStmt, ReturnStmt, ReturnVoidStmt`

Kinds of Jimple Stmts

- `ThrowStmt`
  throws an exception
- `RetStmt`
  not used; returns from a JSR (deprecated)
- `MonitorStmt: EnterMonitorStmt, ExitMonitorStmt`
  for mutual exclusion (synchronized blocks)

```
this.m();
```

Where's the definition of this?

`IdentityStmt`:

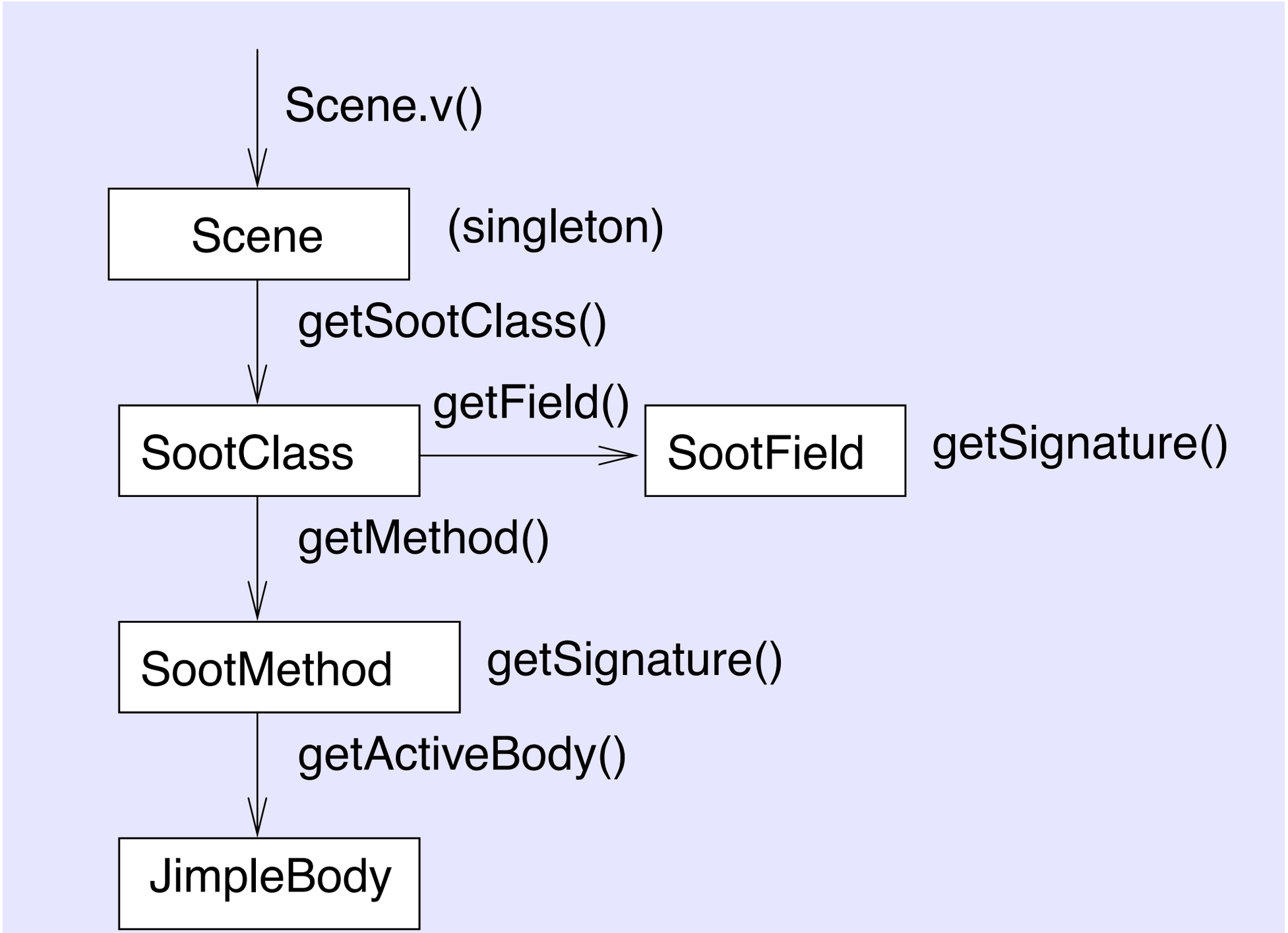
- Used for assigning parameter values and @this-ref to locals.
- Gives each local at least one definition point.

Jimple representation of `IdentityStmts`:

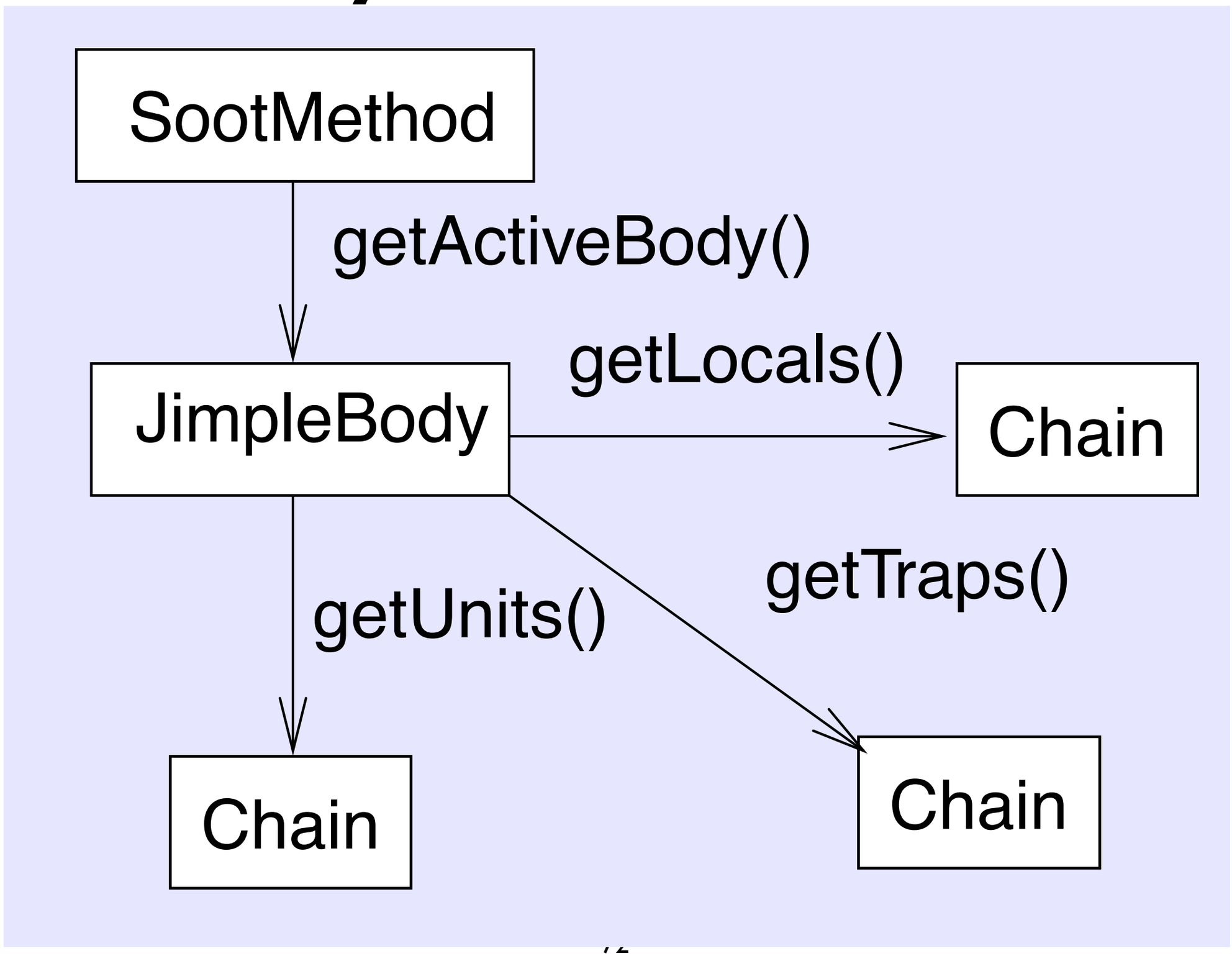
```
r0 := @this;
i1 := @parameter0;
```

```
public int foo(java.lang.String) { // locals
  r0 := @this;                     // IdentityStmt
  r1 := @parameter0;

  if r1 != null goto label0; // IfStmt

  $i0 = r1.length();  // AssignStmt
  r1.toUpperCase();   // InvokeStmt
  return $i0;         // ReturnStmt

label0:               // created by Printer
   return 2;
}
```

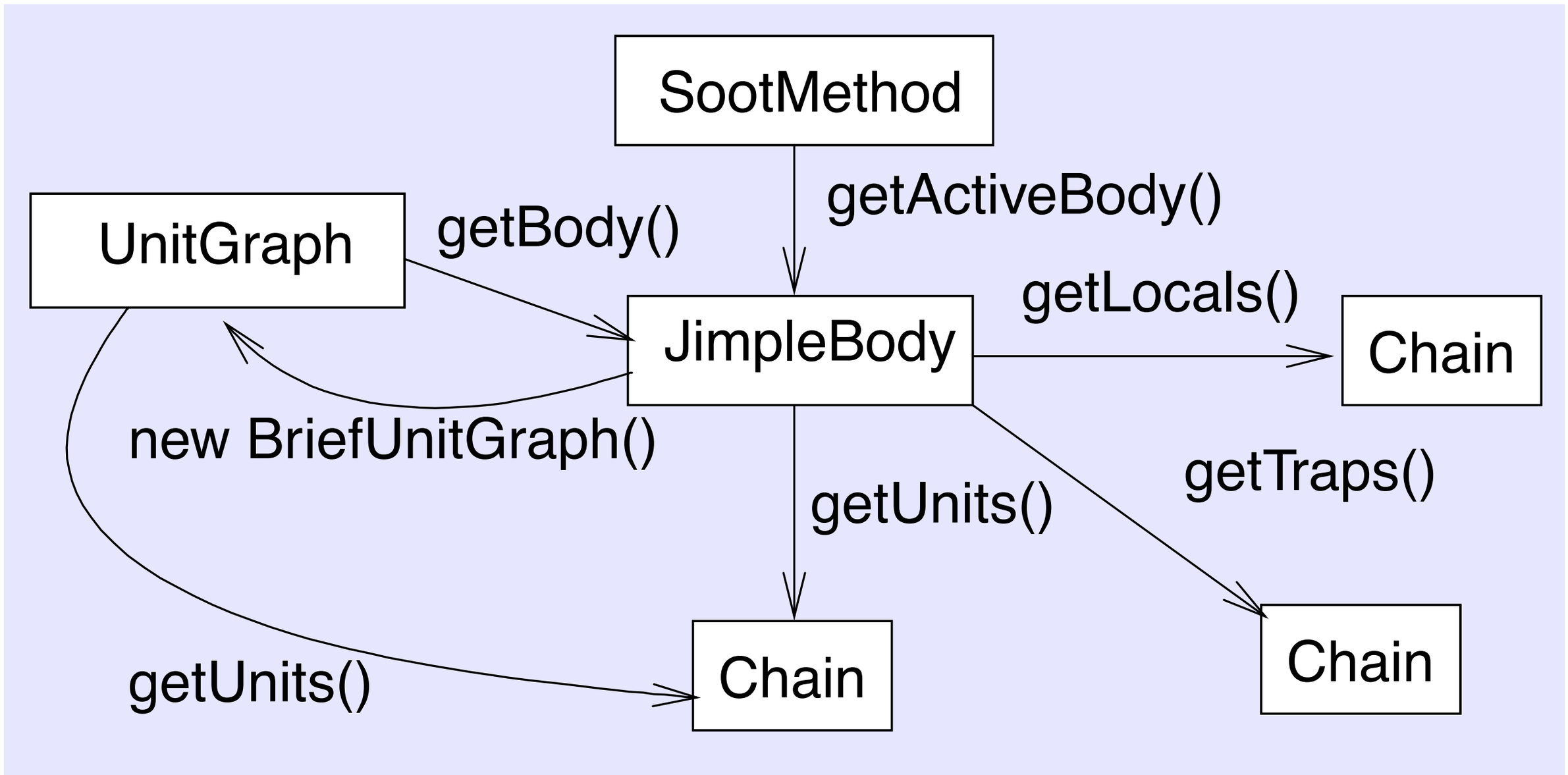
Browsing Jimple



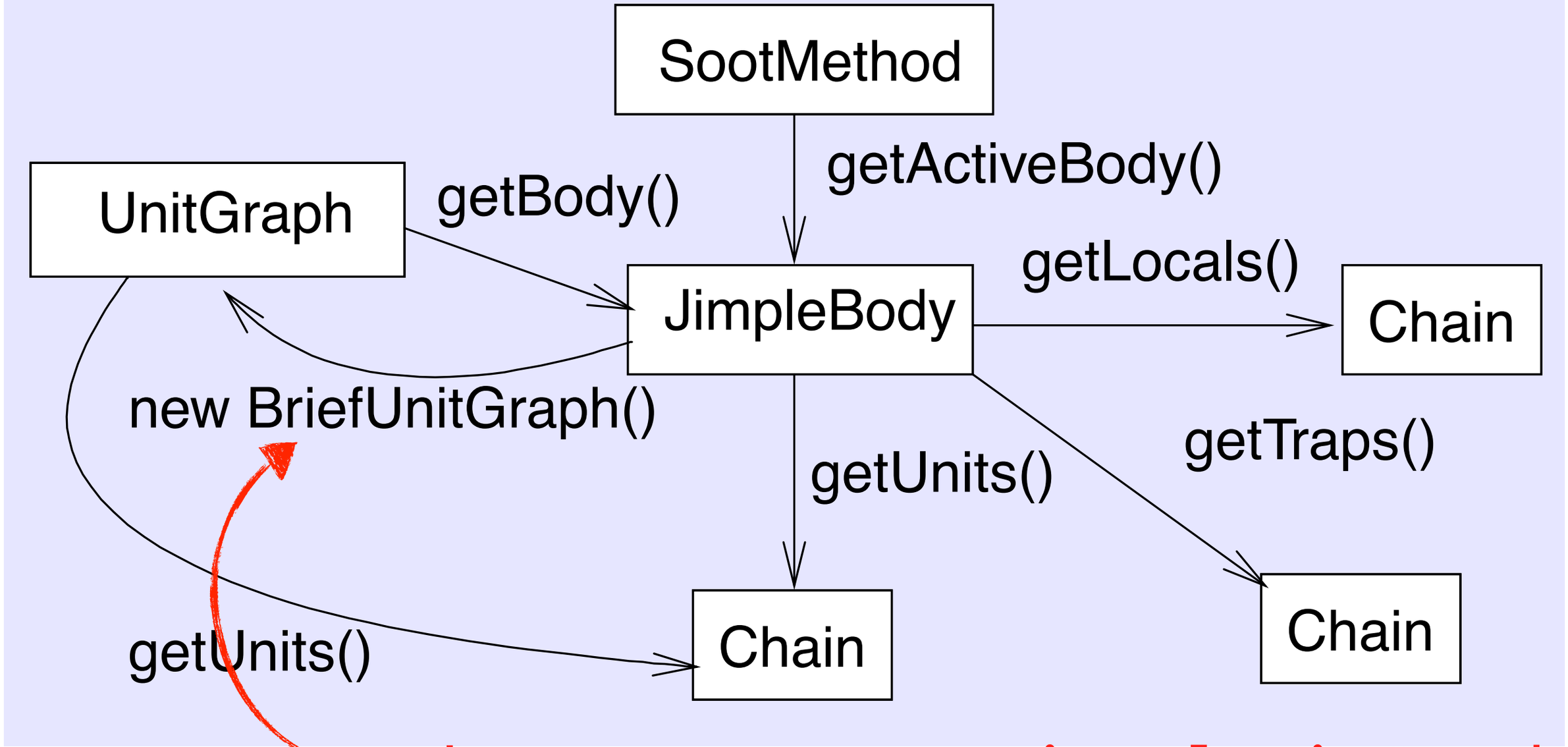
Body-centric view



Getting a CFG...

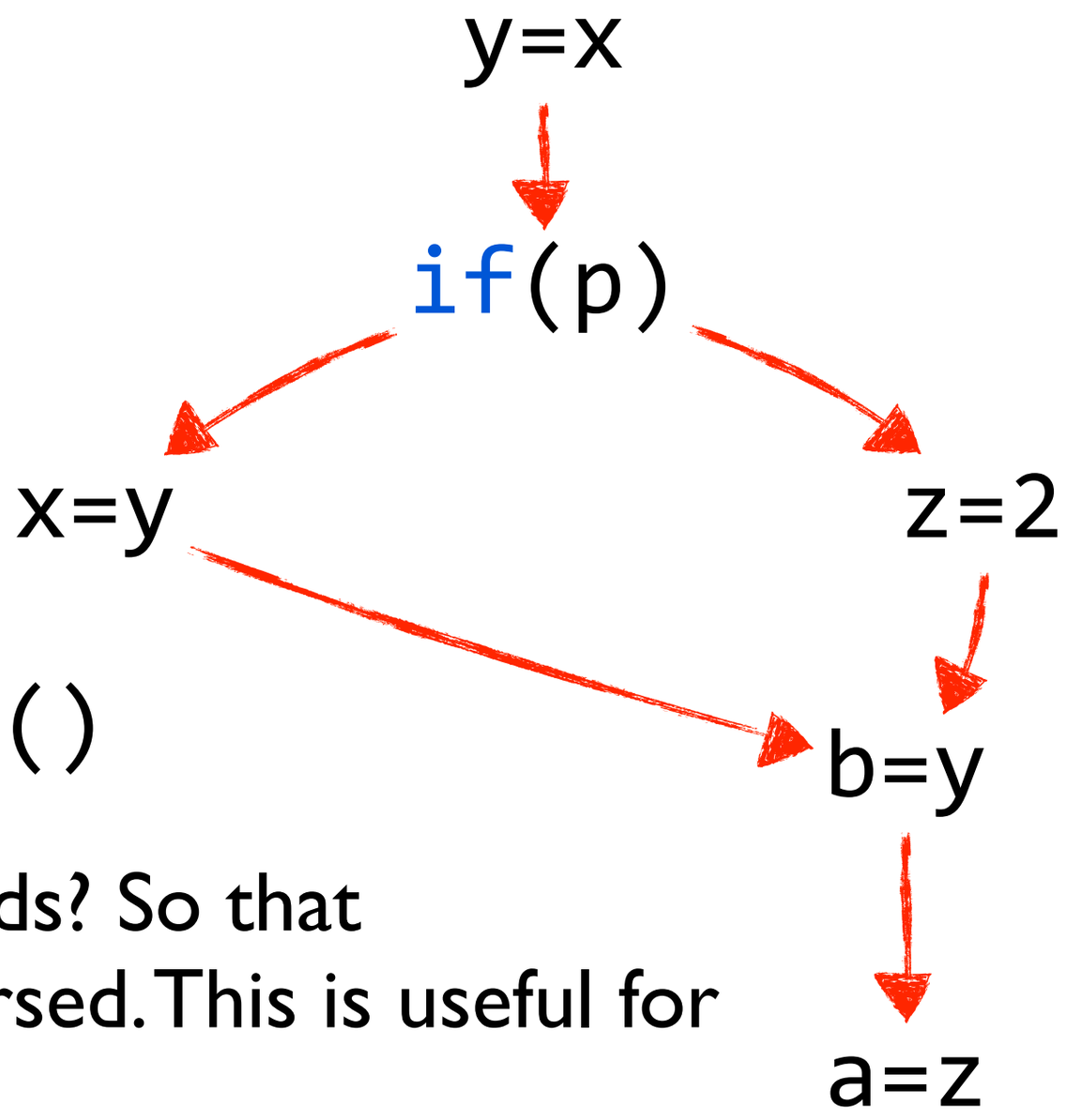


Getting a CFG...



better: `new ExceptionalUnitGraph`
(models exceptional flow as well)

Main operations on UnitGraph



- `getBody()`
- `getHeads()`, `getTails()`
  - Why allowing for multiple heads? So that UnitGraphs can easily be reversed. This is useful for backward analyses.
- `getPredsOf(u)`, `getSuccsOf(u)`

More about Soot and Jimple in Lab "ICA"

Tomorrow, same time, same place

almost

Summary

- Intermediate representations can abstract from concrete input languages
- Jimple is an intermediate language in three-address code format
  - most things are explicit
  - every statement is atomic, no nesting
  - simplifies notation of flow functions

Next week: intra-procedural static analysis