Debug All Your Code: Portable Mixed-Environment Debugging

Byeongcheol Lee
Martin Hirzel
Robert Grimm
Kathryn McKinley

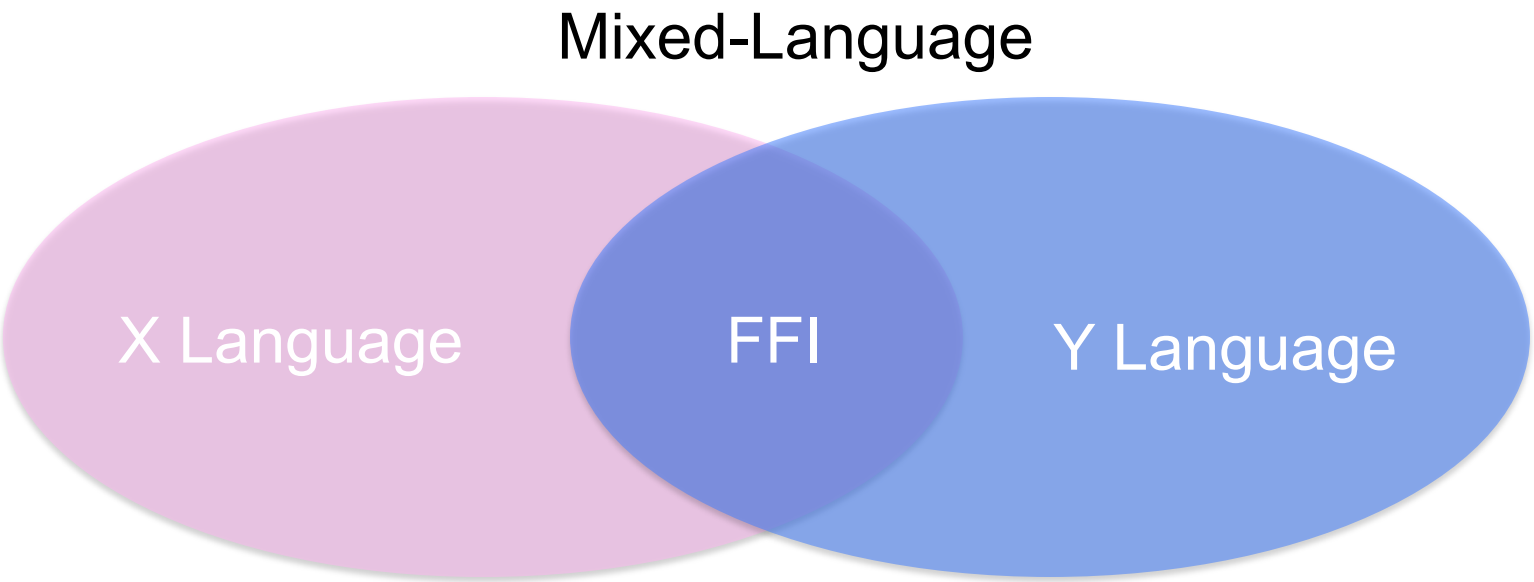
OOPSLA 2009

THE UNIVERSITY OF TEXAS AT AUSTIN
IBM
NEW YORK UNIVERSITY

Portable mixed-environment debugging

Programmers build systems in multiple languages.

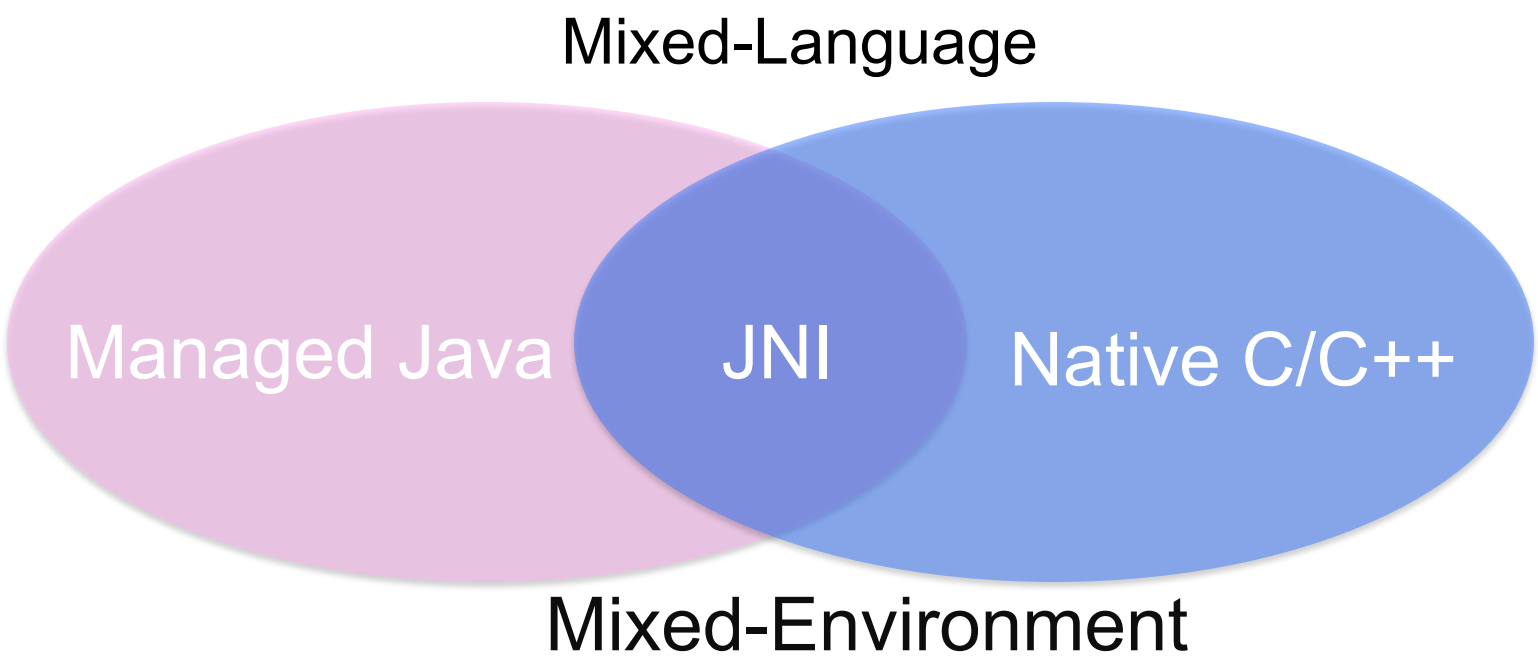
1. Leverage legacy code and existing libraries.
2. Match language features to a task.



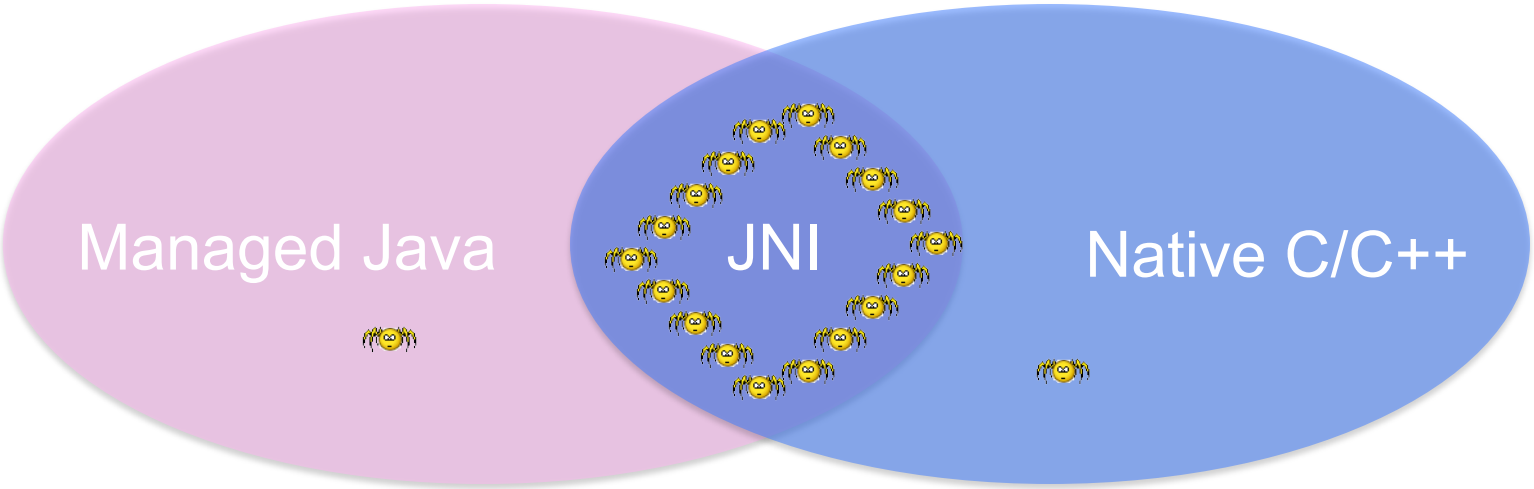
Portable mixed-environment debugging

Programmers build systems in multiple languages.

1. Leverage legacy code and existing libraries.
2. Match language features to a task.

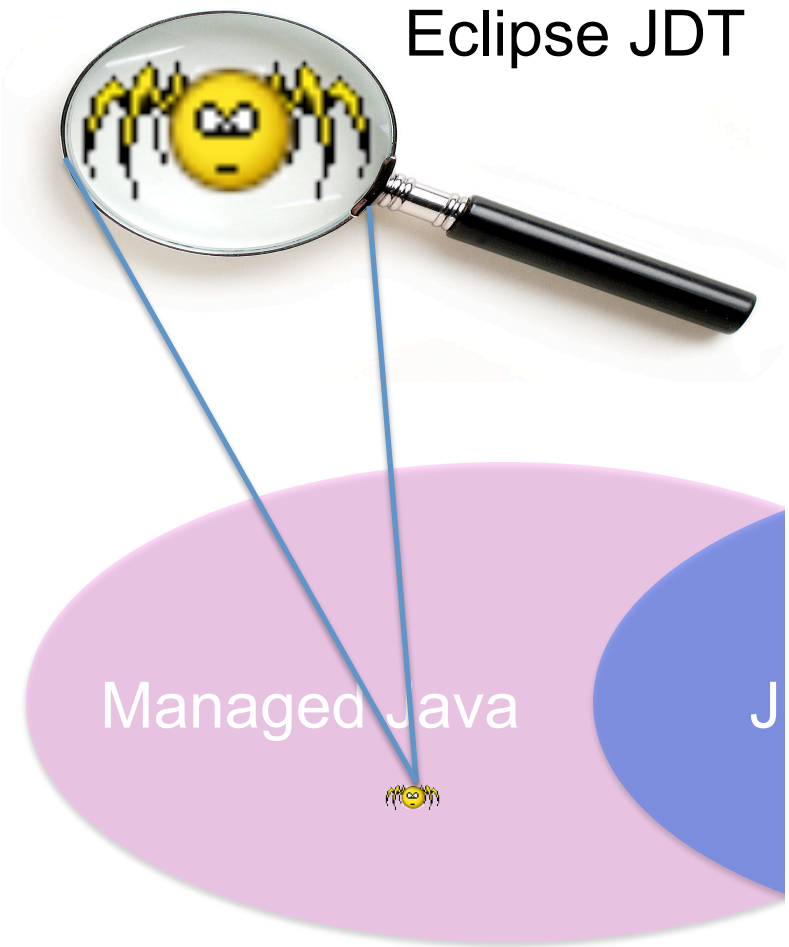


Bugs appear in all your code!



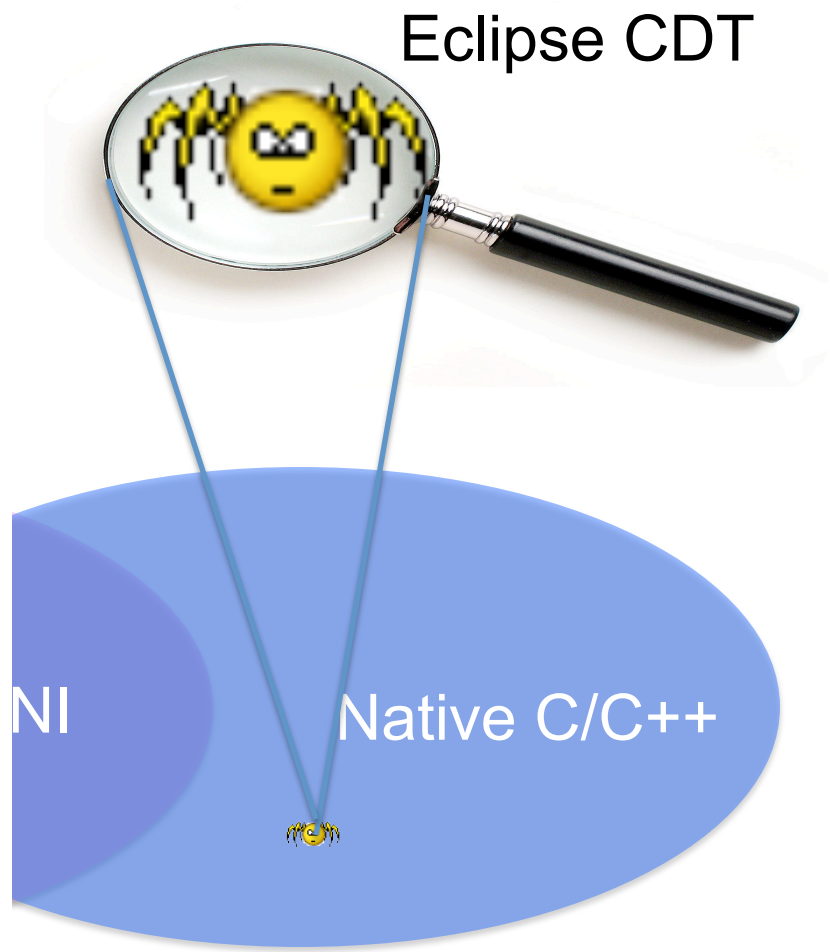
Mixed-Environment

The problem with single-environment debugging



Mixed-Environment

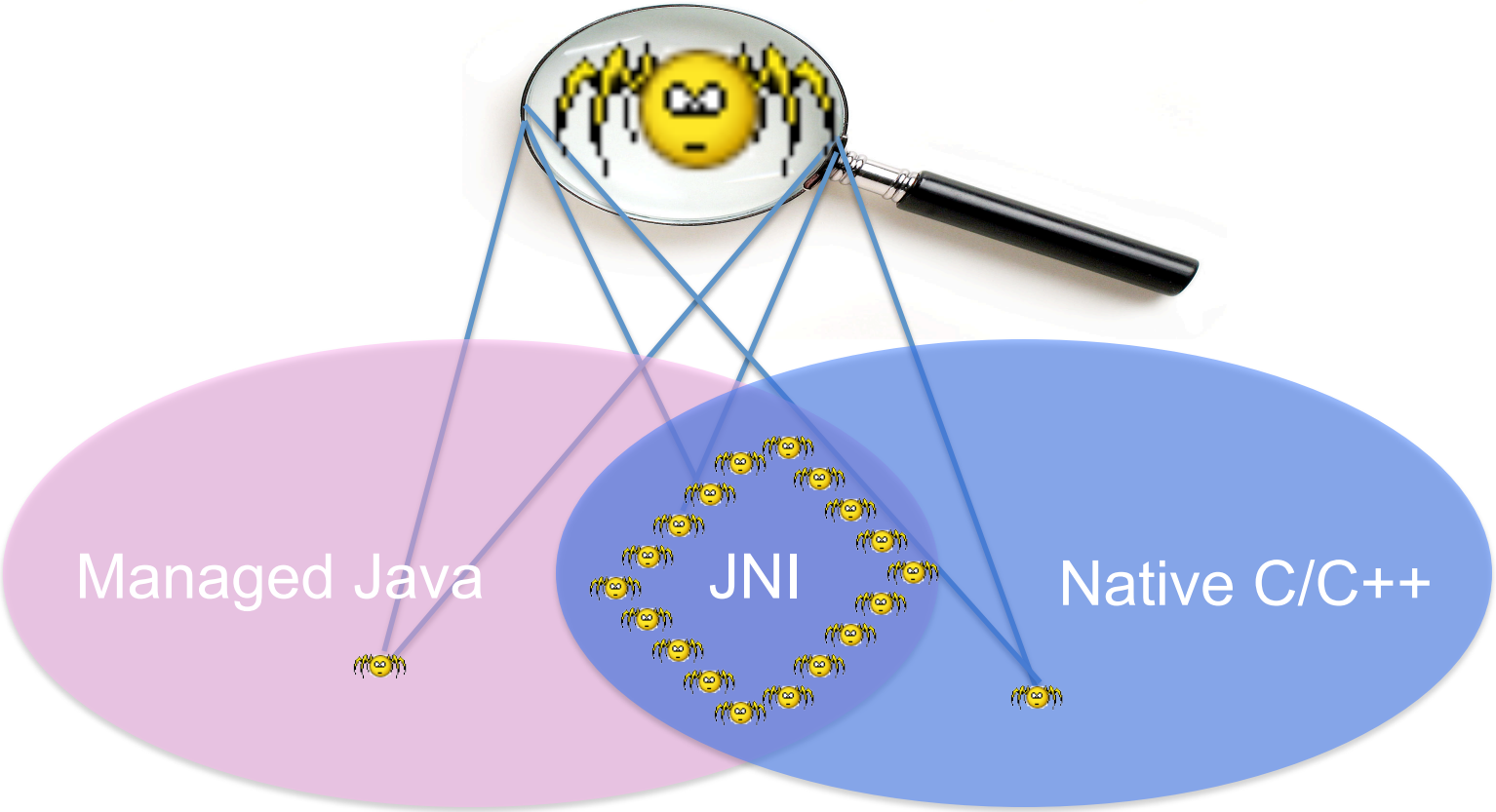
The problem with single-environment debugging



Mixed-Environment

Our goal

Portable mixed-environment debugging



Mixed-Environment

Our contribution and results

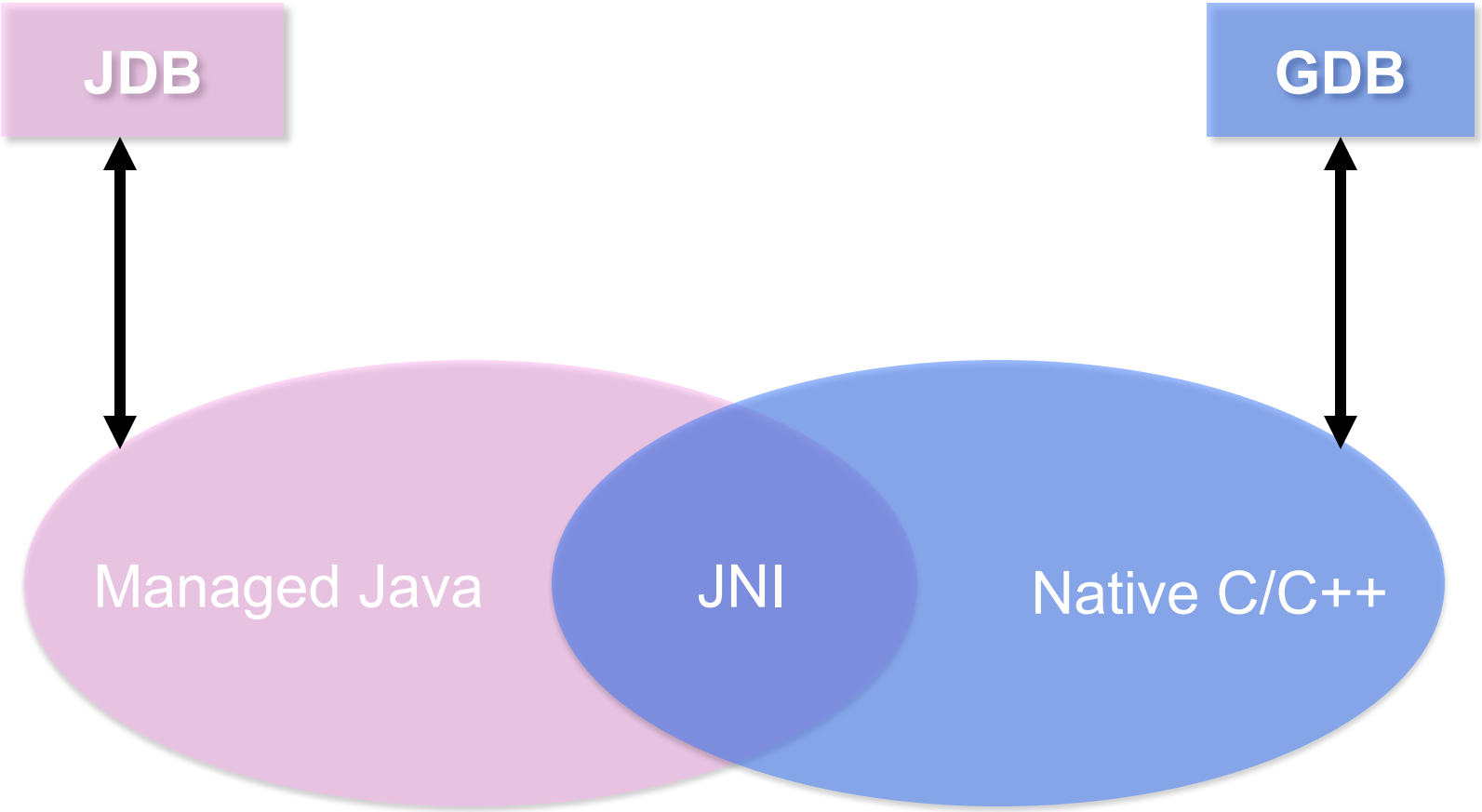
Composition

1. Add an intermediate agent.
2. Attach single-environment debuggers.
3. Dispatch debuggers dynamically.

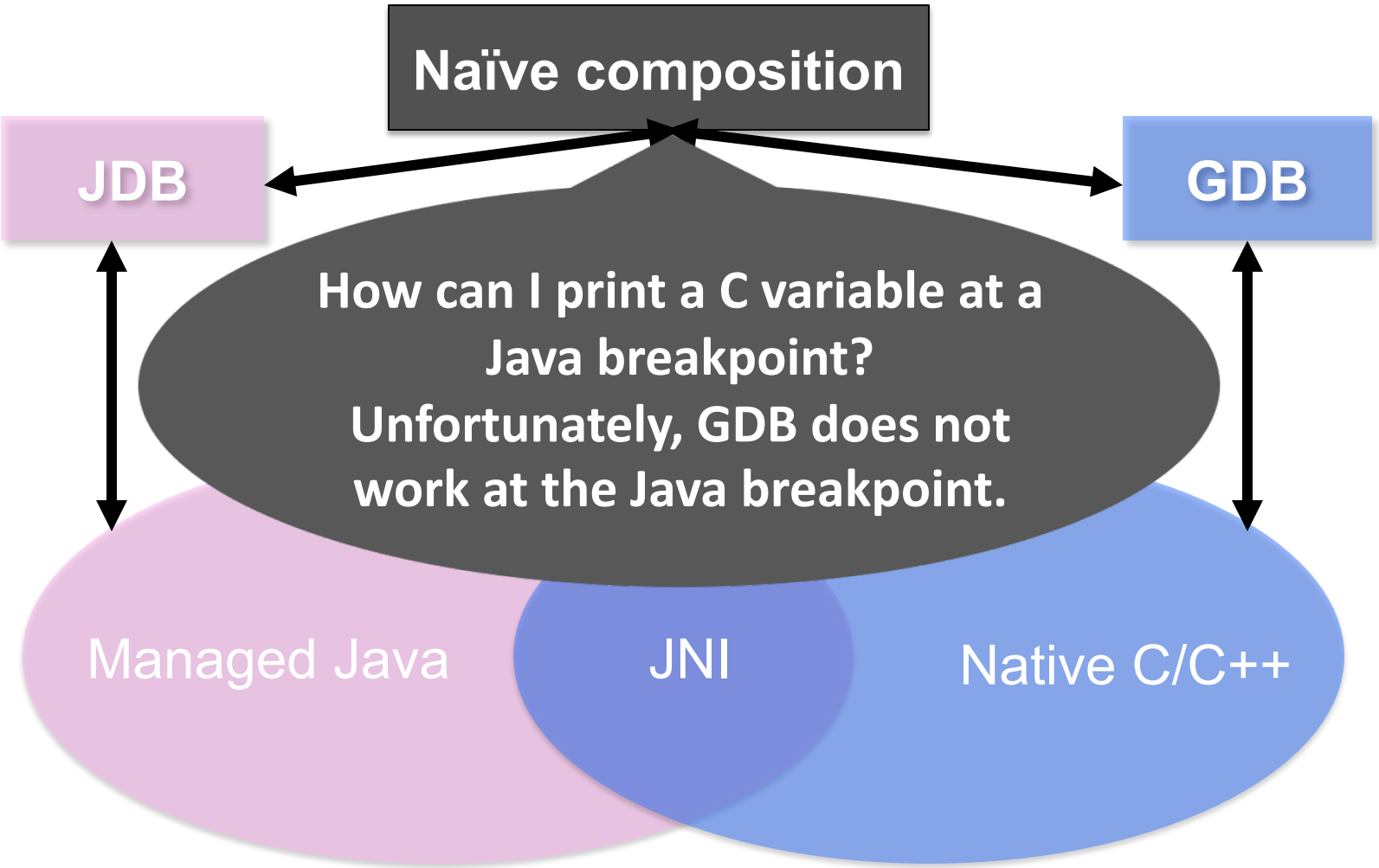
Blink results

1. **Simple**: Add 10 K SLOC of new code.
2. **Portable**: Support Linux, Windows, Hotspot, J9, GCC, Microsoft C++.
3. **Powerful**: Catch FFI bugs.

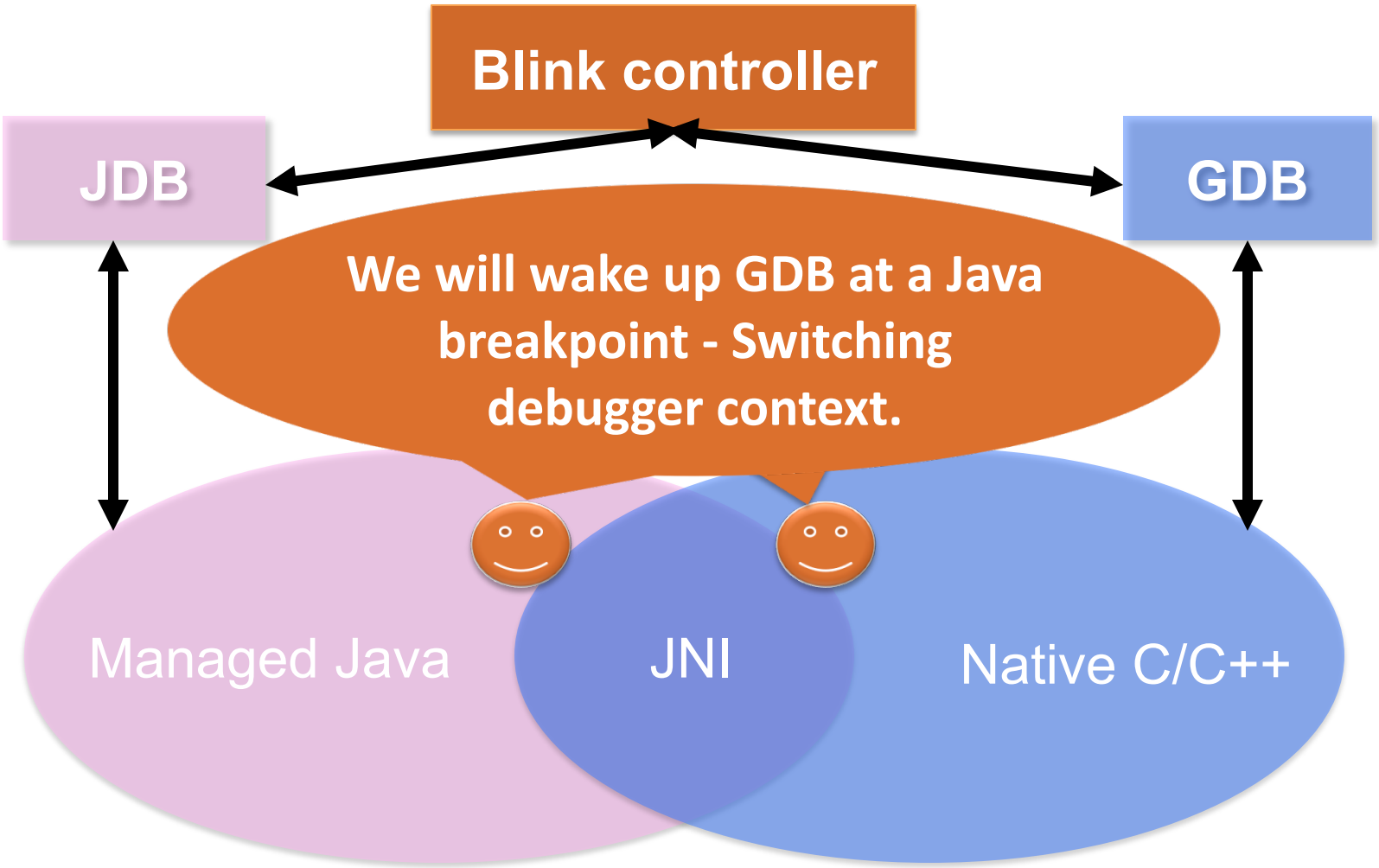
Problem: GDB does not work at a Java breakpoint.



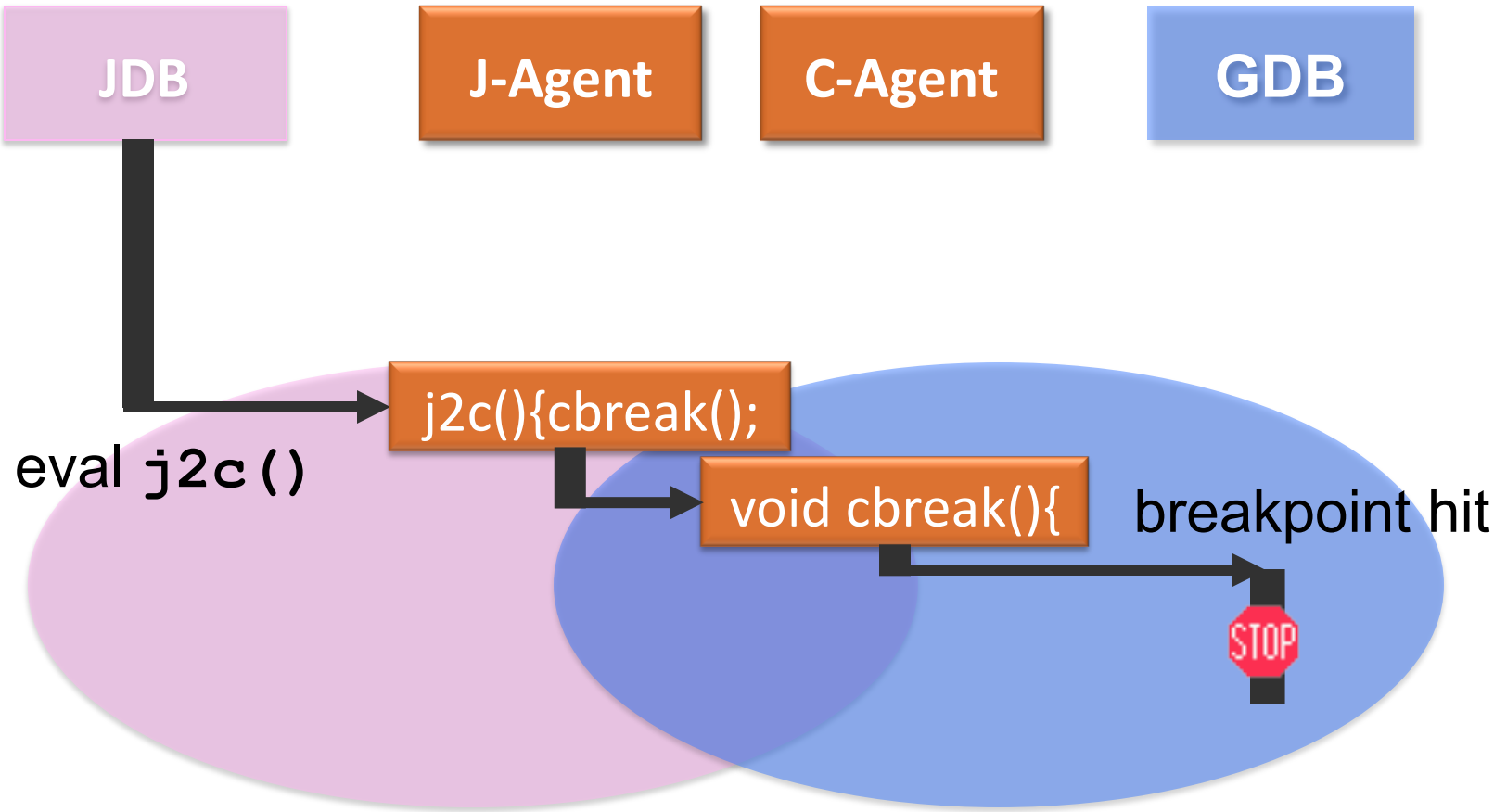
Problem: GDB does not work at a Java breakpoint.



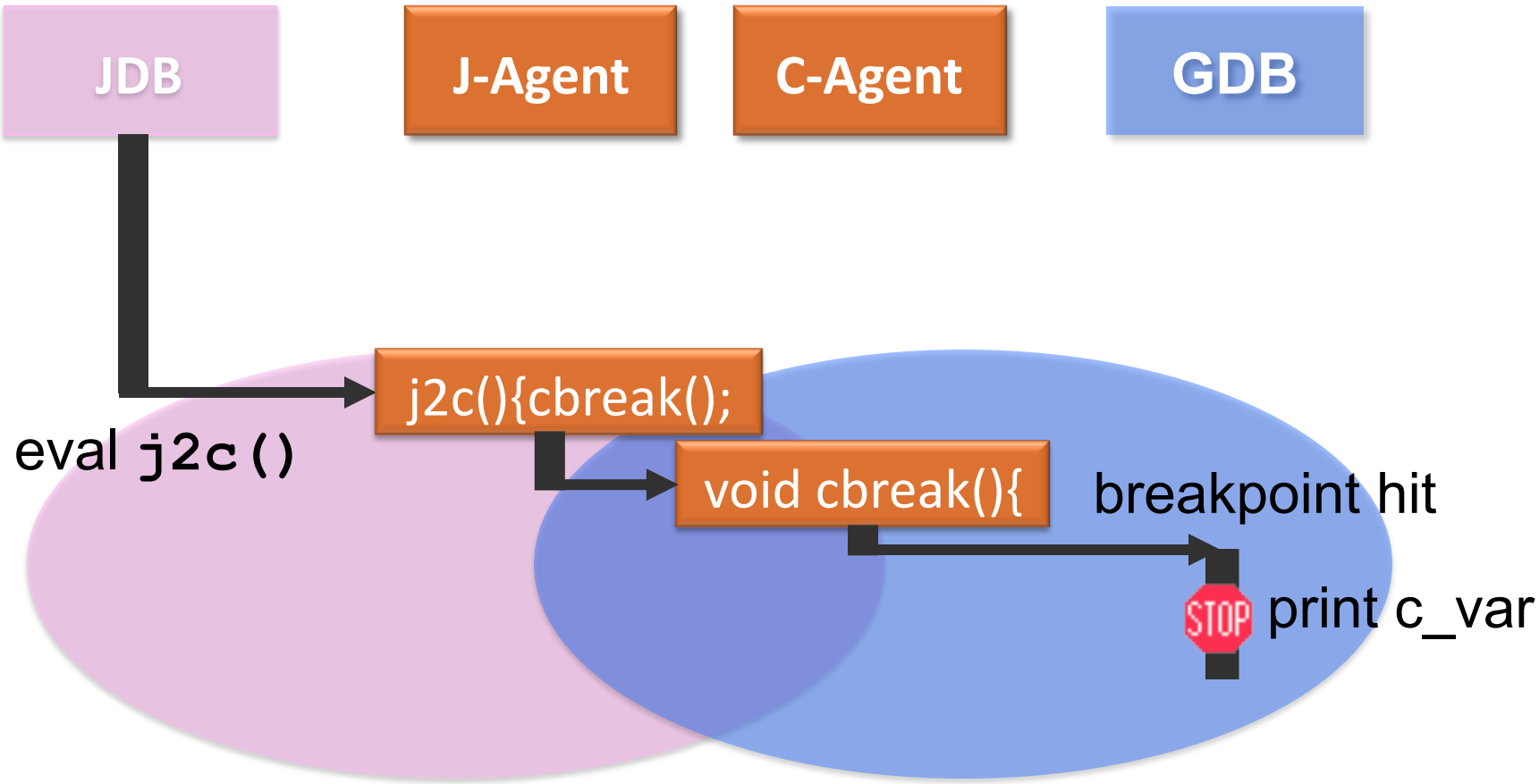
Our solution: the intermediate agent switches debugger context.



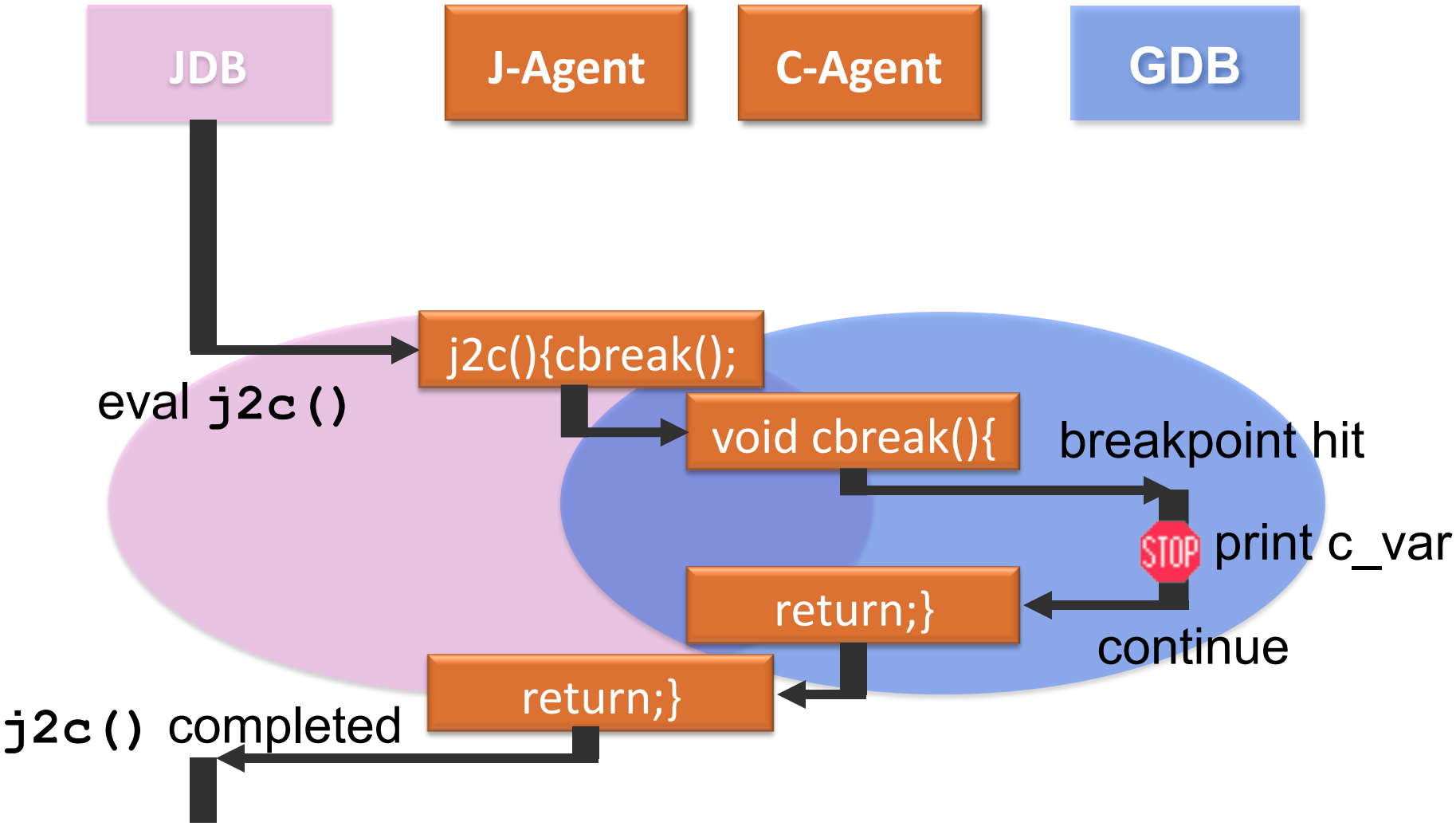
Our solution: switch debugger context from Java to C.



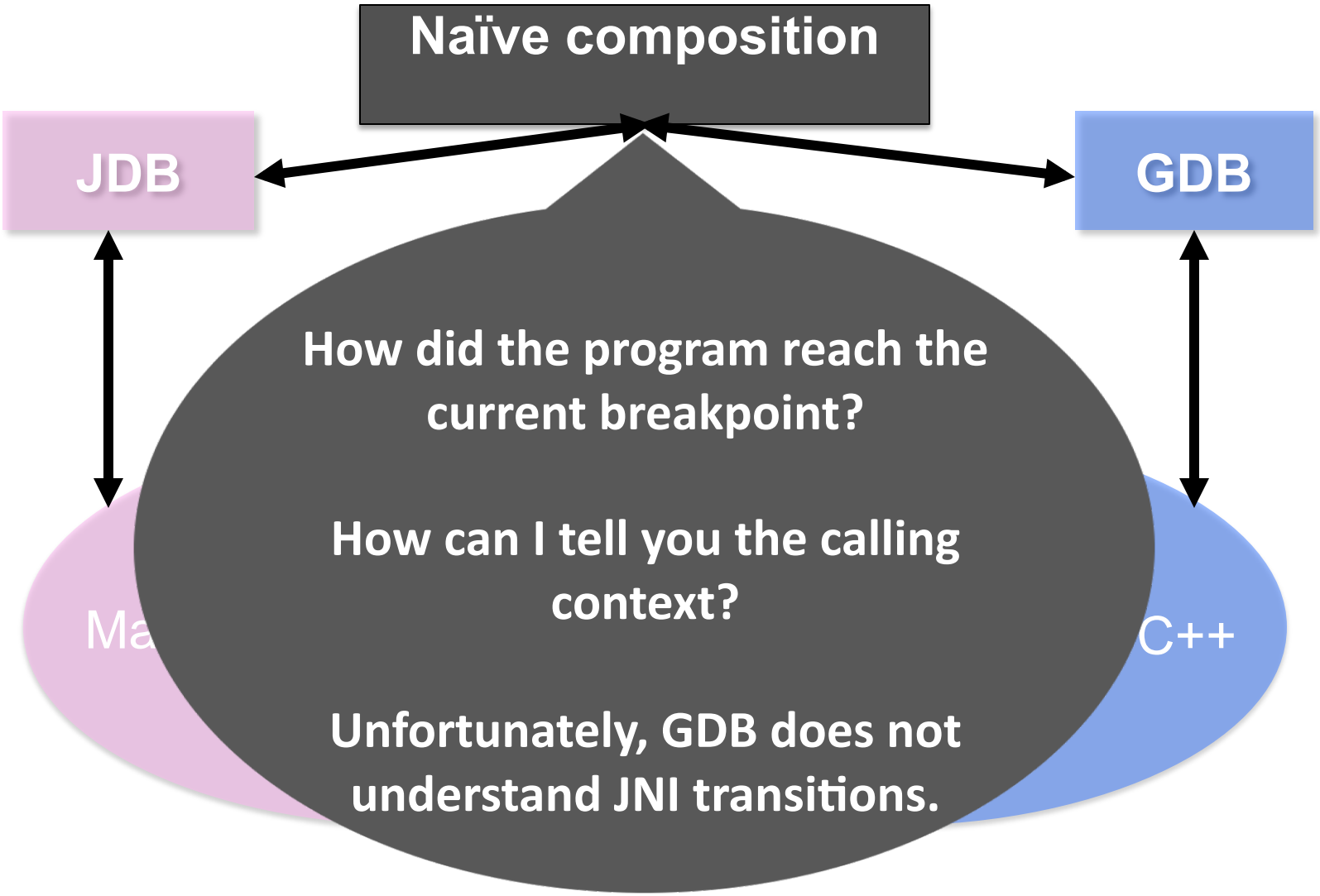
Our solution: switch debugger context from Java to C.



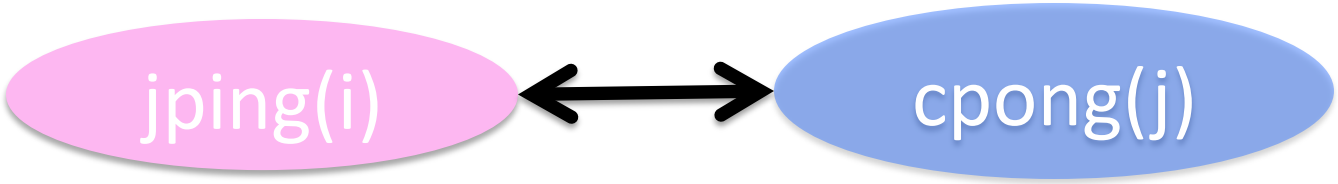
Our solution: switch debugger context from Java to C.



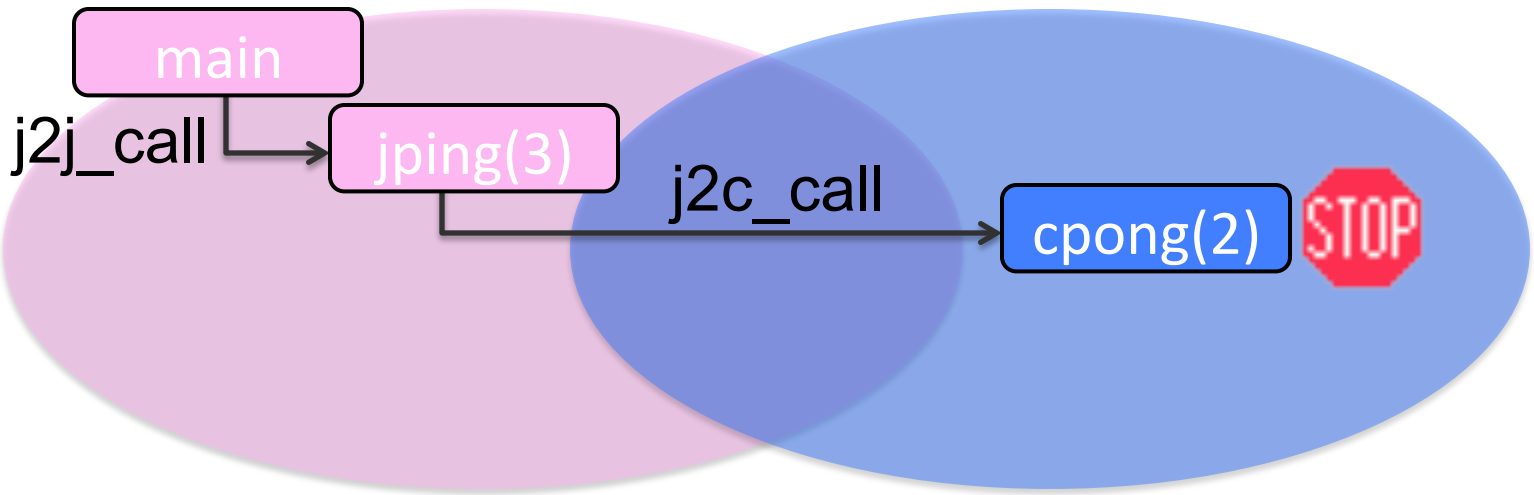
Problem: GDB does not understand JNI calling conventions.



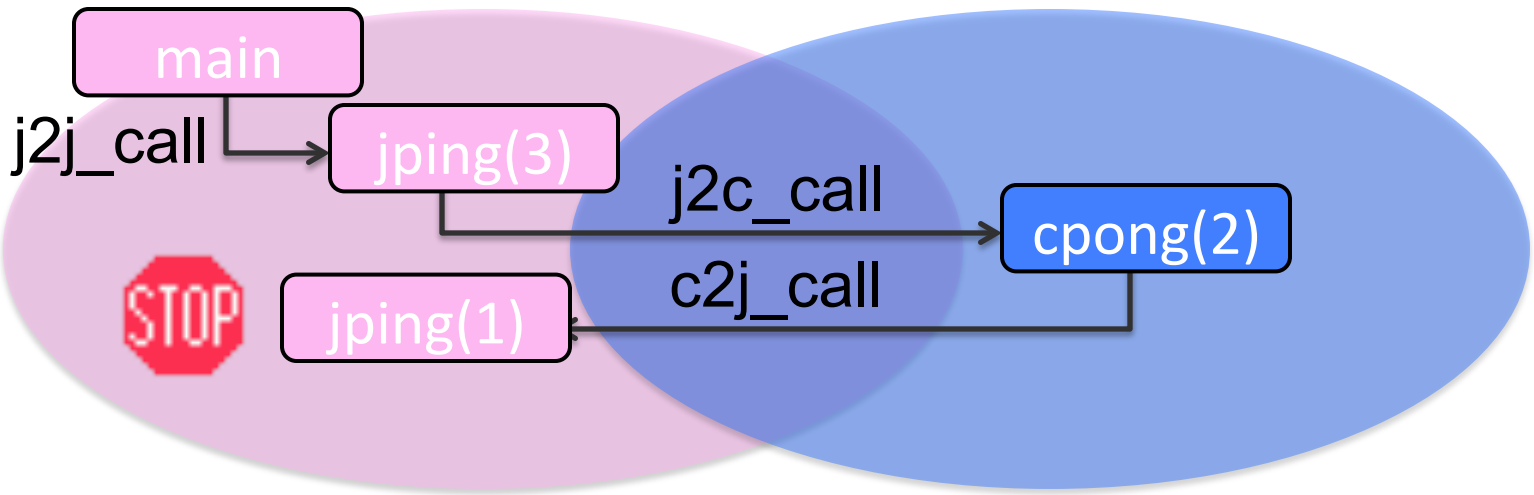
Problem: GDB does not understand JNI calling conventions.



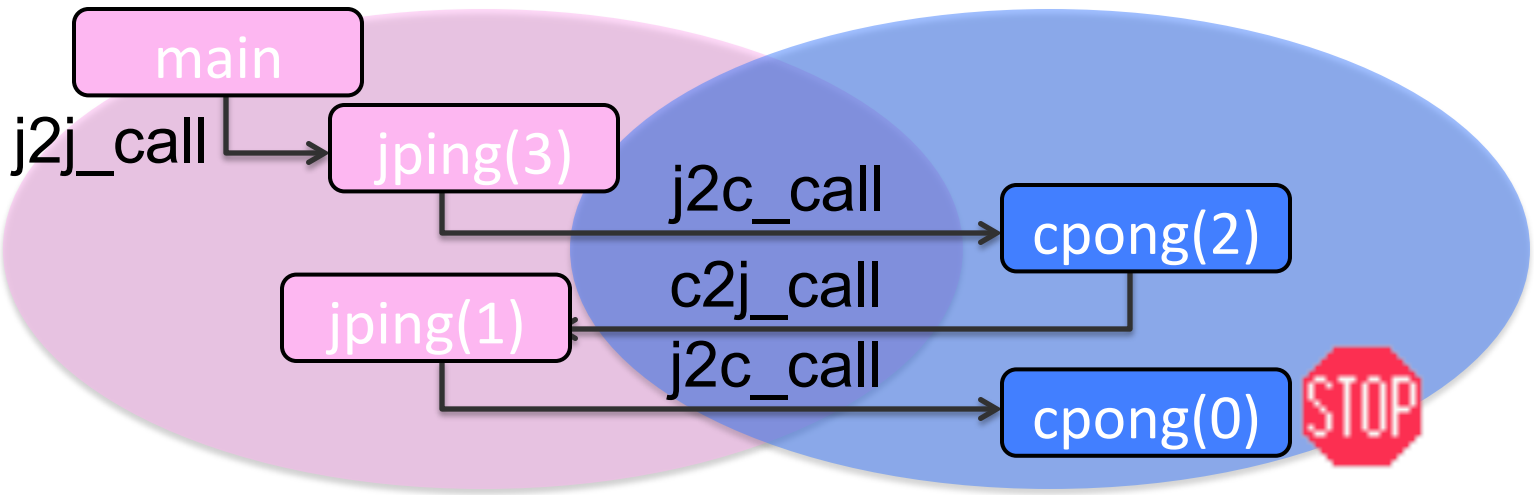
Problem: GDB does not understand JNI calling conventions.



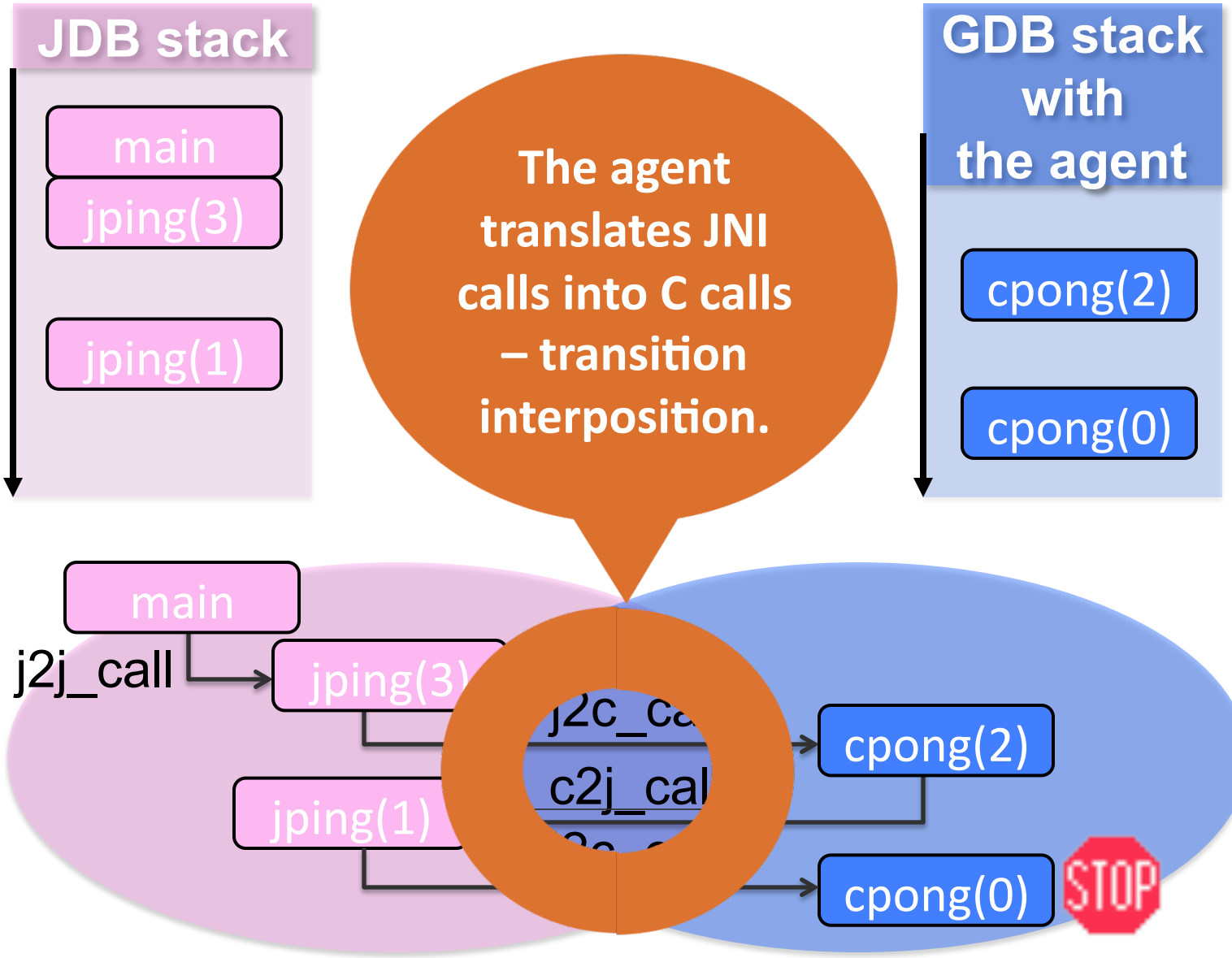
Problem: GDB does not understand JNI calling conventions.



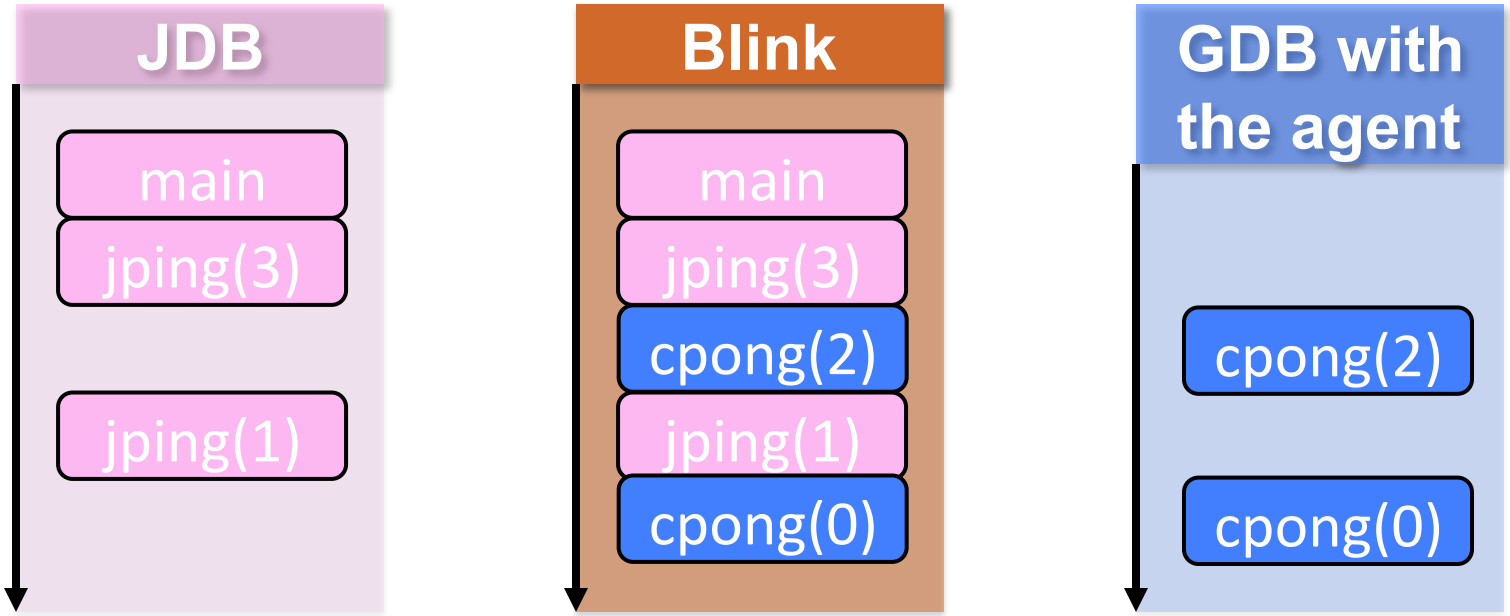
Problem: GDB does not understand JNI calling conventions.

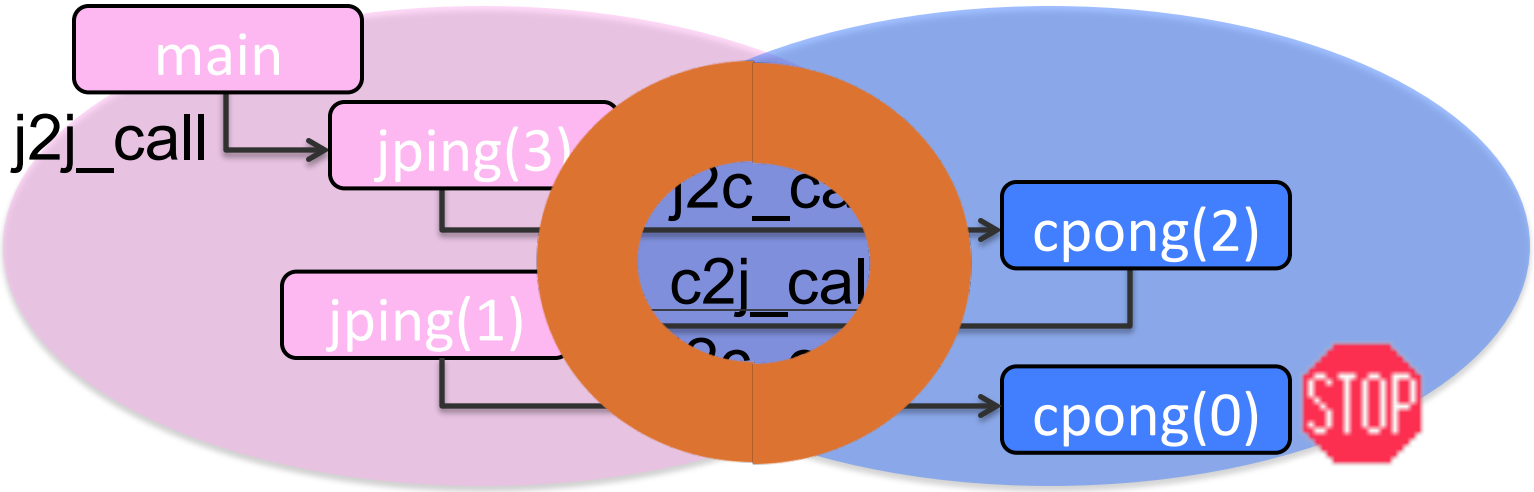


Our solution: translate JNI transitions into C transitions.

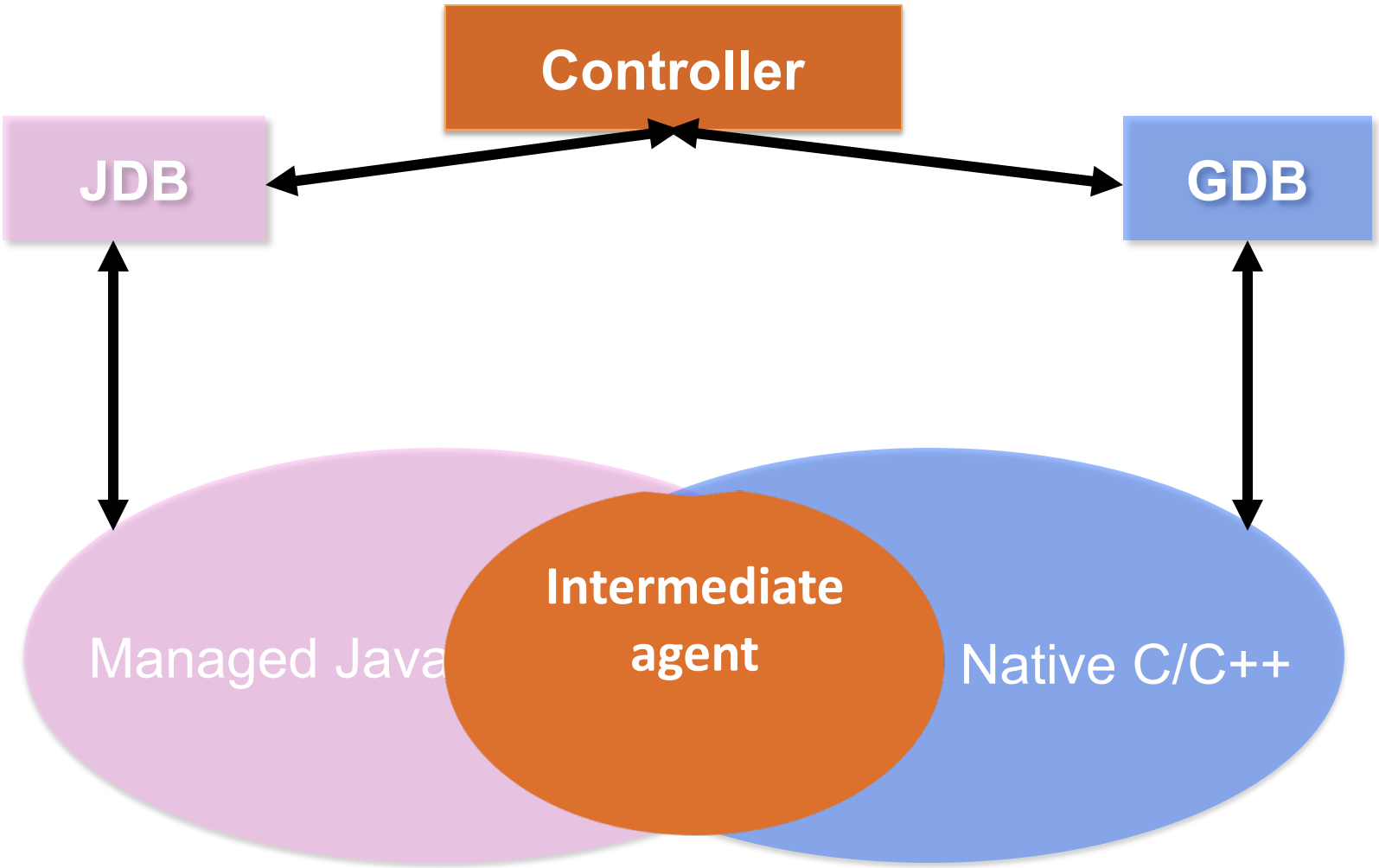


Our solution: compose a calling context.





Our solution: controller and intermediate agent

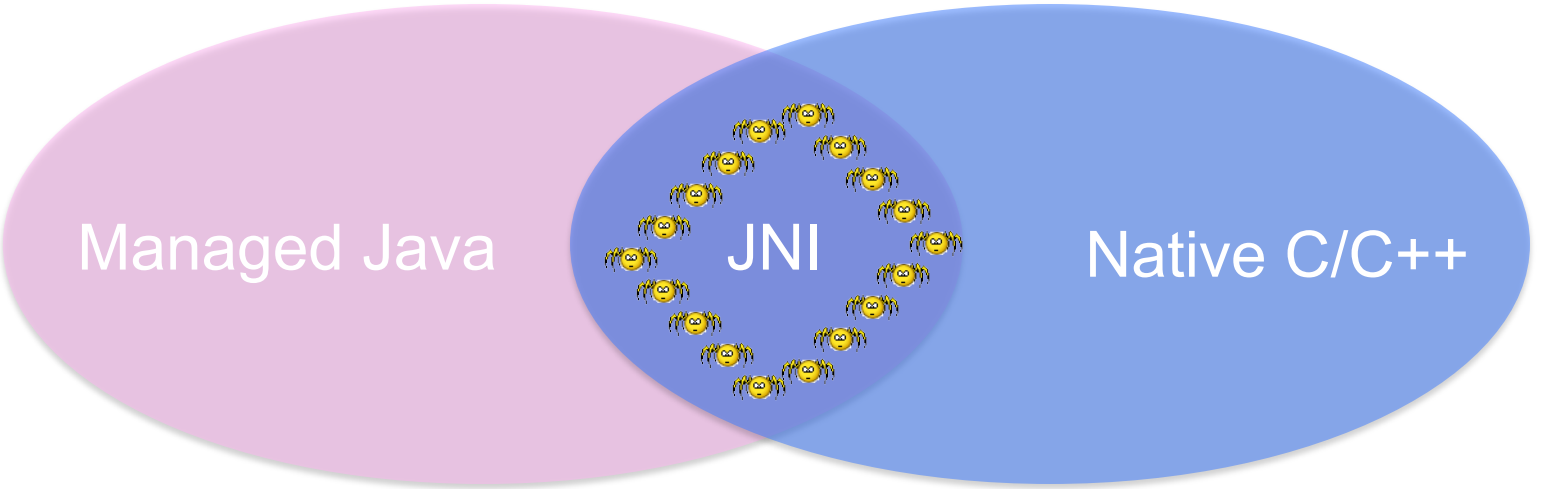


Outline

I. Problem
II. Debugger composition
   A. Switching debugger context
   B. Interposing transitions
III. Advanced features
   A. Evaluating Jeannie mixed-environment expressions
   B. Detecting FFI bugs
IV. Evaluation

Debugging boundary code

What do I need to debug boundary code?

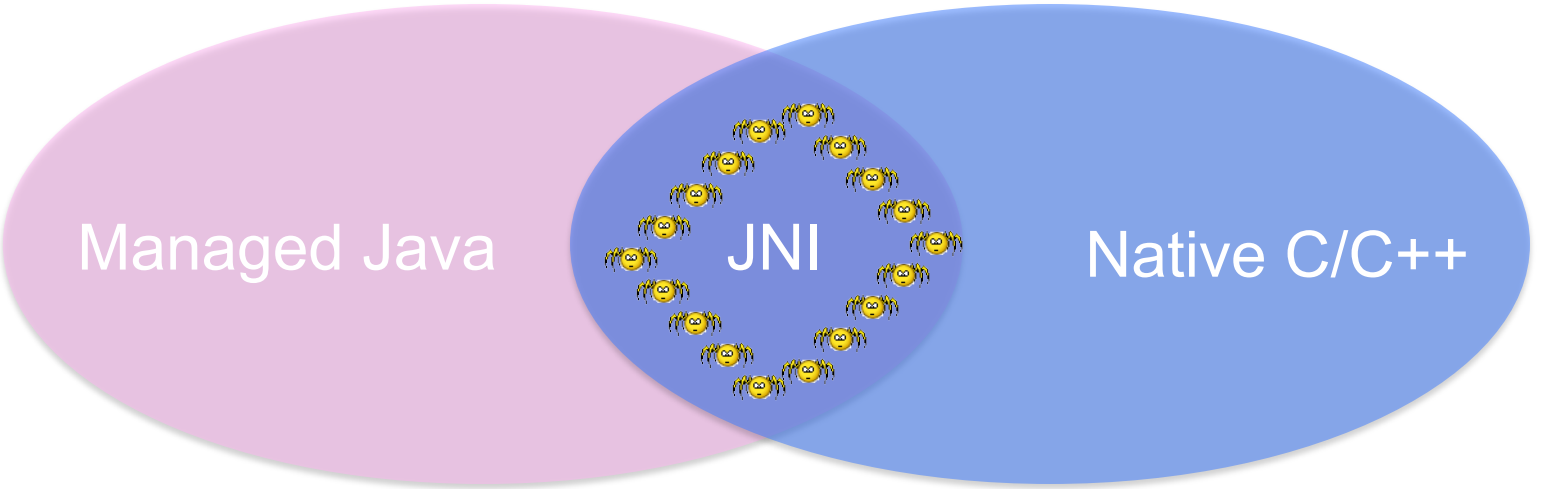


Mixed-Environment

Advanced features to debug boundary code

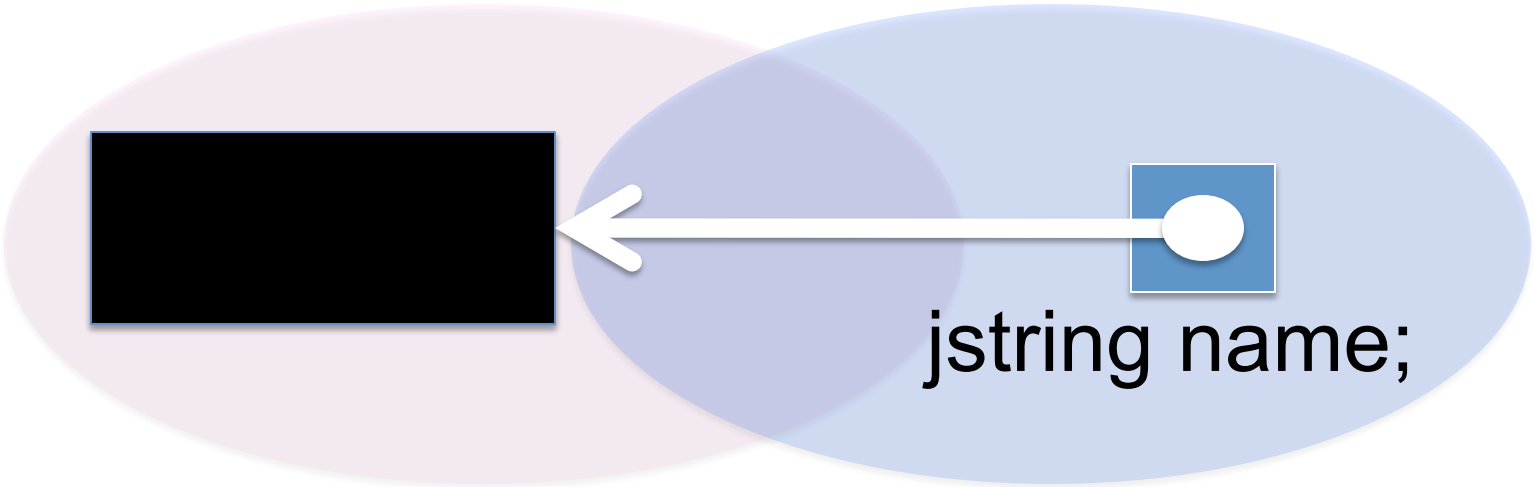
1. Evaluating Jeannie Expressions

2. Detecting FFI bugs

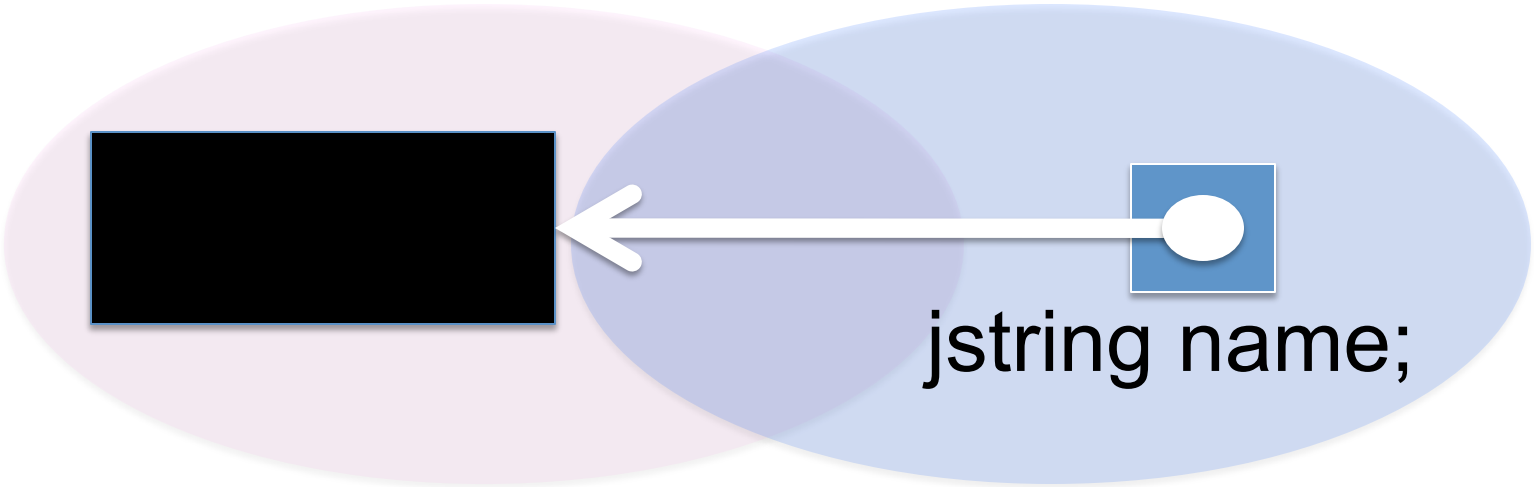


Mixed-Environment

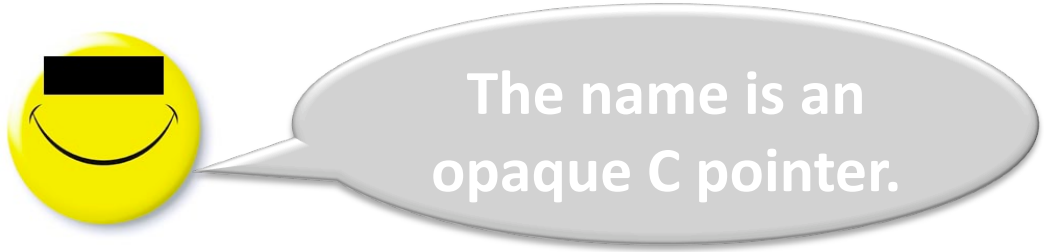
Problem: you can not de-reference opaque pointers in C.



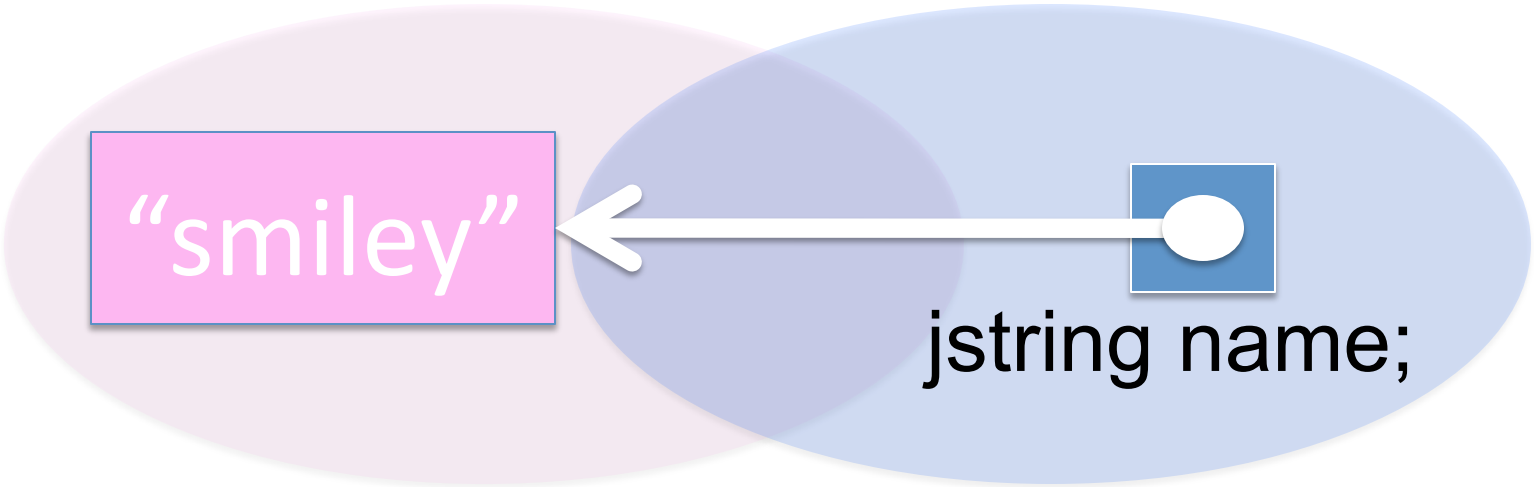
Problem: you can not de-reference opaque pointers in C.

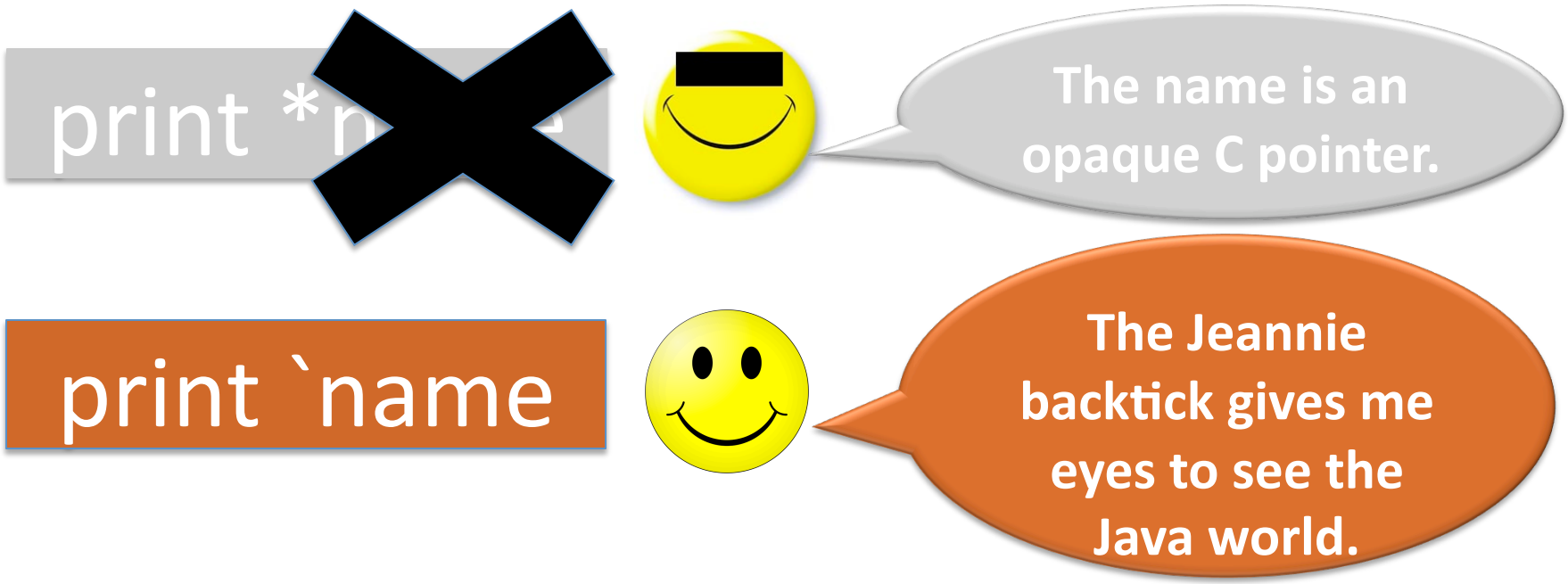


print *name

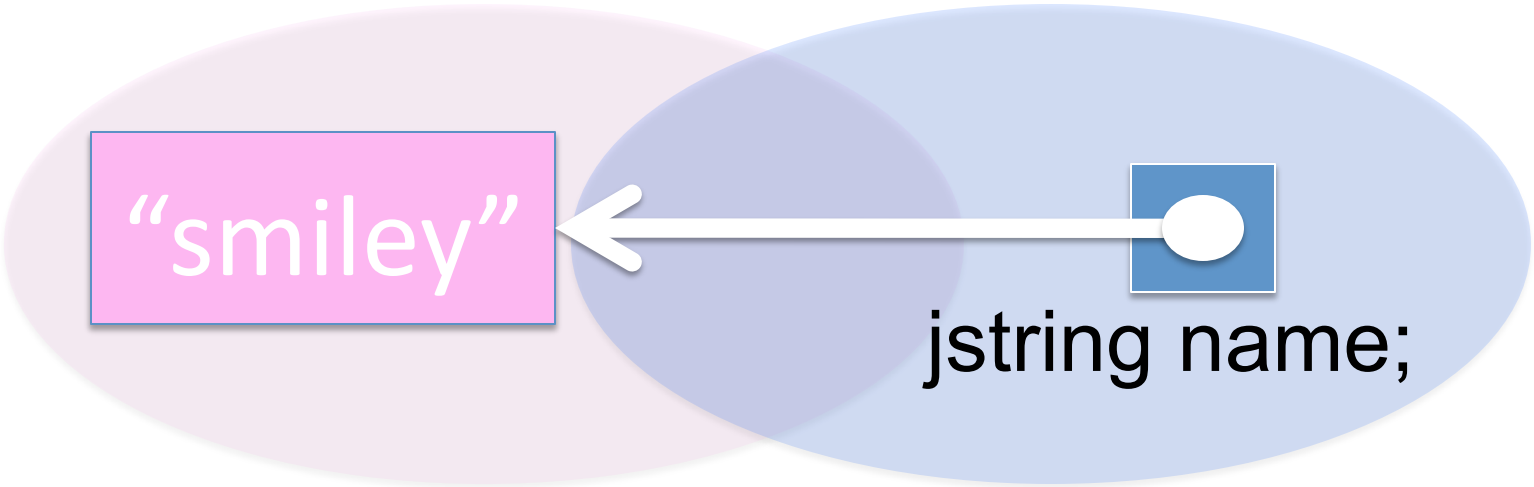


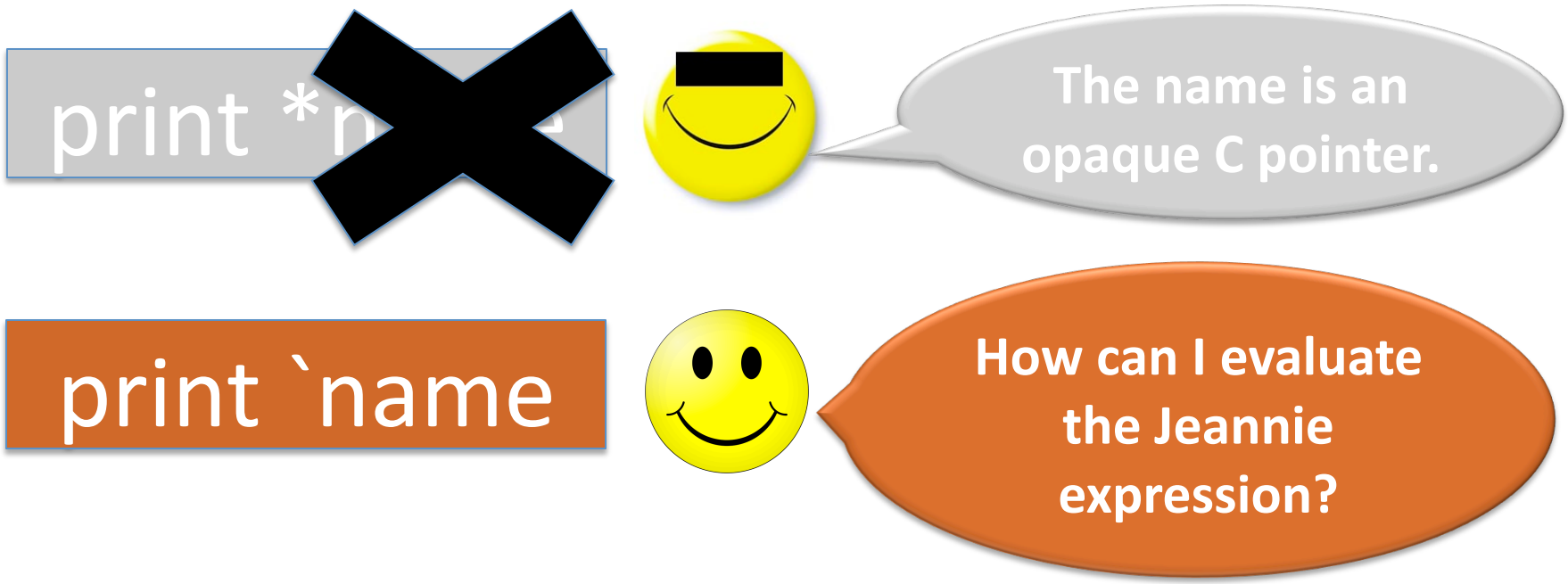
Our solution: use Jeannie expression



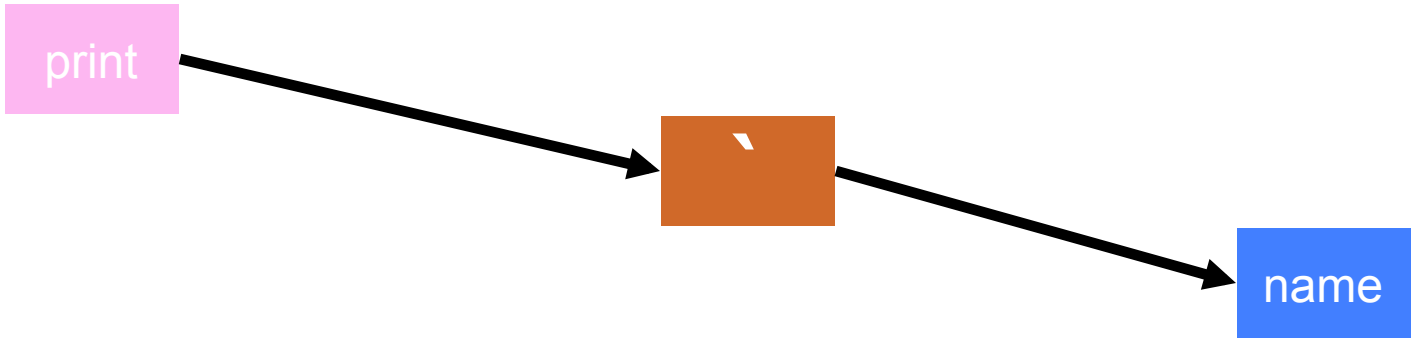


Our solution: use Jeannie expression.

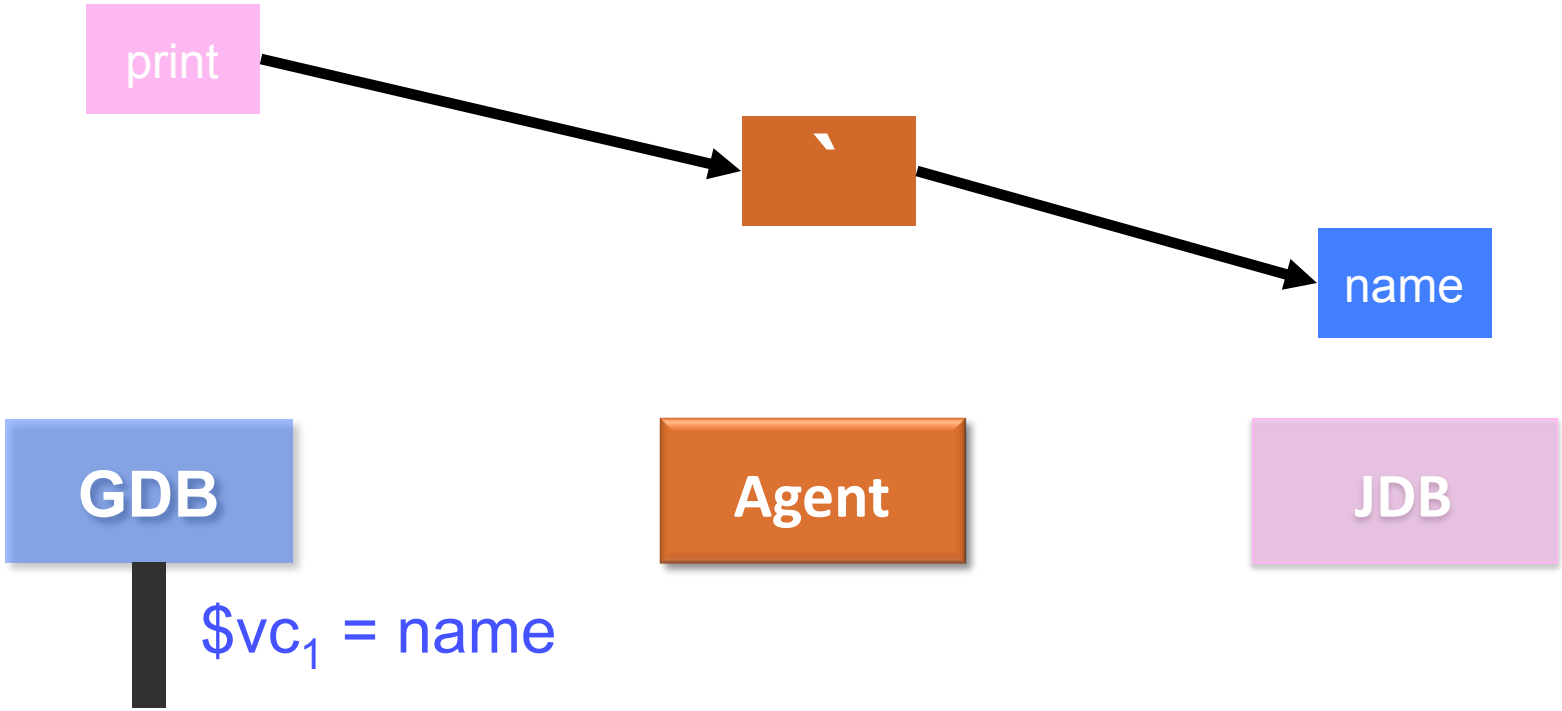




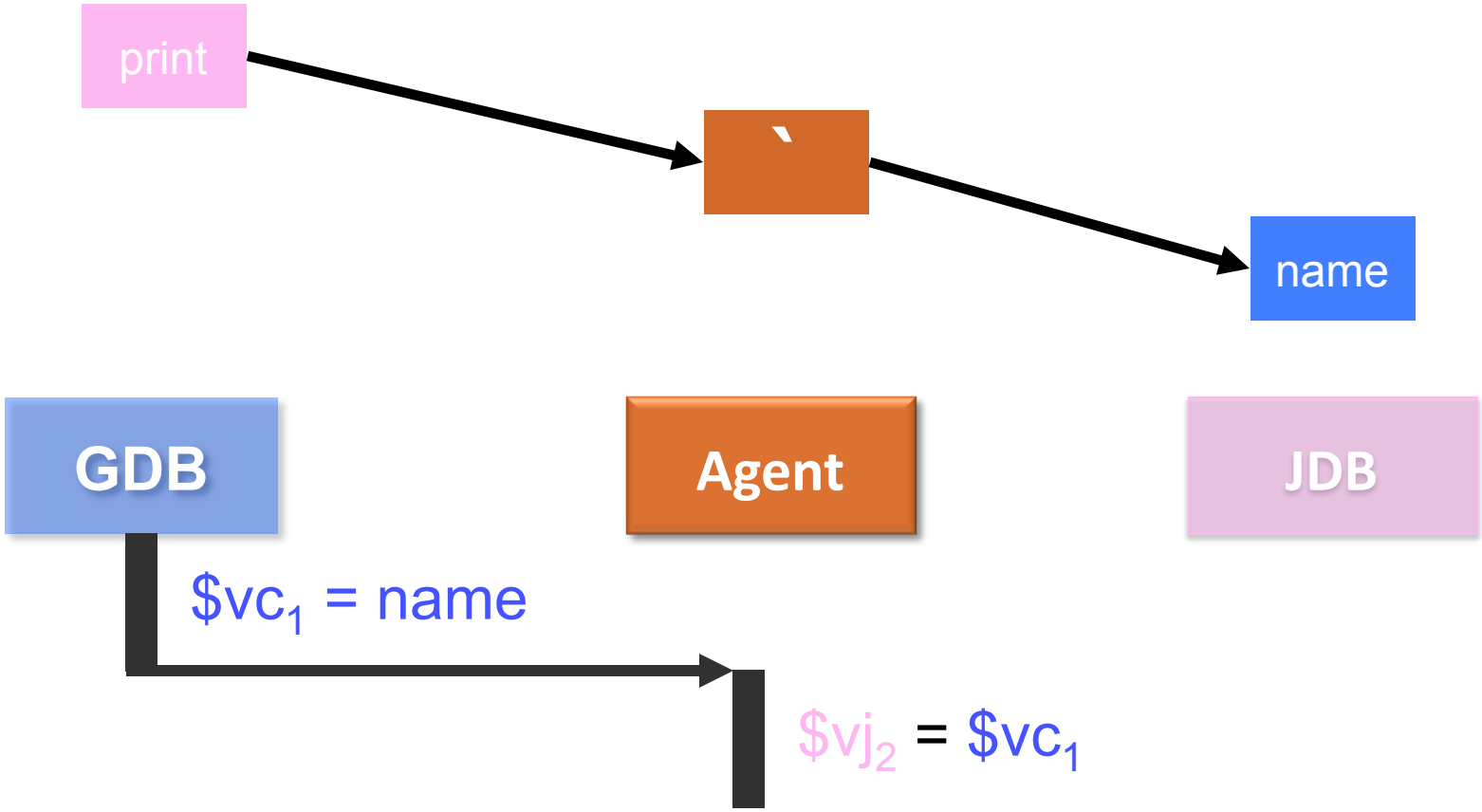
Build abstract syntax tree



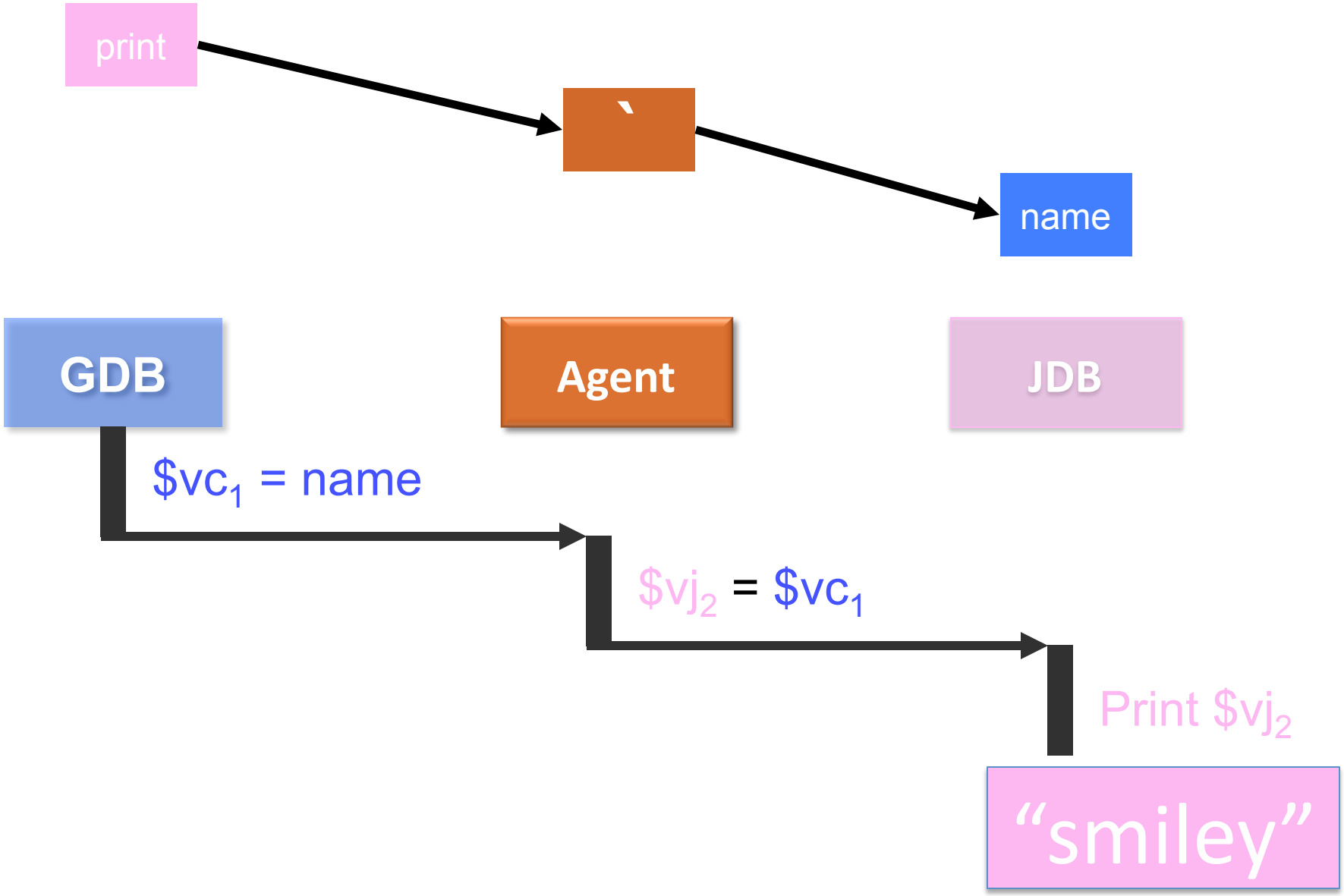
Evaluate AST in Bottom-up order.



Evaluate AST in Bottom-up order.

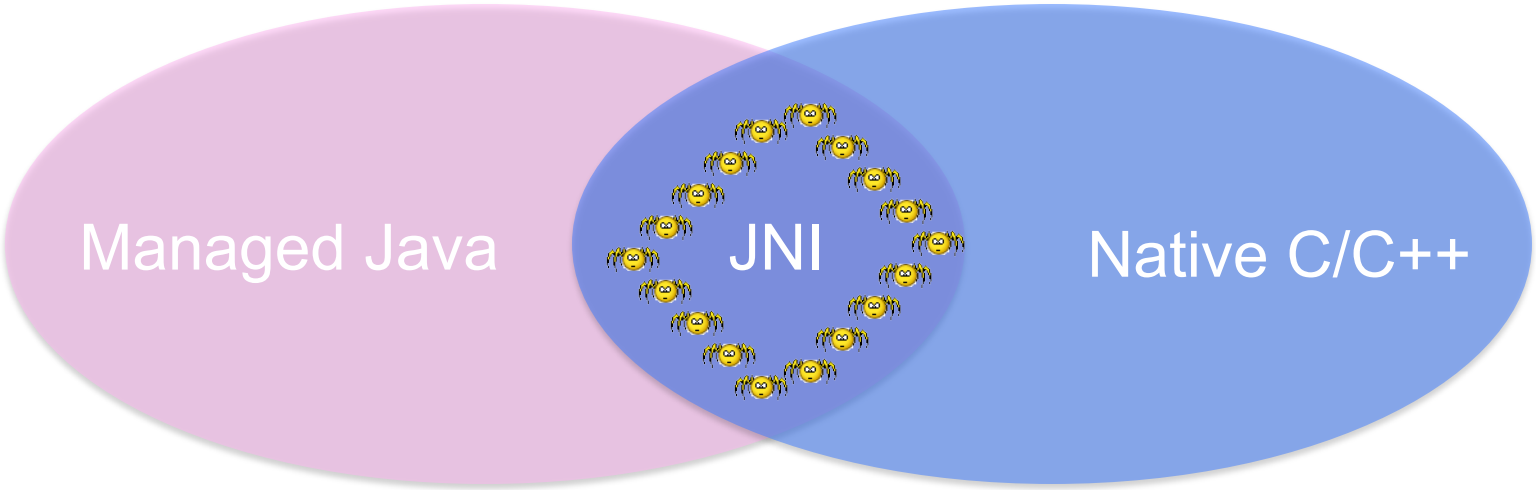


Evaluate AST in Bottom-up order.

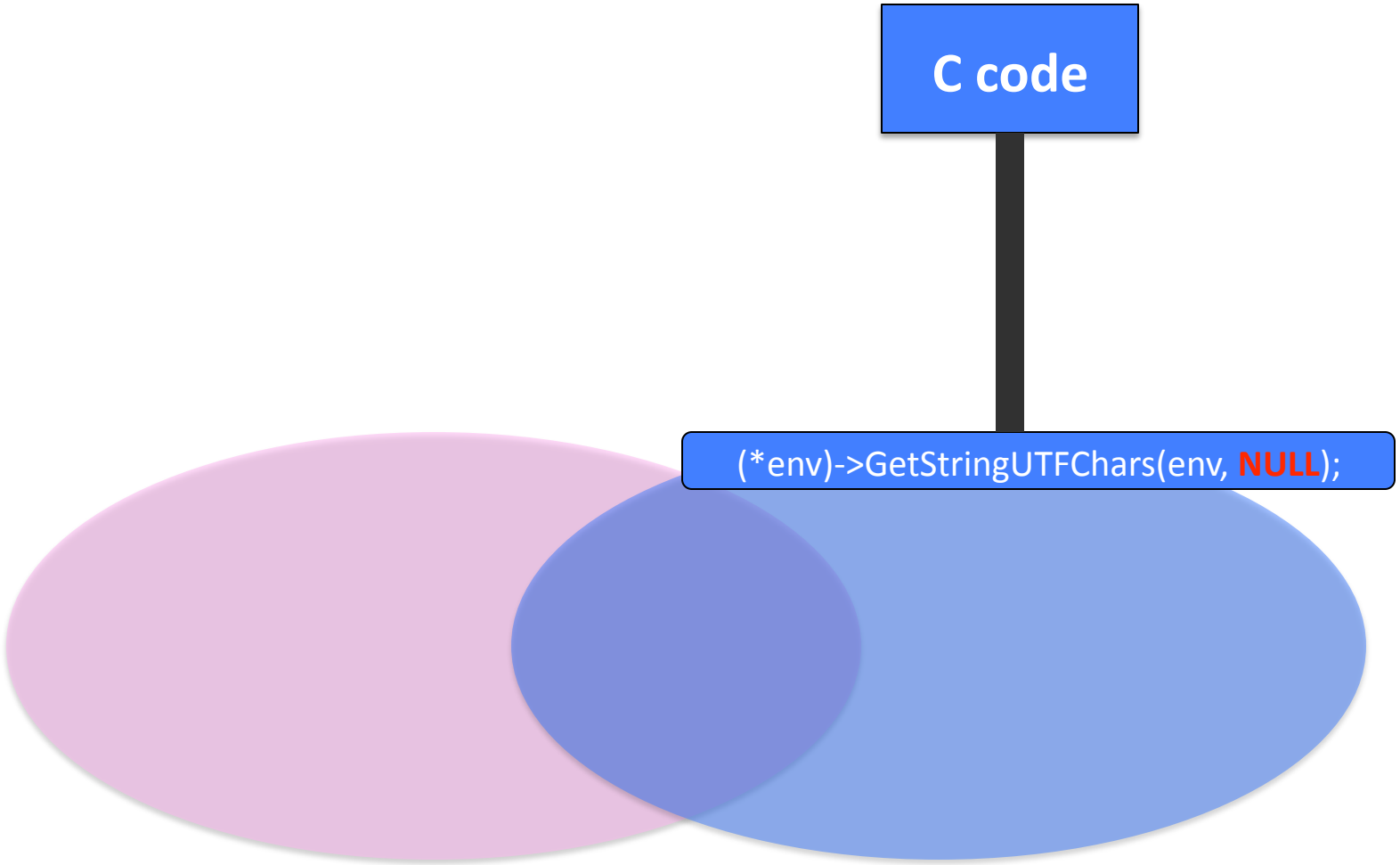


A. Evaluating Jeannie Expressions

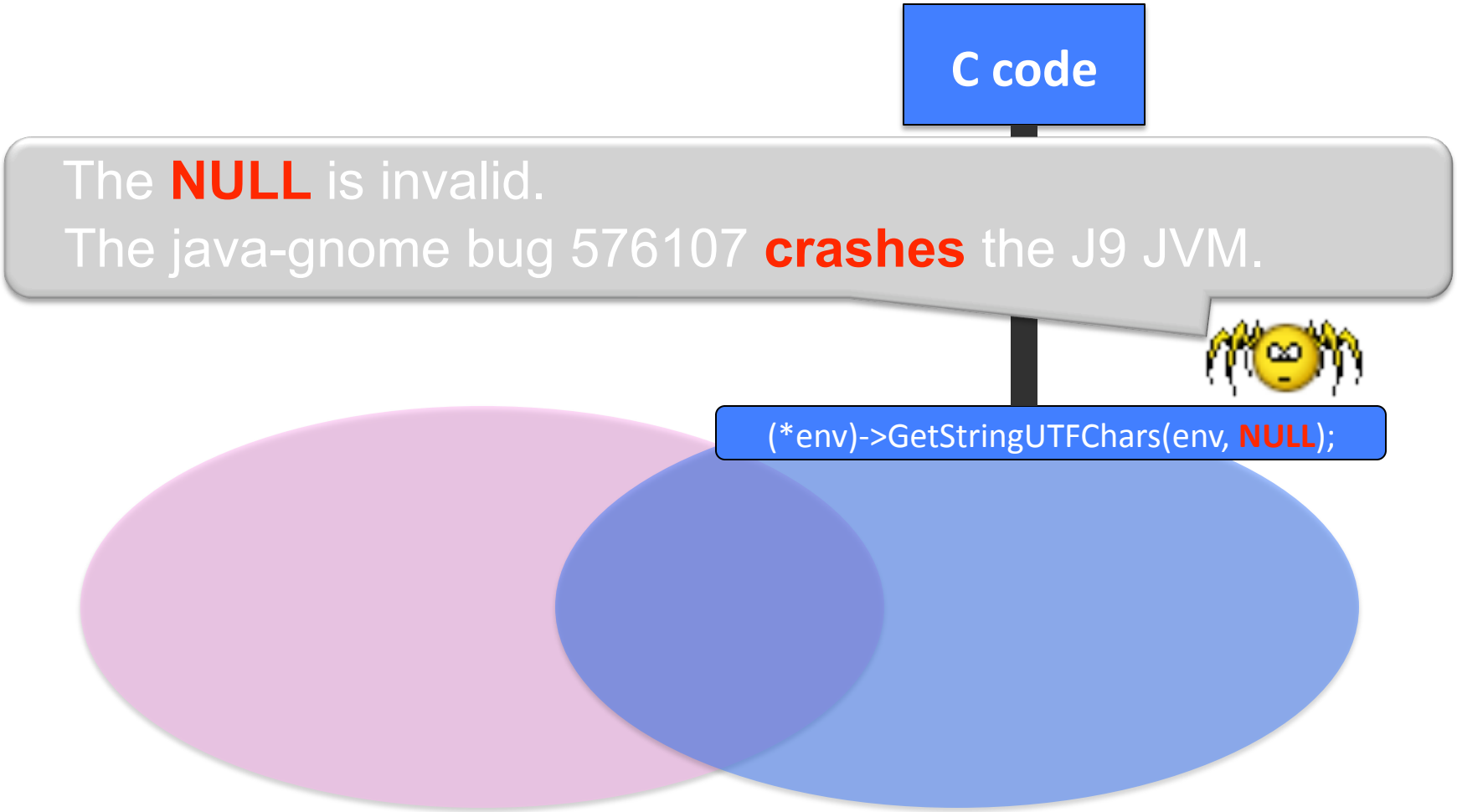
B. Detecting FFI bugs



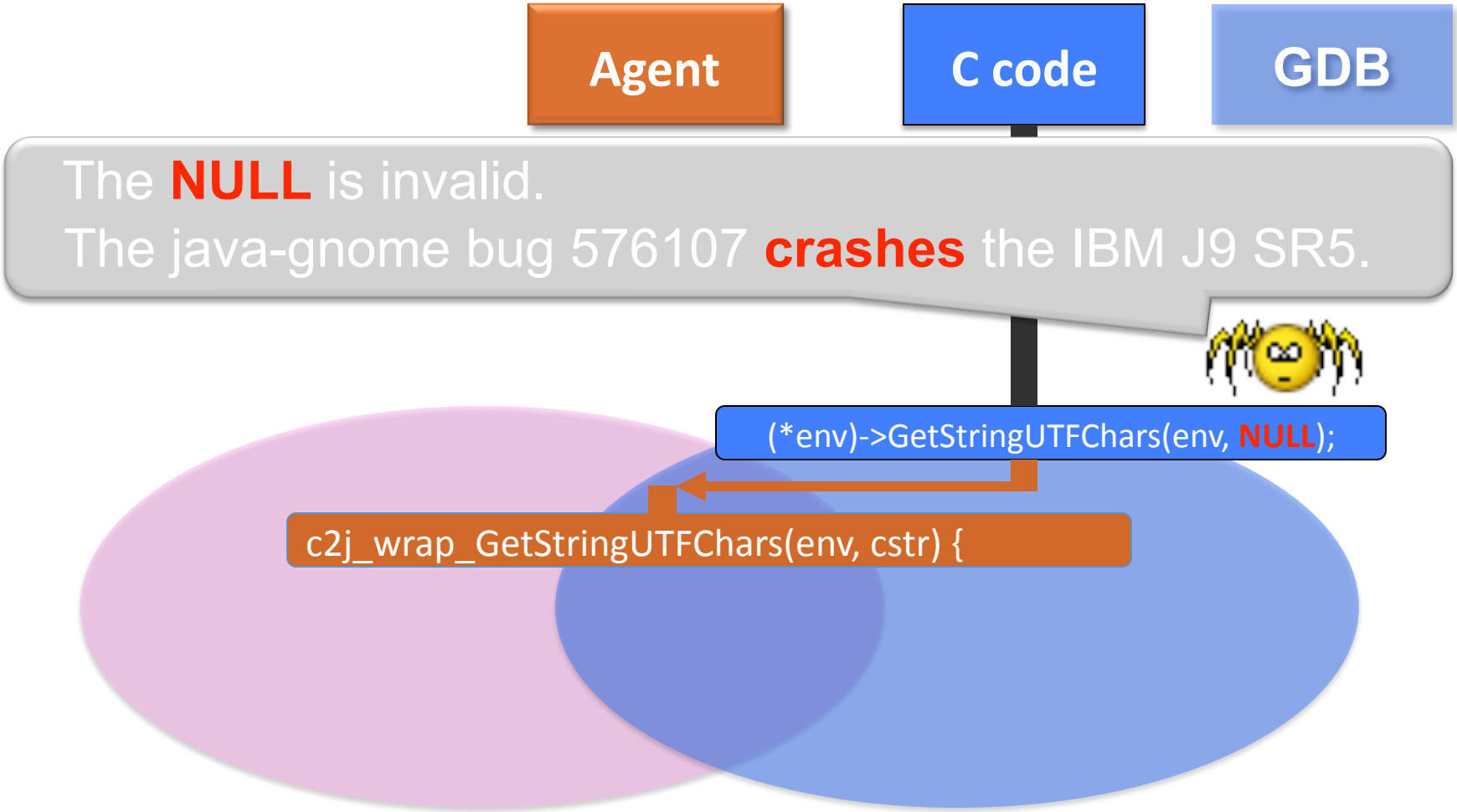
Problem: you may misuse foreign function interface.



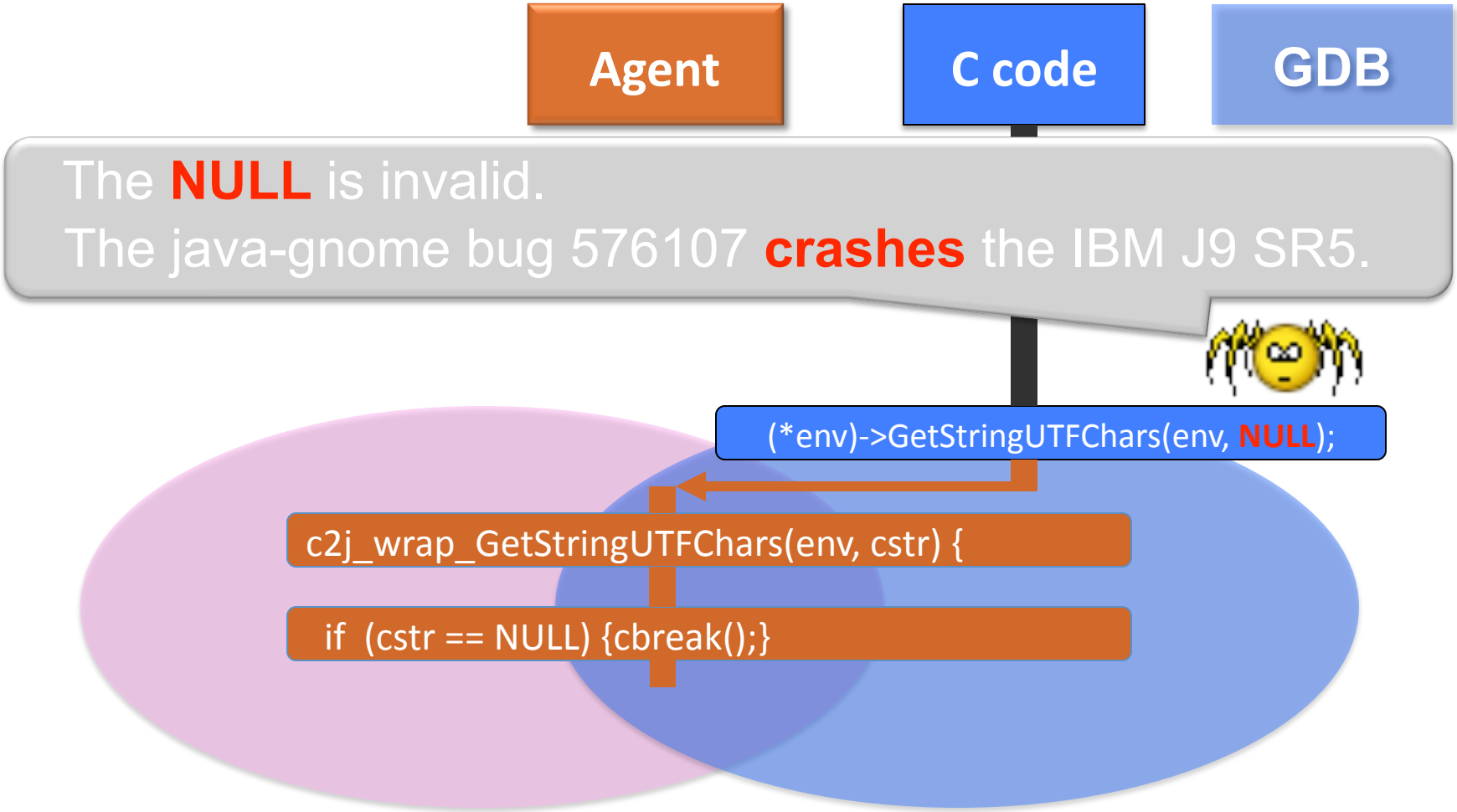
Problem: you may misuse foreign function interface.



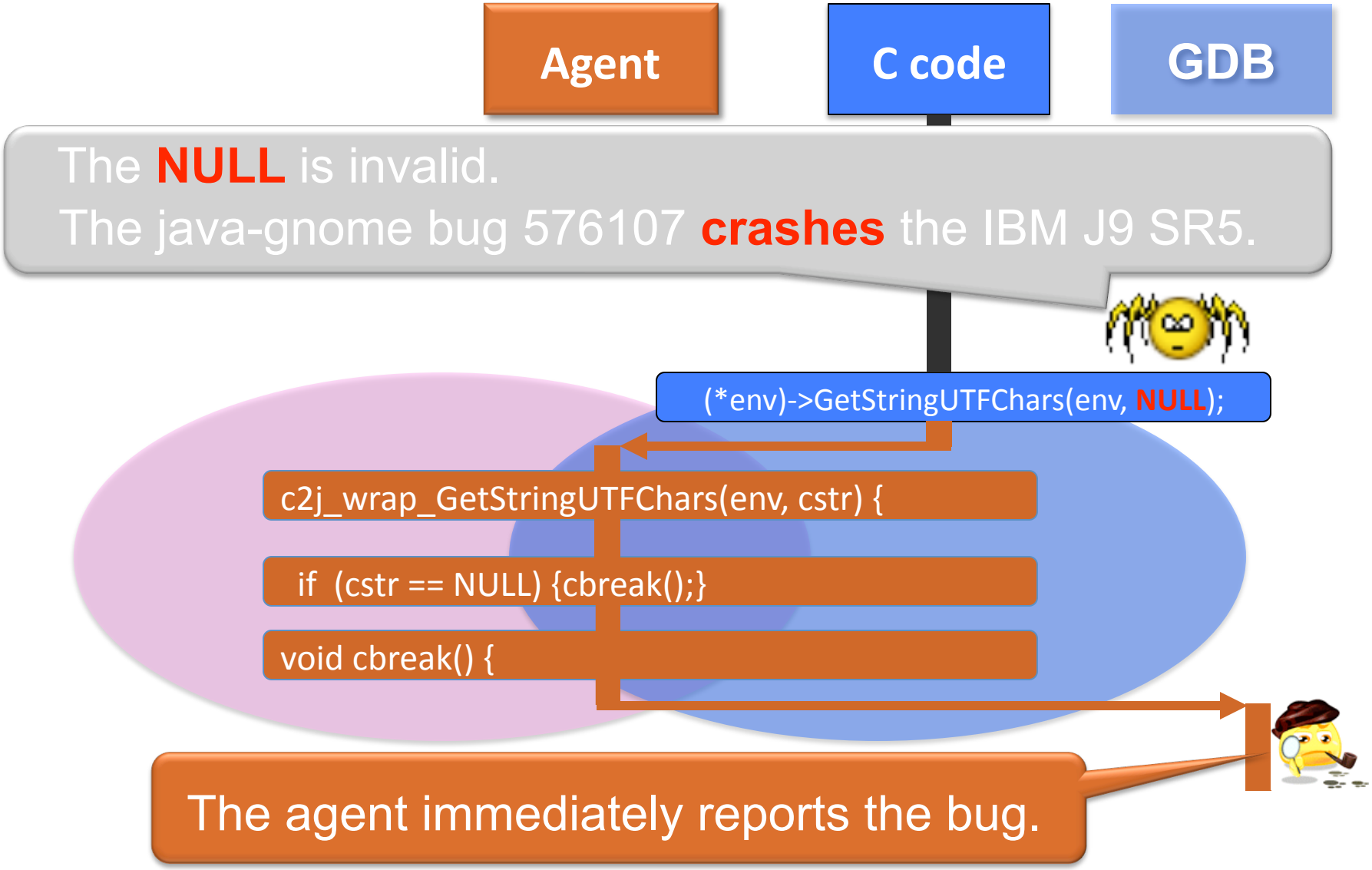
Our solution: detect FFI bugs



Our solution: detect FFI bugs.



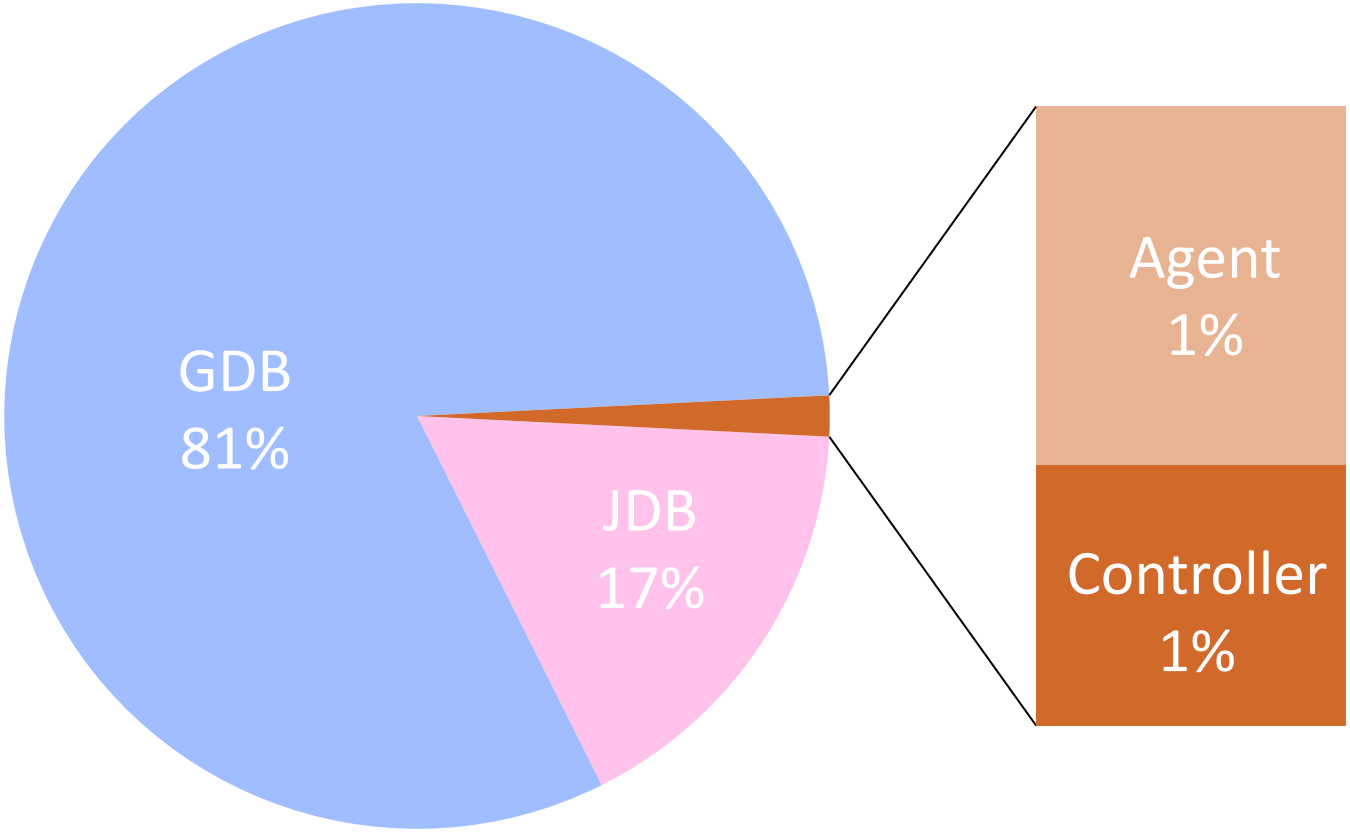
Our solution: detect FFI bugs.



Outline

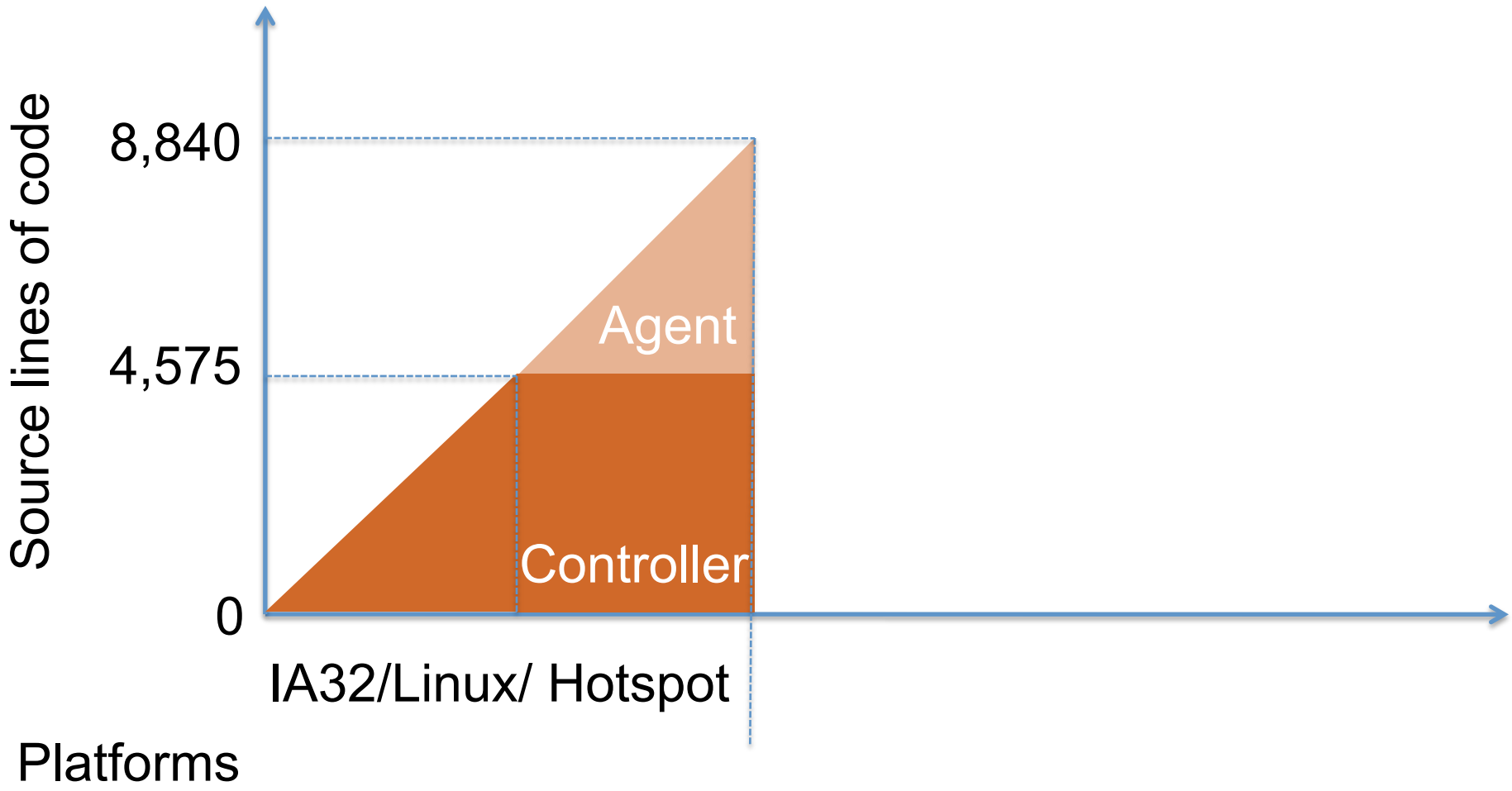
I. Problem
II. Debugger composition
   A. Switching debugger context
   B. Interposing transitions
III. Advanced features
   A. Evaluating Jeannie mixed-environment expressions
   B. Detecting FFI bugs
IV. Evaluation

Composition is simple.

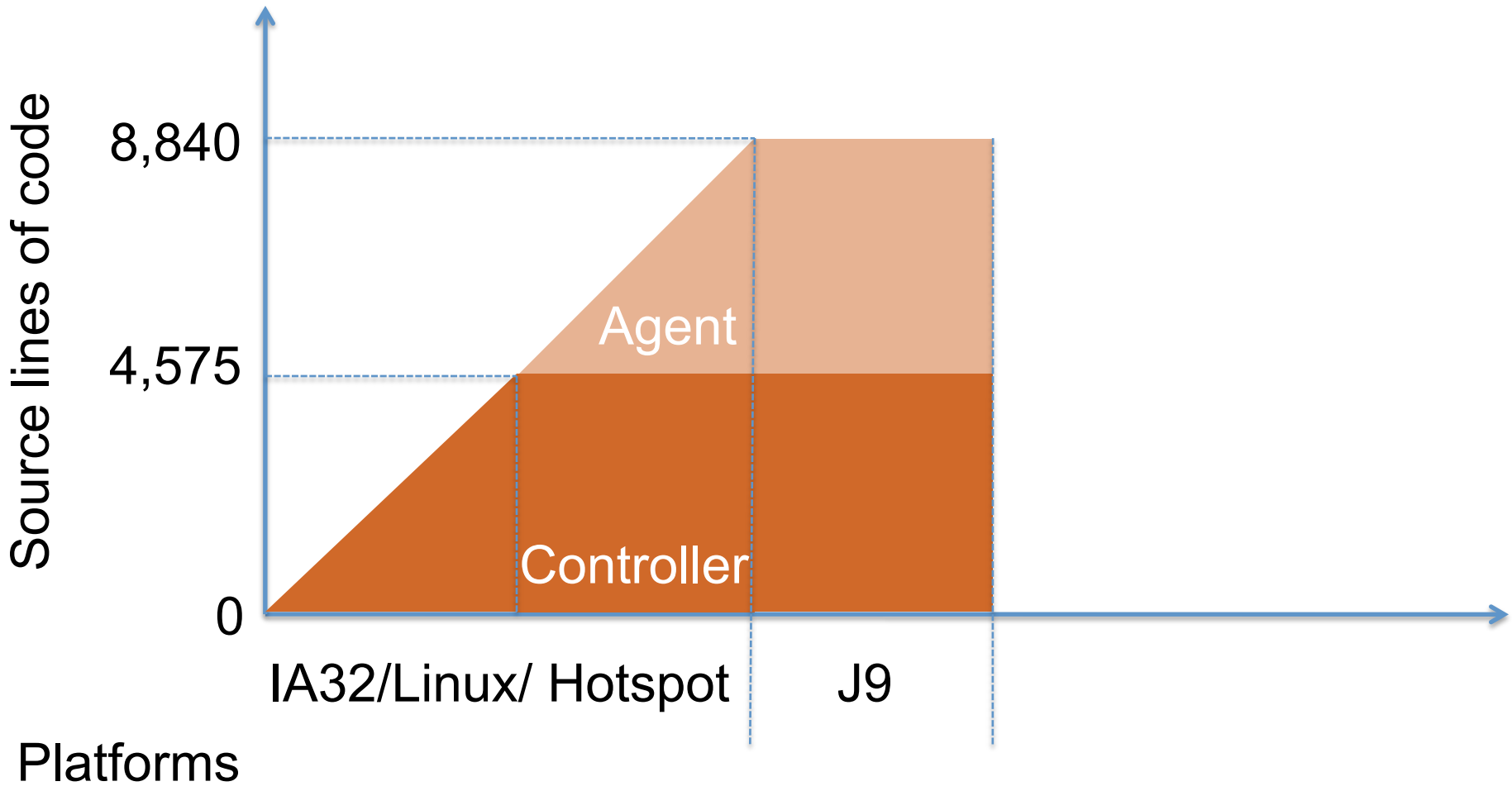


515K source lines of code in total

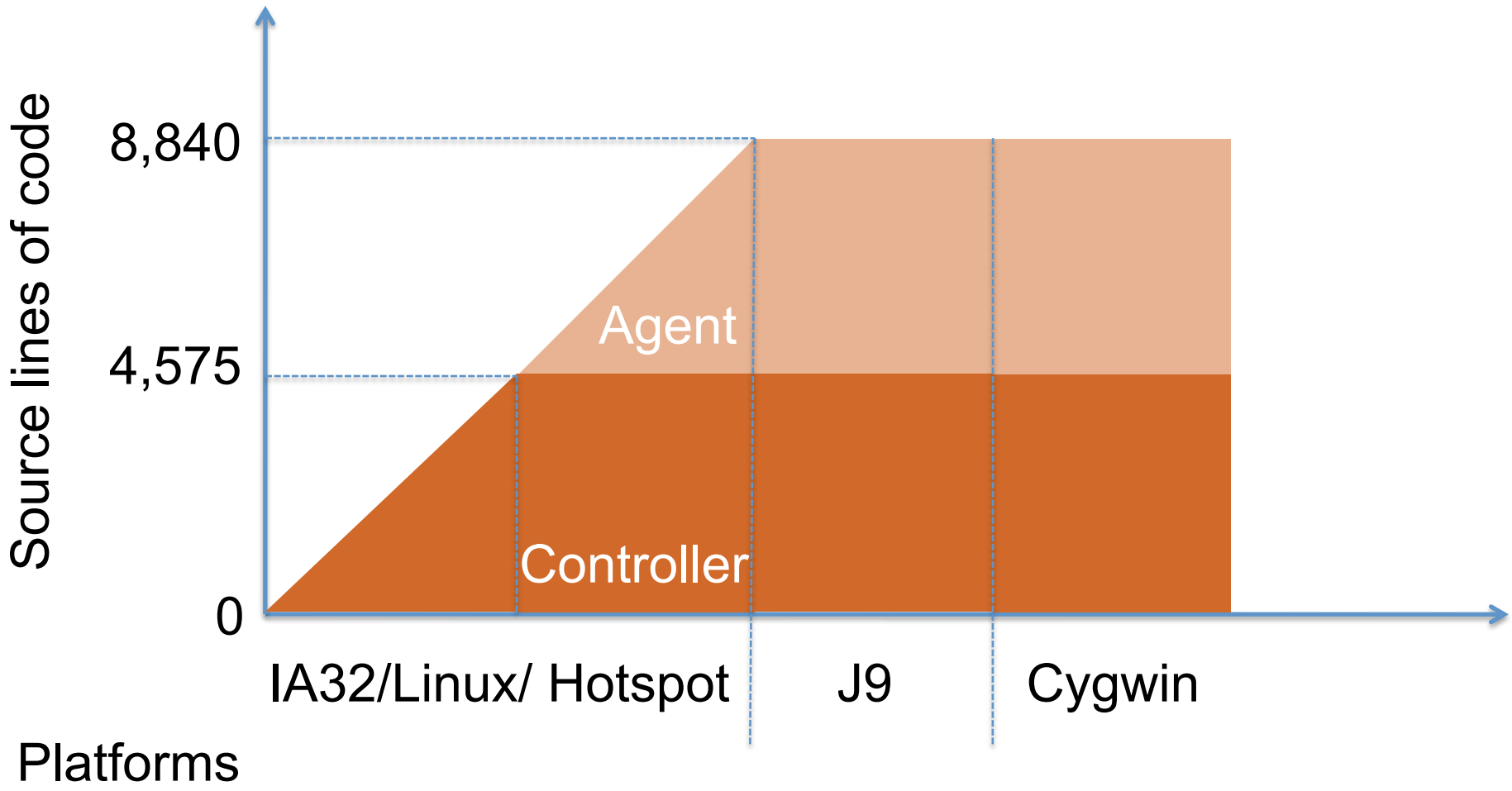
Composition is portable.



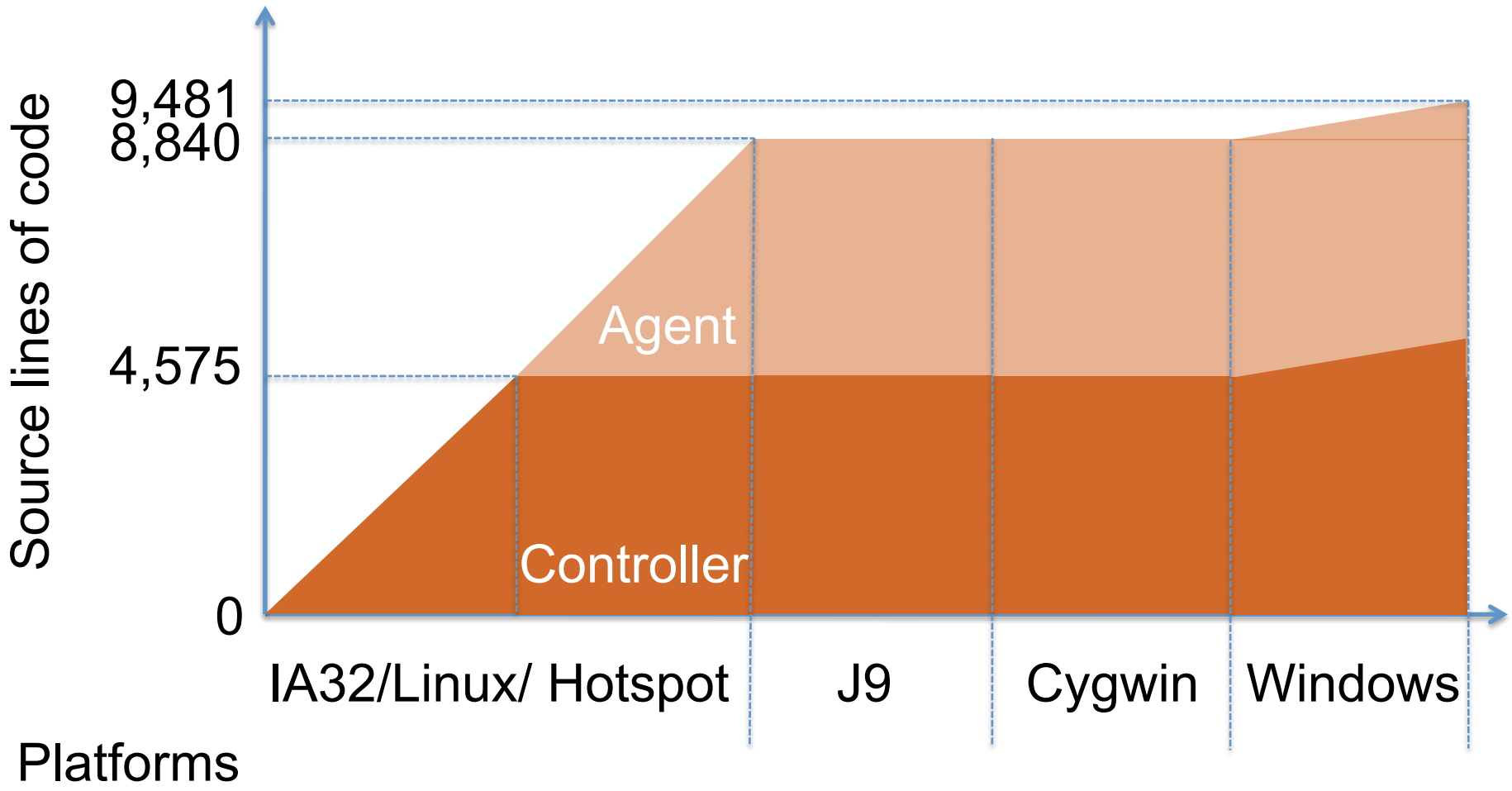
Composition is portable.



Composition is portable.



Composition is portable.



Composition is powerful.

The bug 576107 in java-gnome 4.0.10

	Hotspot VM 1.6.0_10	J9 VM SR5
Production run	running	crash

Composition is powerful.

The bug 576107 in java-gnome 4.0.10

	Hotspot VM 1.6.0_10	J9 VM SR5
Production run	running	crash
Runtime checking (-Xcheck:jni)	warning	warning

Composition is powerful.

The bug 576107 in java-gnome 4.0.10

	Hotspot VM 1.6.0_10	J9 VM SR5
Production run	running	crash
Runtime checking (-Xcheck:jni)	warning	warning
jdb	running	crash
gdb	running	fault

Composition is powerful.

The bug 576107 in java-gnome 4.0.10

	Hotspot VM 1.6.0_10	J9 VM SR5
Production run	running	crash
Runtime checking (-Xcheck:jni)	warning	warning
jdb	running	crash
gdb	running	fault
Blink	**breakpoint**	**breakpoint**

Related work

- Mixed-environment debuggers
  - Intel XDI for Harmony JVM
  - SUN dbx
  - Microsoft .NET debuggers
- Advance debugging features
  - Static analyses
    - BEAM [Kondoh & Onodera '08]
    - J-Saffire [Furr & Foster '06]
    - ILEA [Tan & Morrisett '07]
  - Language designs
    - Jeannie [Hirzel & Grimm '07]
    - SafeJNI [Tan et al. '06]
  - Wrapper generators
    - Automatic binding generator [Ravitch '09]
    - SWIG [Beazley '96]

Summary

- Portable mixed-environment debugging
- Composition with an intermediate agent
  1. Switching debugger context
  2. Interposing transitions
- Results
  1. Simple
  2. Portable
  3. Powerful

Thank you