Source-Level Dataflow-Based Fixes

Experiences from using IntraJ and MagpieBridge

Idriss Riouak - Lund University

EXAMPLE DATAFLOW-BASED ANALYSES

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```

Null pointer Analysis (NPA)

⚠ Possible `NullPointerException` at line 4

```
1  private int hash = 0;
2  int hashFunc(){
3    if(hash==0){
4      int hash = 10;
5      //Complex operations on  hash
6      hash += 10;
7    }
8    return hash;
9  }
```

Dead Assignment Analysis (DAA)

SIMPLIFIED EXAMPLE FROM APACHE FOP (90 KLOC)

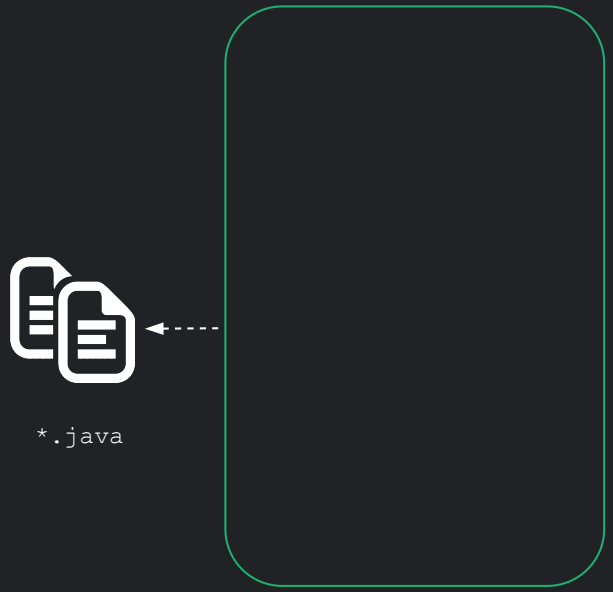
⚠ `Dead Assignment` at line 6. The value of `hash` is never read.

THE BIG PICTURE

`*.java`

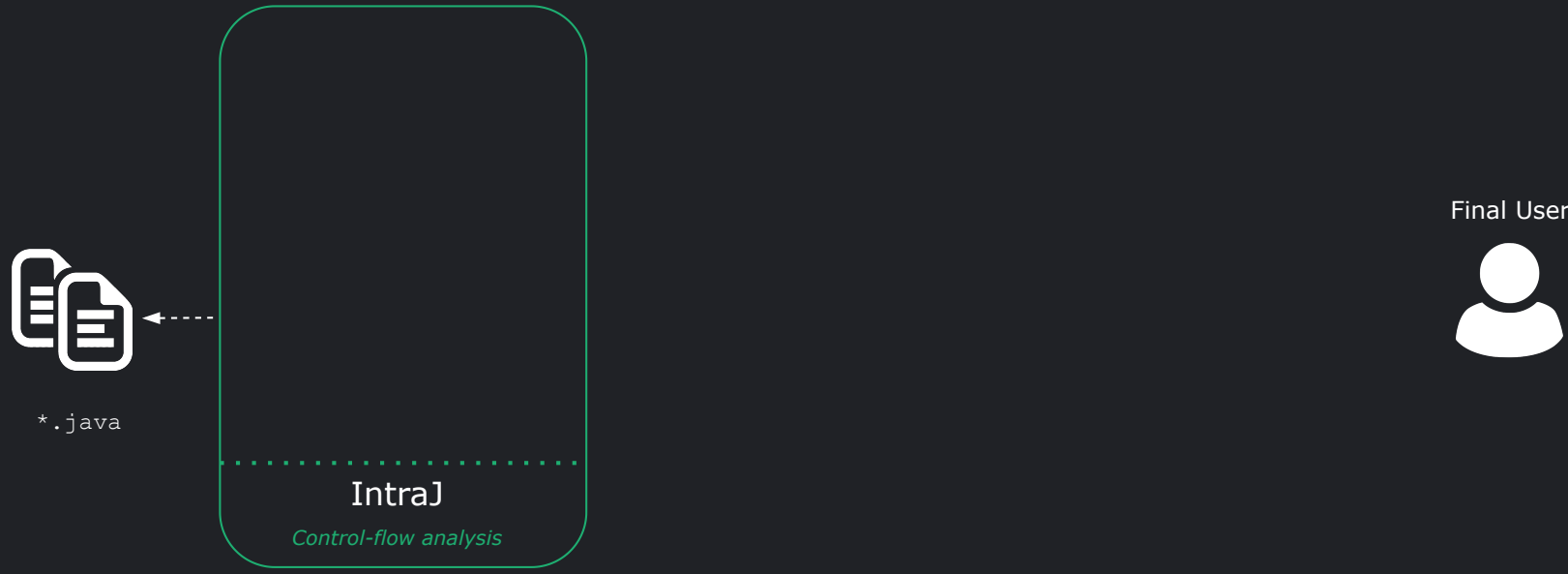
Final User

THE BIG PICTURE

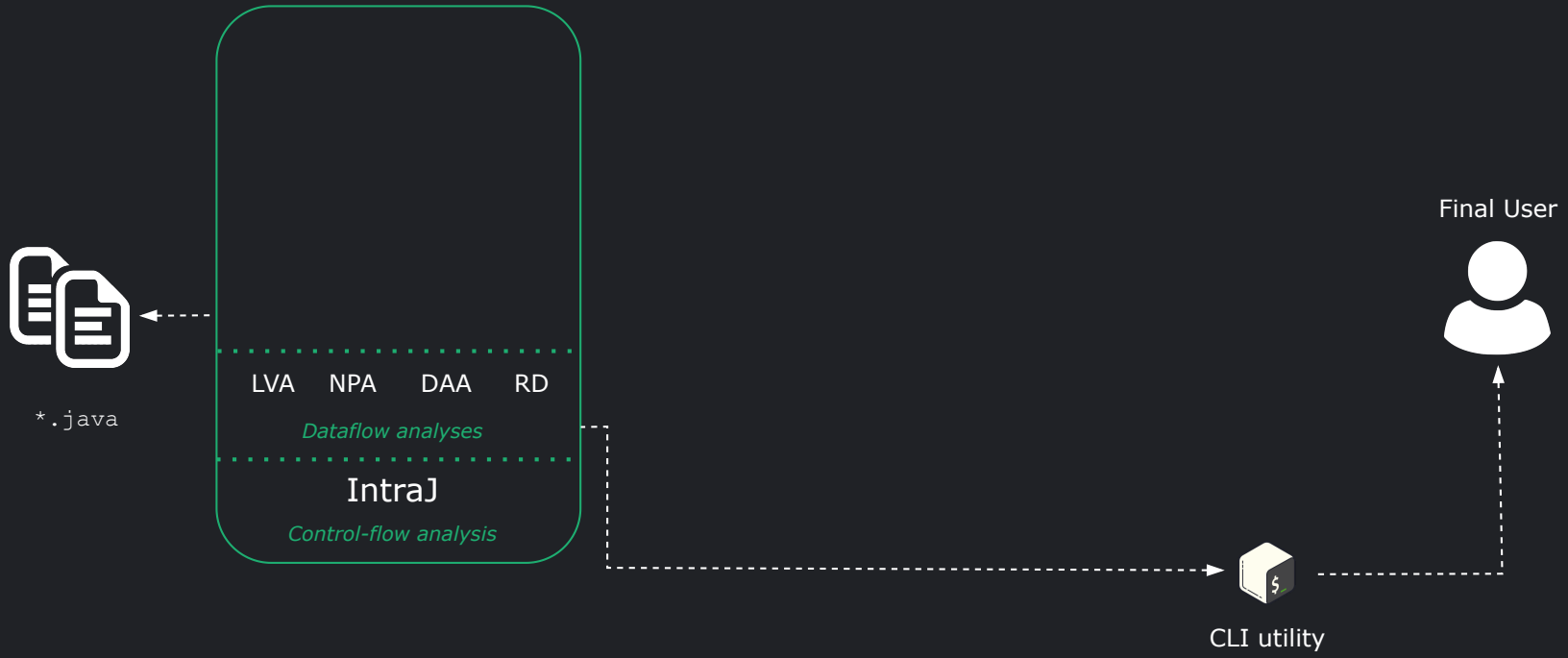


Final User

THE BIG PICTURE



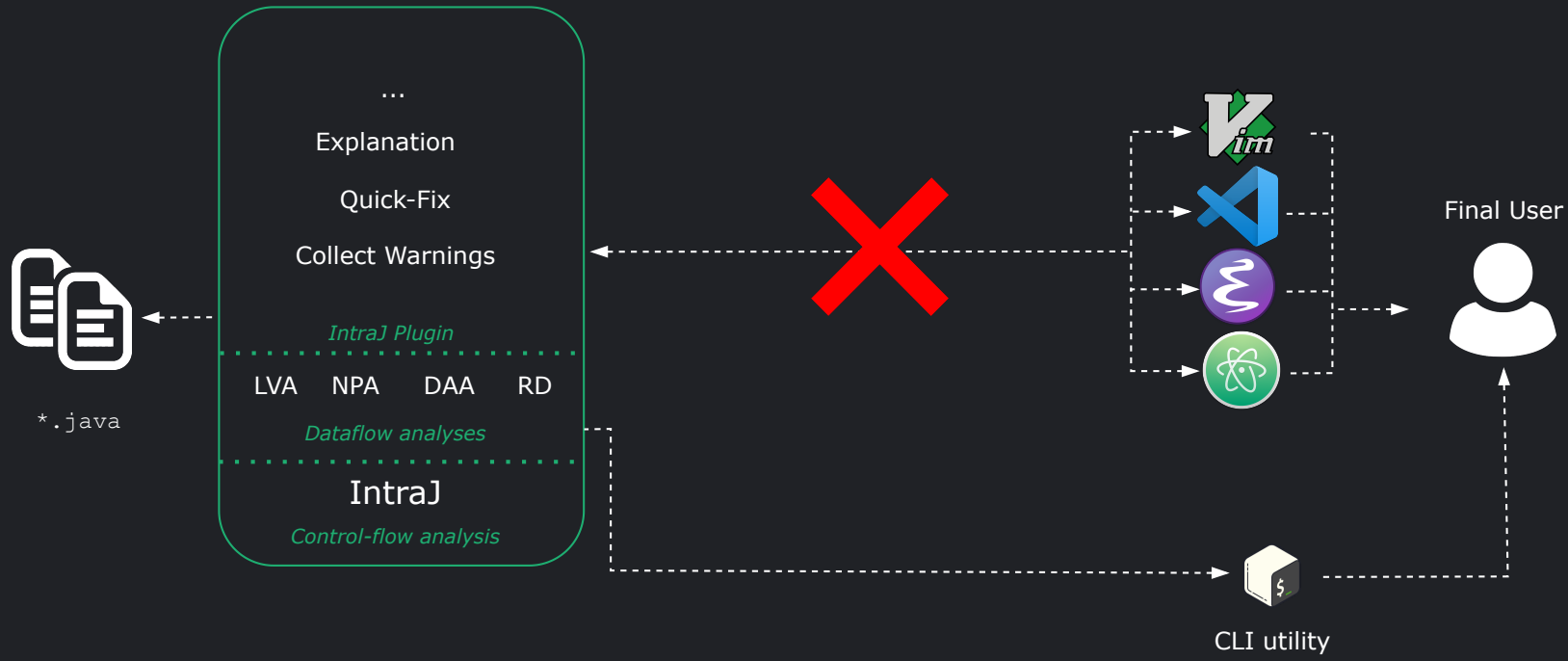
THE BIG PICTURE



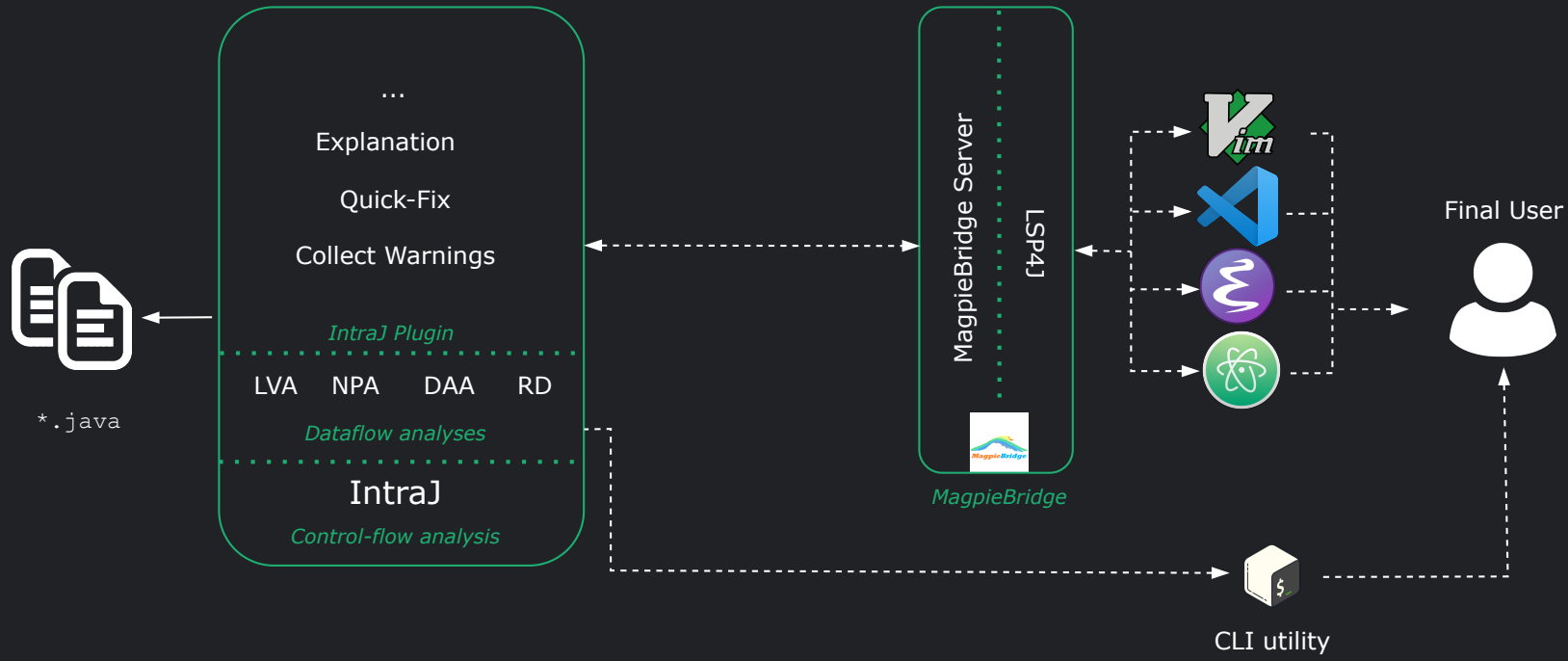
THE BIG PICTURE

THE BIG PICTURE

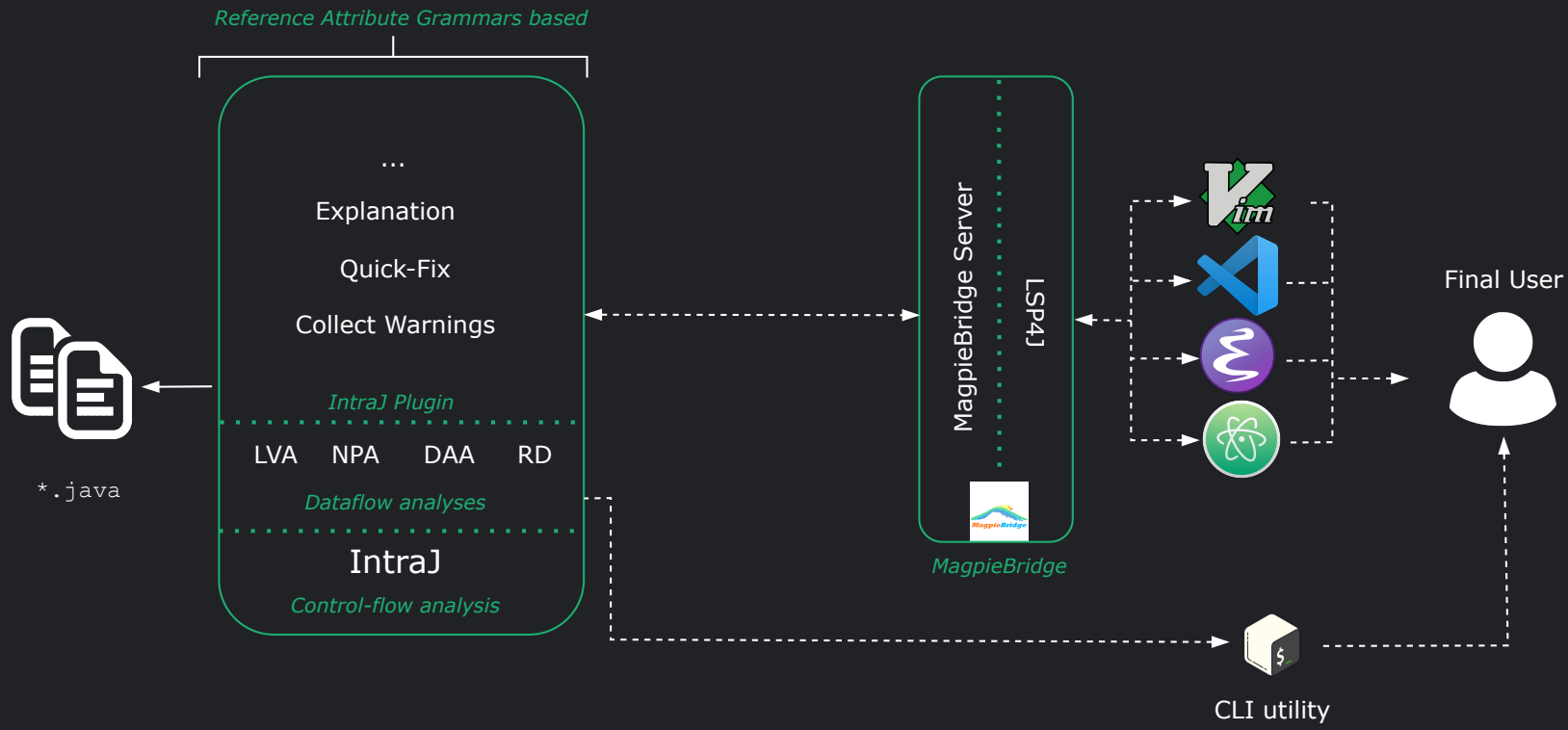
THE BIG PICTURE



THE BIG PICTURE

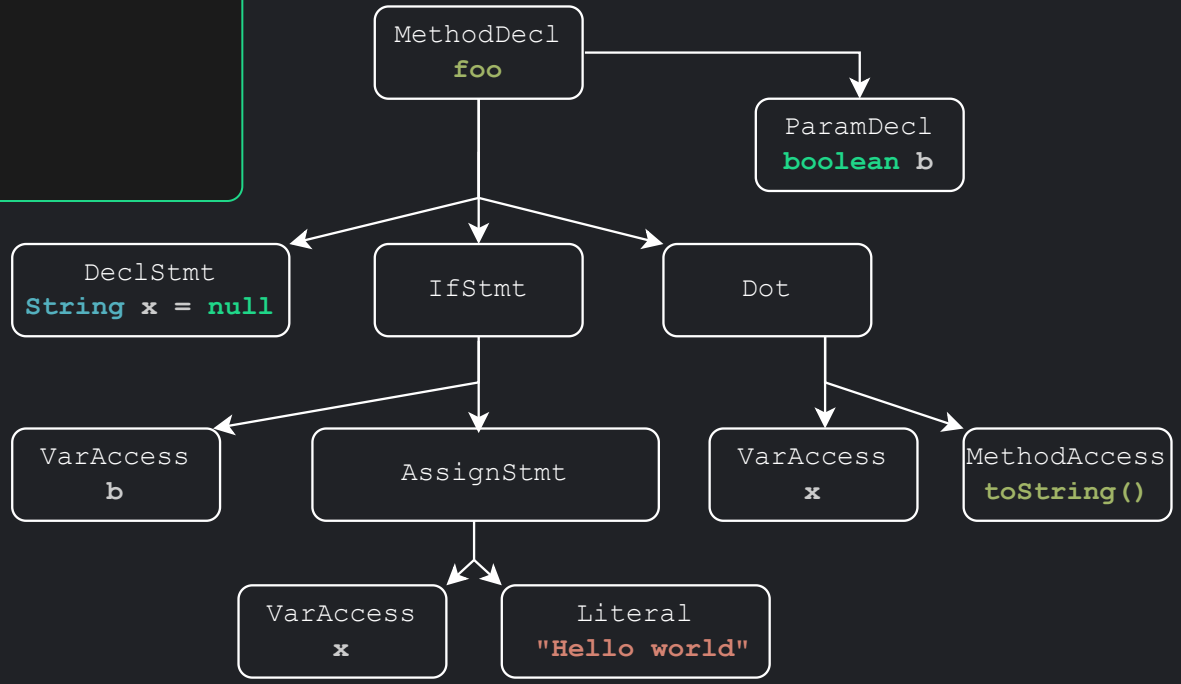


THE BIG PICTURE



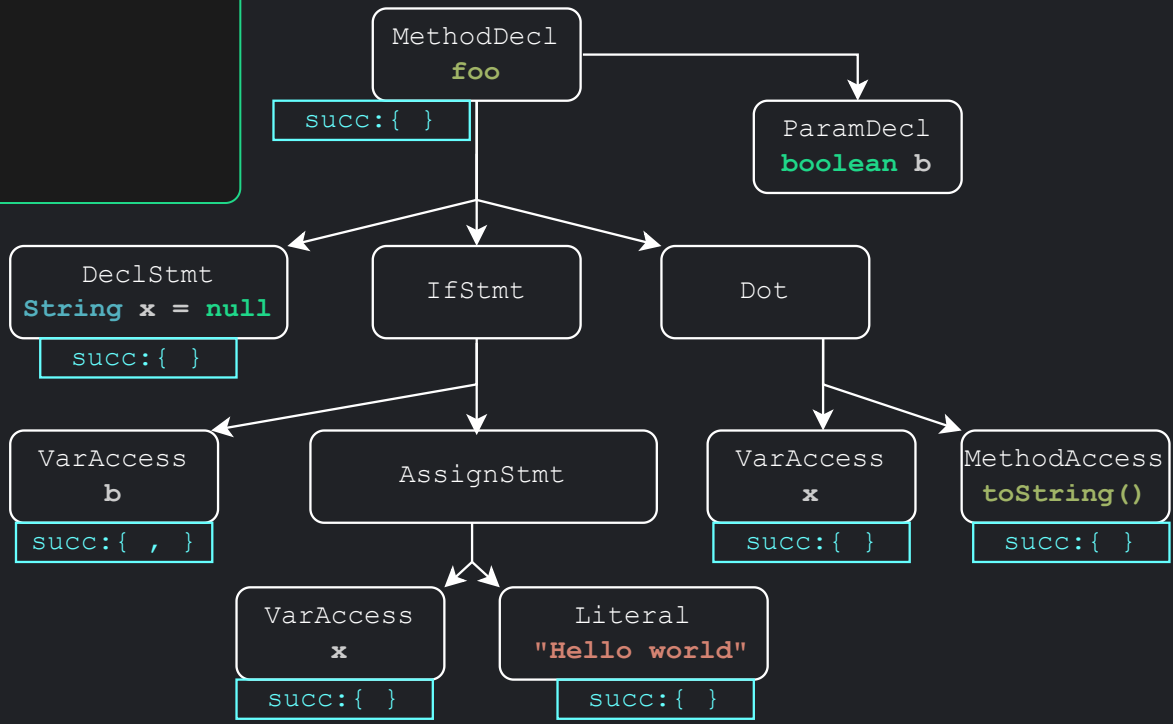
REFERENCE ATTRIBUTE GRAMMARS

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```



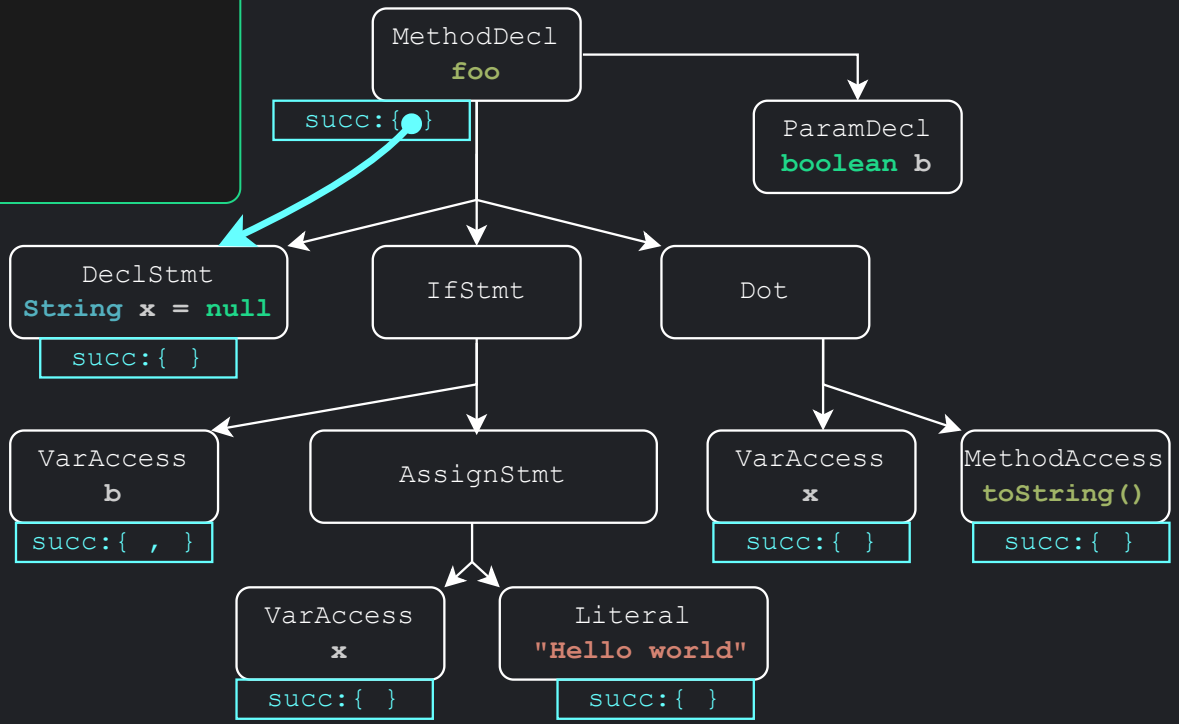
REFERENCE ATTRIBUTE GRAMMARS

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```



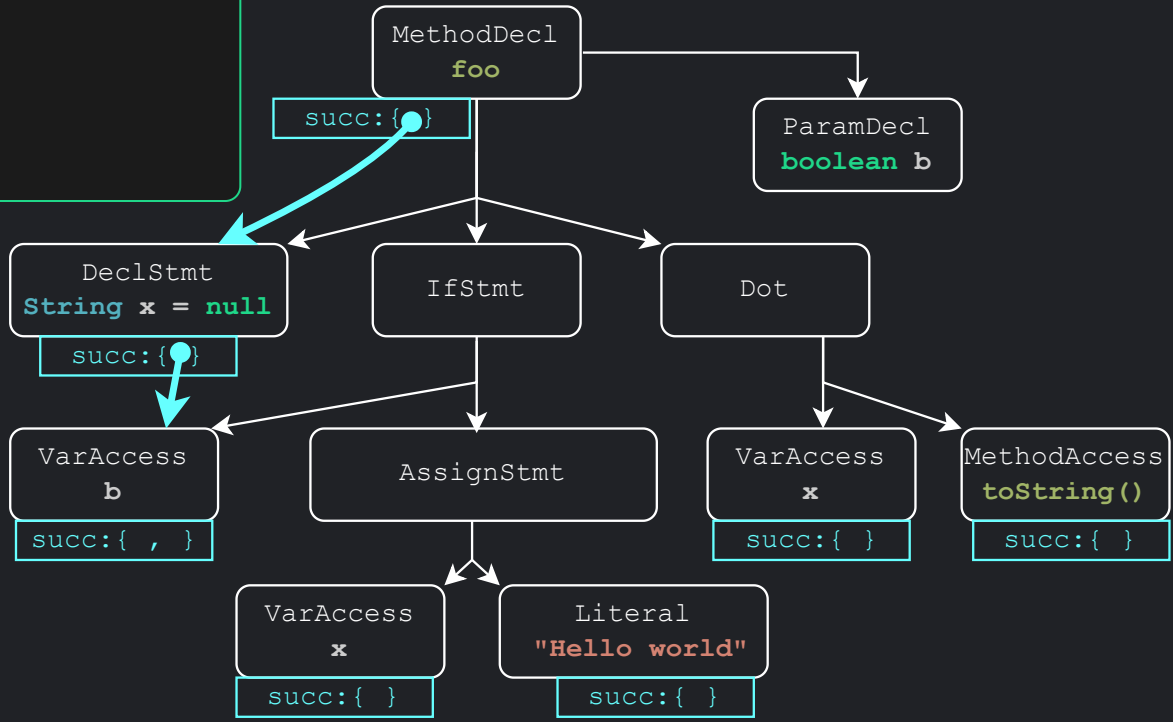
REFERENCE ATTRIBUTE GRAMMARS

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```



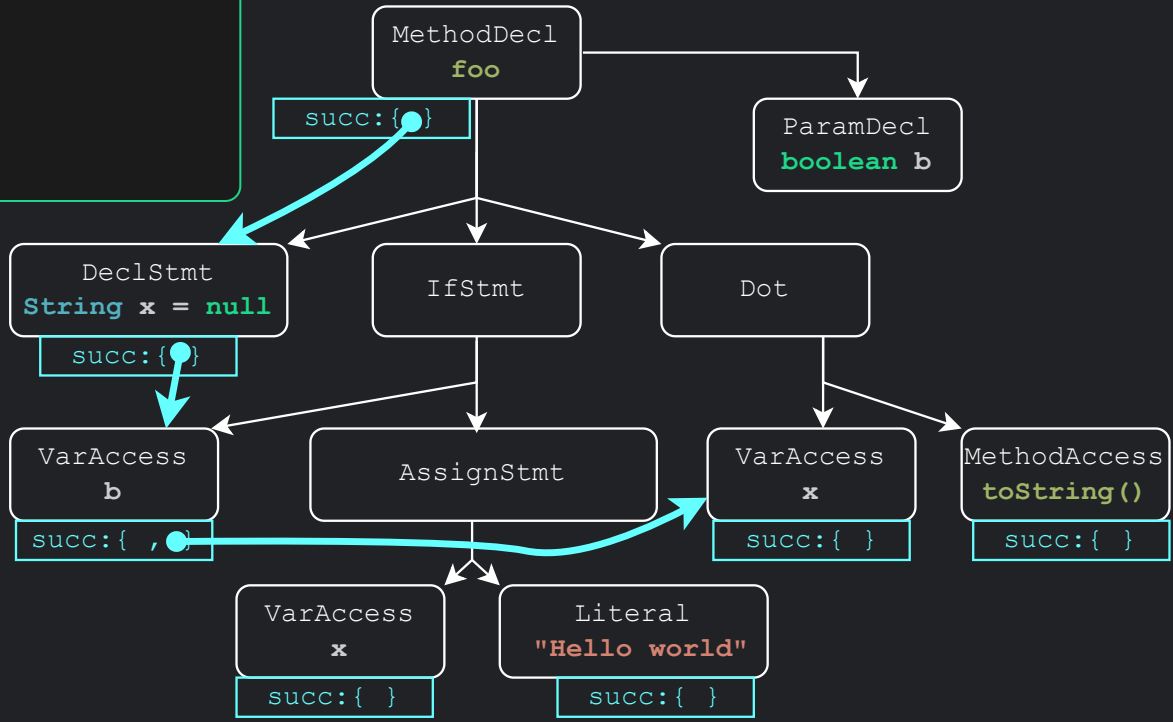
REFERENCE ATTRIBUTE GRAMMARS

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```



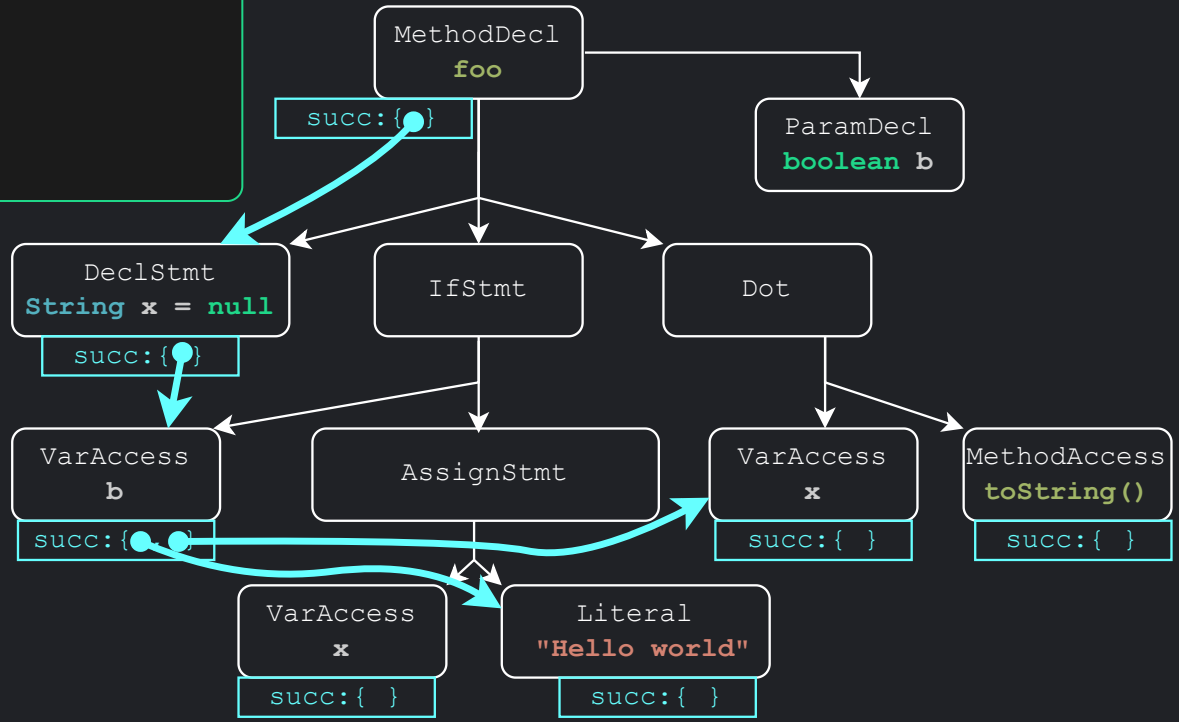
REFERENCE ATTRIBUTE GRAMMARS

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```



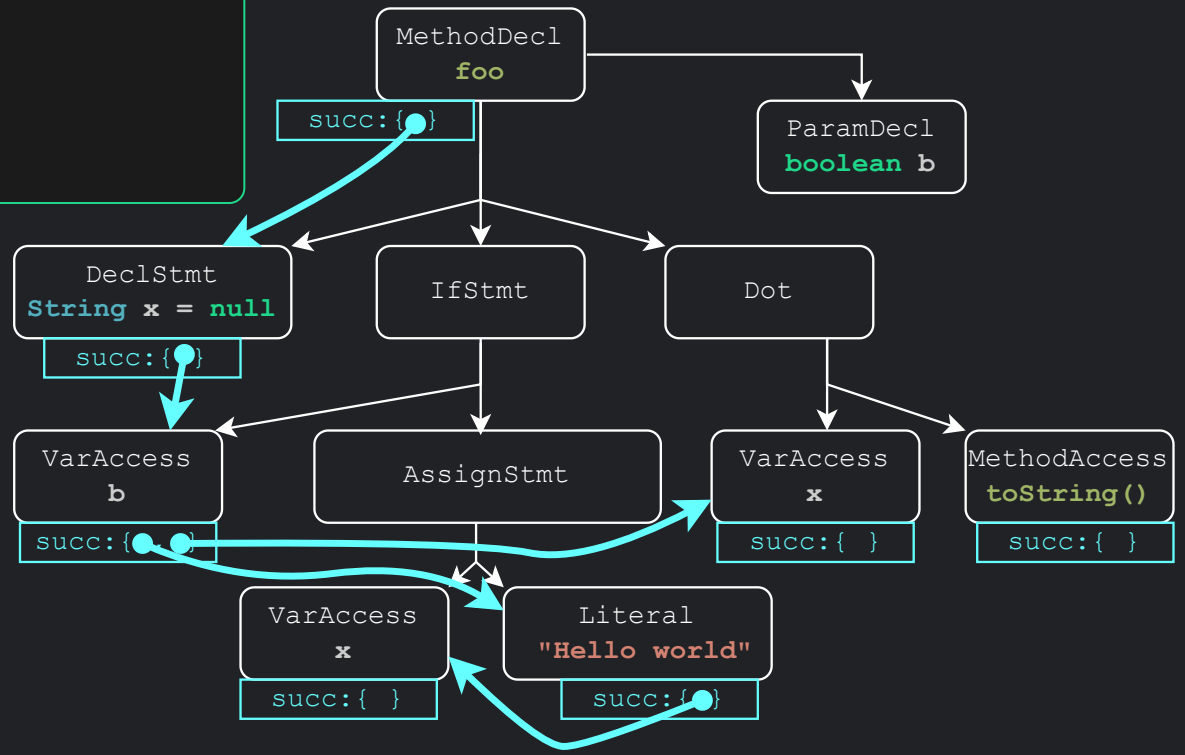
REFERENCE ATTRIBUTE GRAMMARS

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```



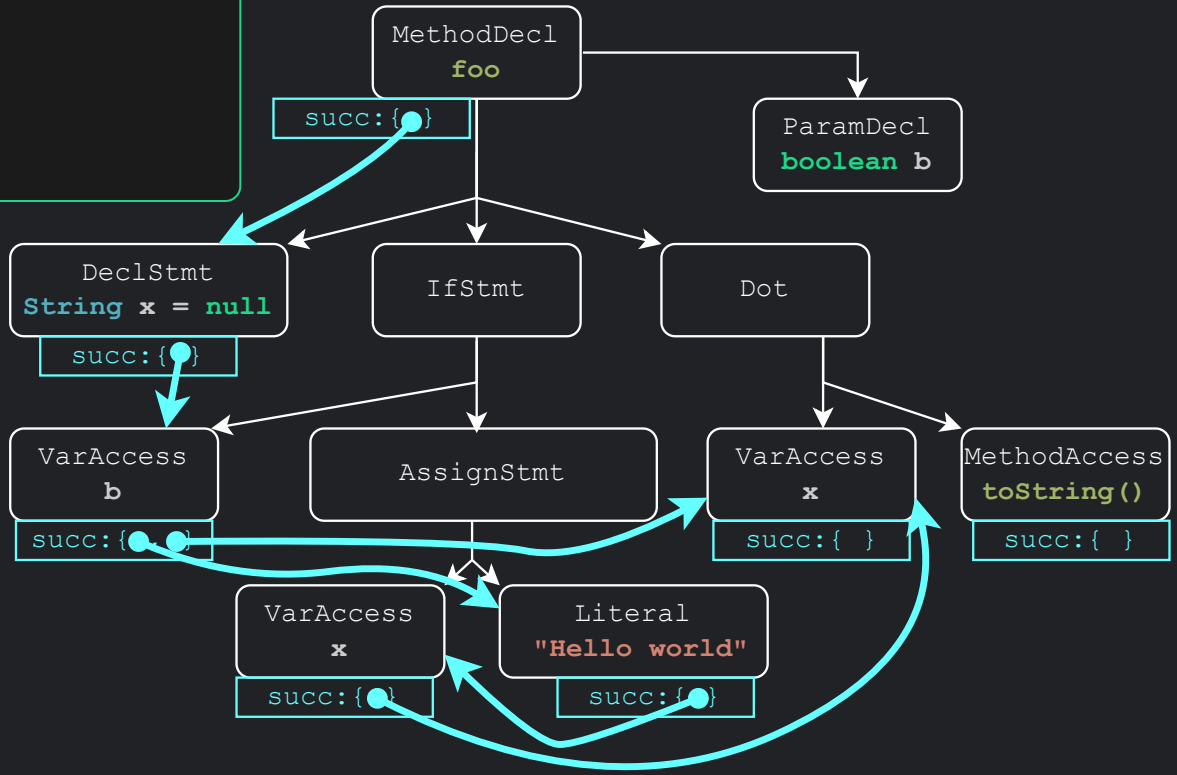
REFERENCE ATTRIBUTE GRAMMARS

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```



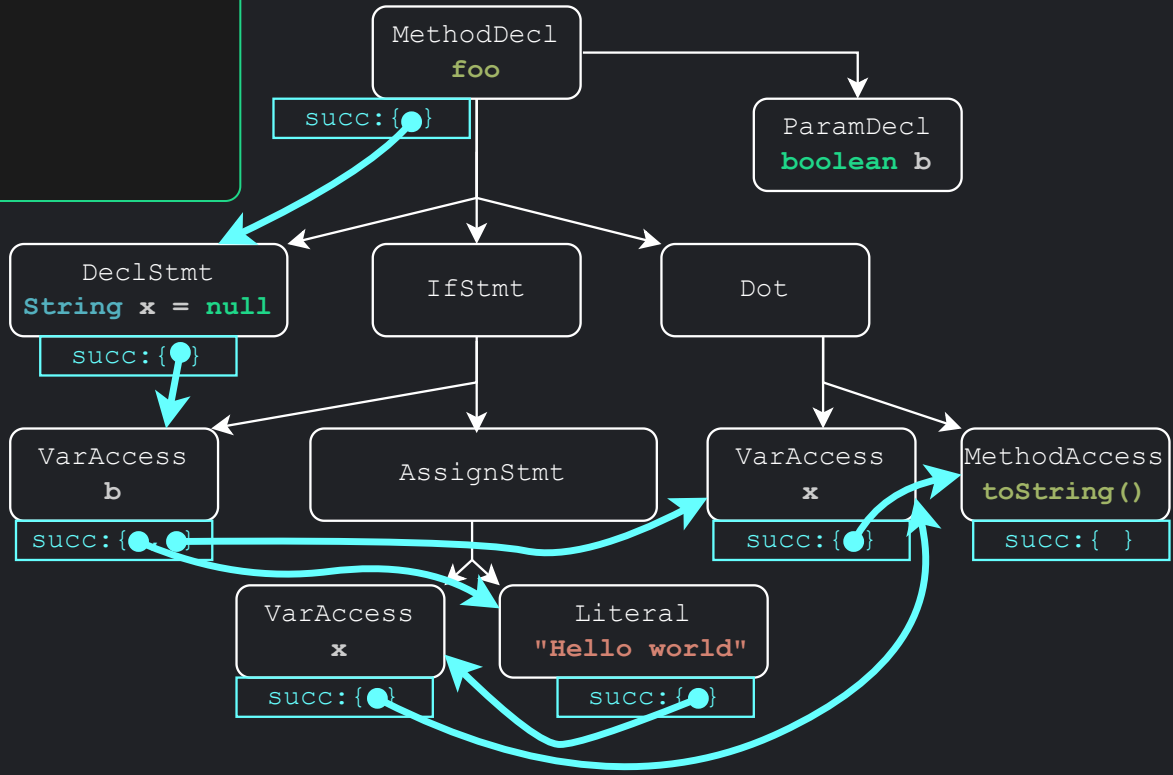
REFERENCE ATTRIBUTE GRAMMARS

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```



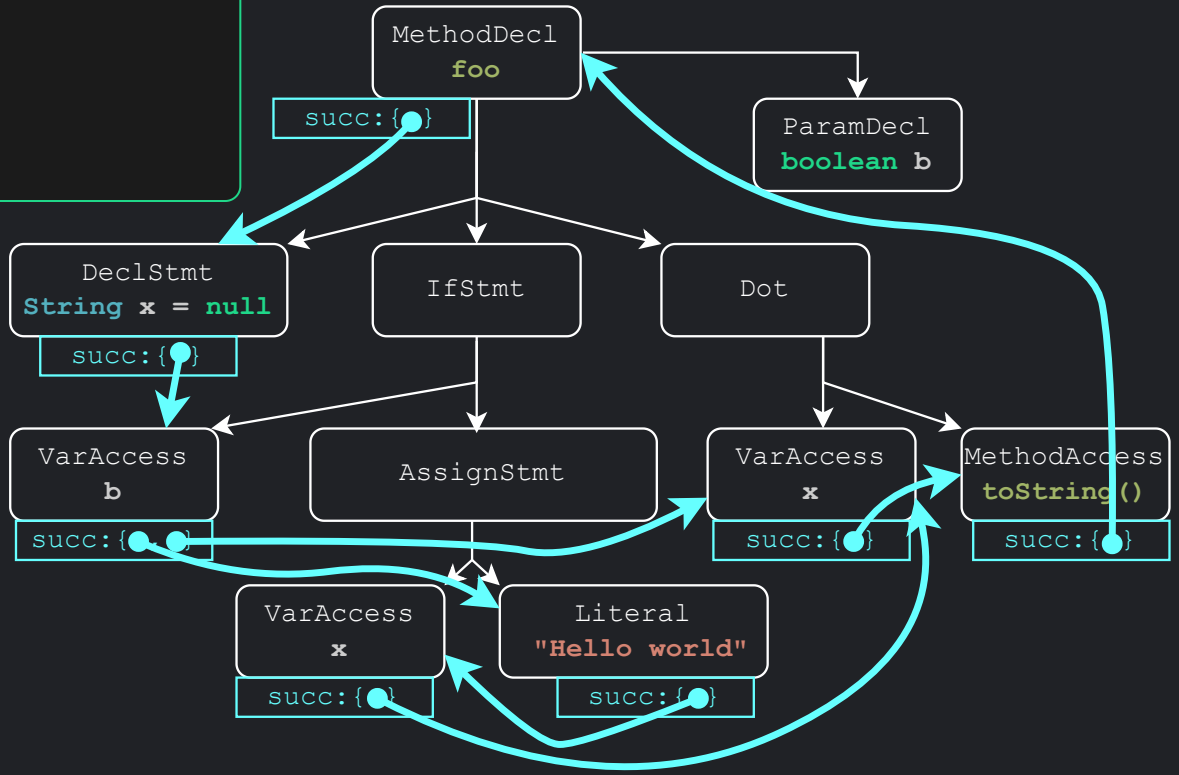
REFERENCE ATTRIBUTE GRAMMARS

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```



REFERENCE ATTRIBUTE GRAMMARS

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```

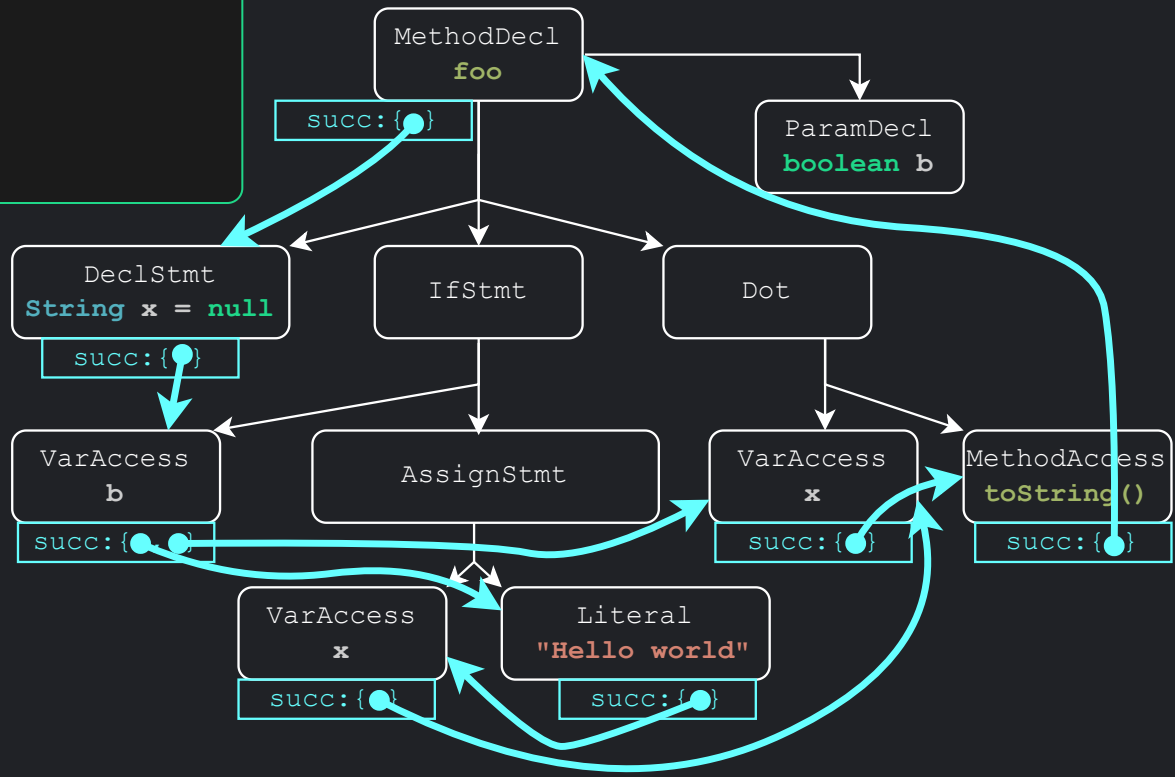


REFERENCE ATTRIBUTE GRAMMARS

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```

- JastAdd ecosystem
  - On-demand evaluation
  - Fix point computation
  - Higher-Order Attibutes

jastadd



INTRAJ

- Builds the CFGs on the AST
- Handles ***implicit control-flows***
- Analyses competitive with existing tools e.g., ***SonarQube***

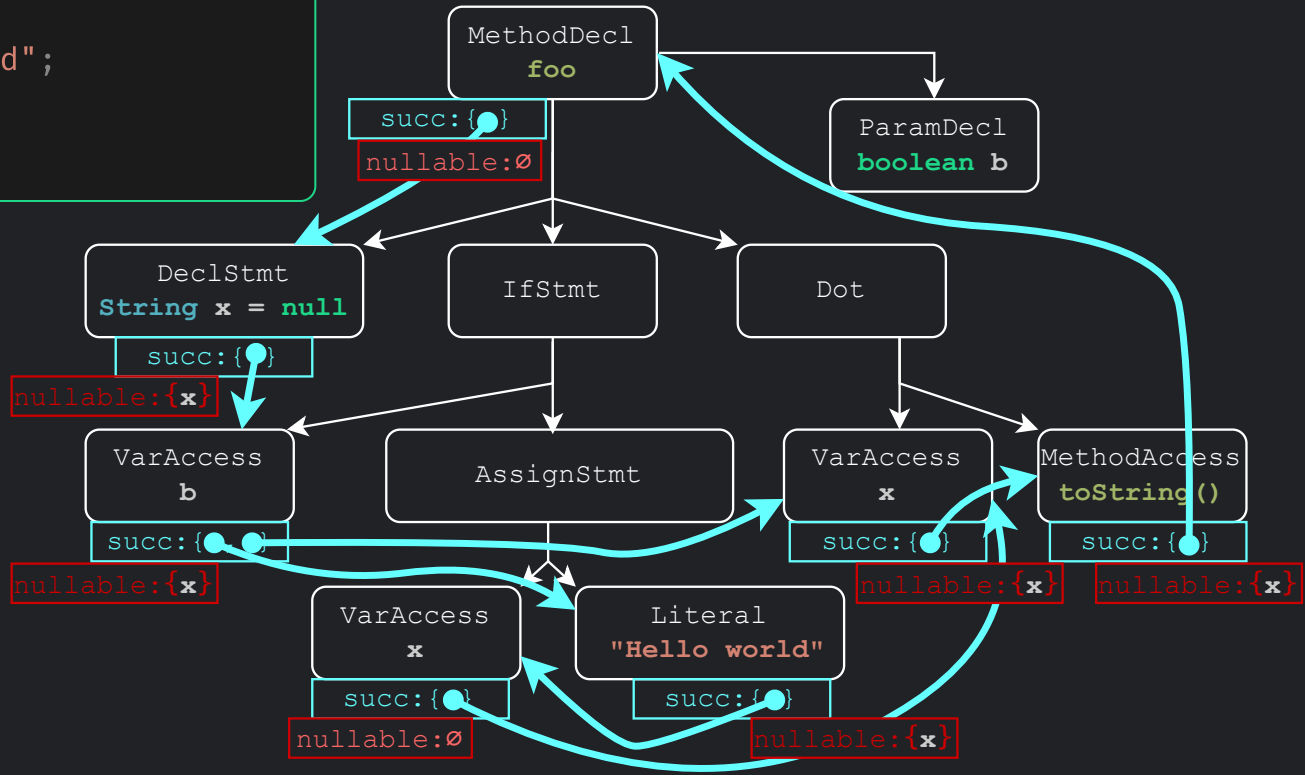
If you want to know more ...

GitHub

Paper

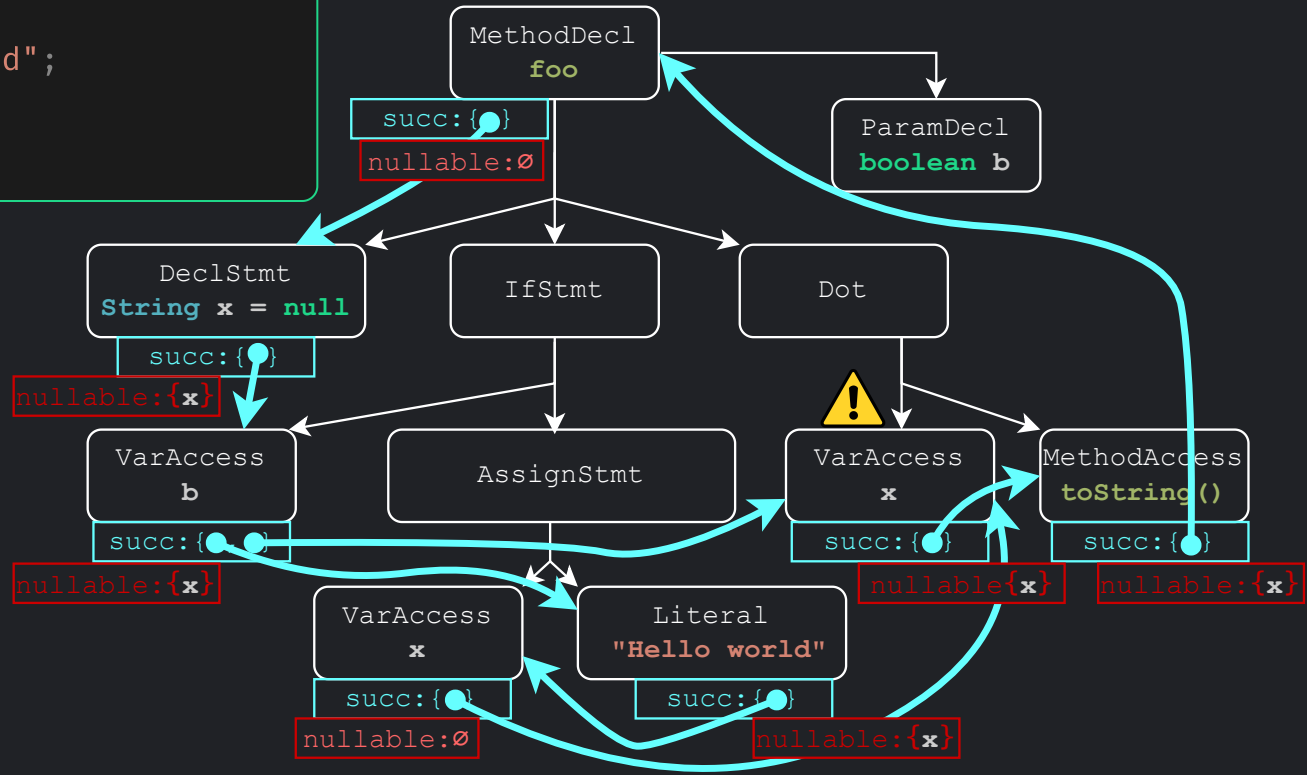
NULL POINTER ANALYSIS

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```

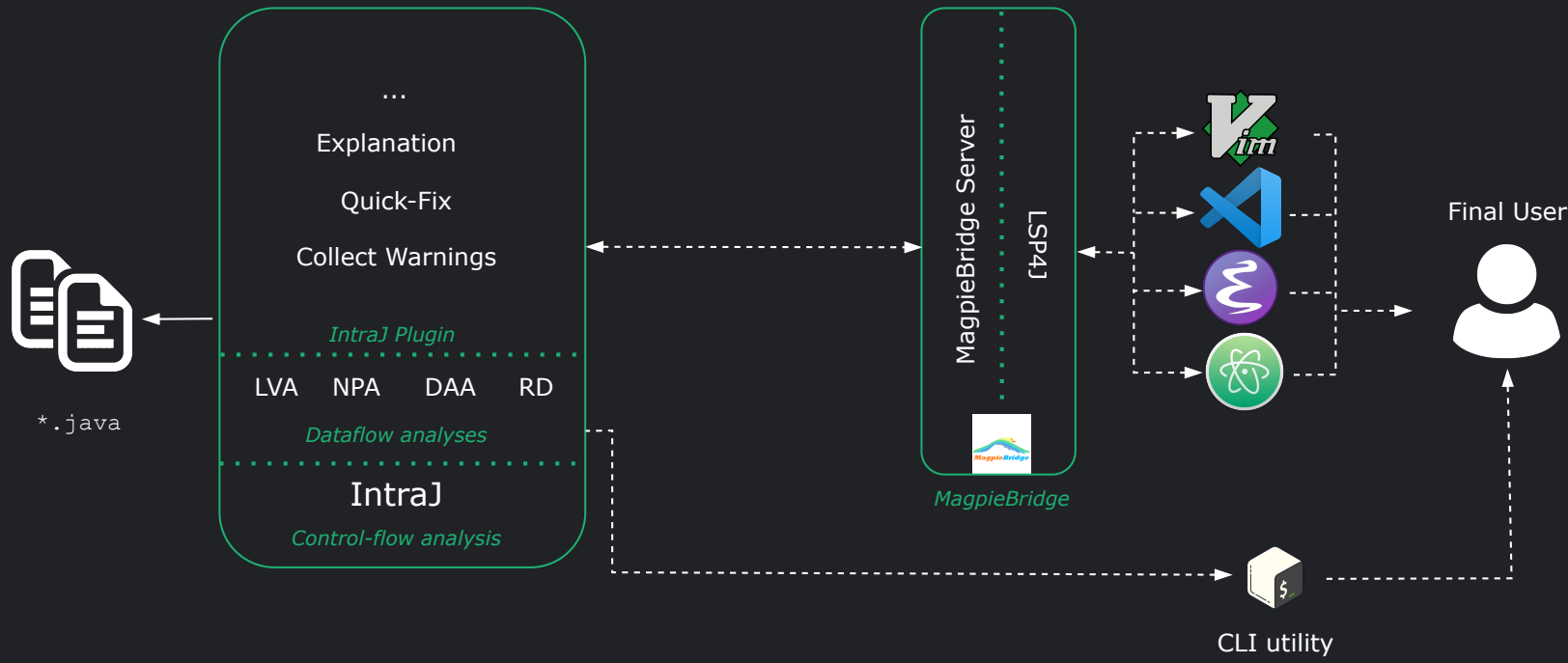


NULL POINTER ANALYSIS

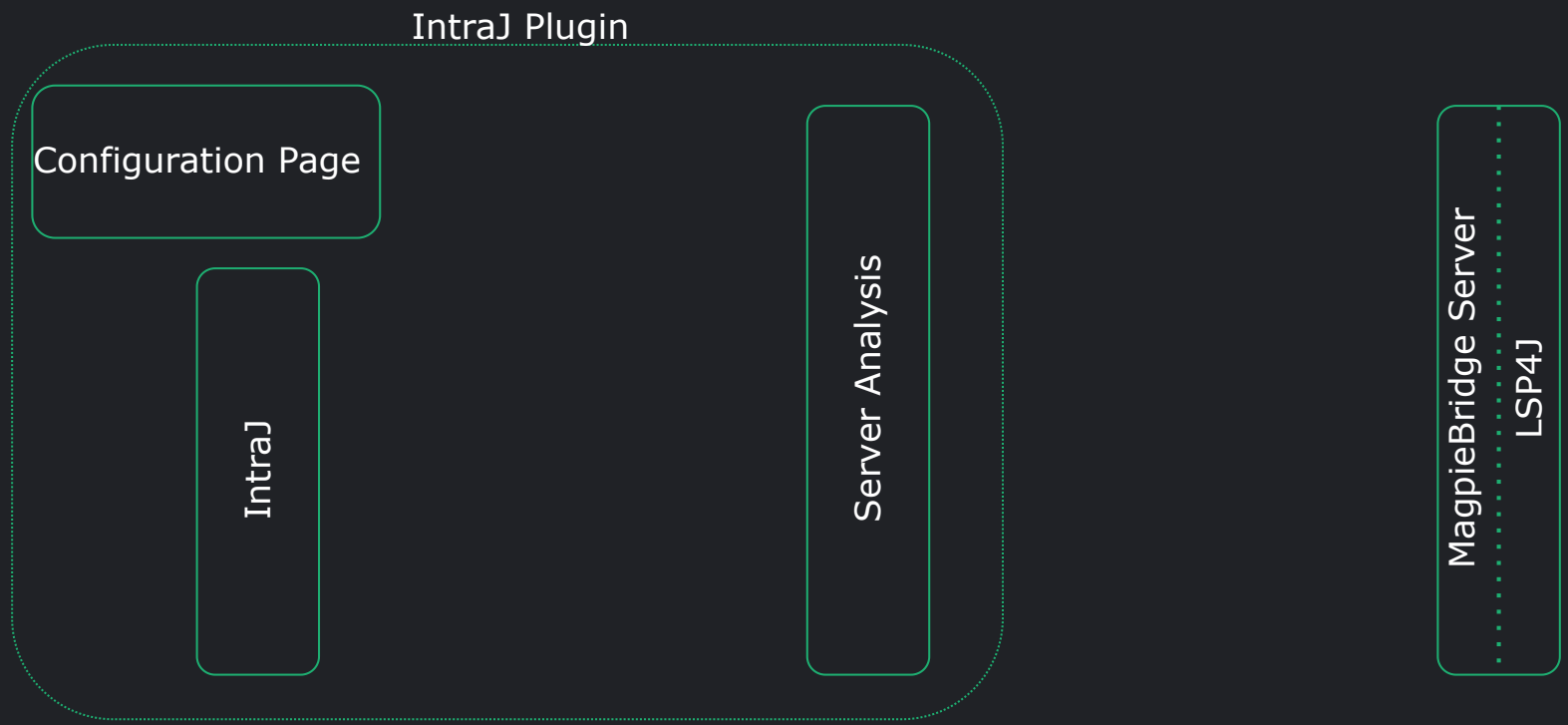
```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```



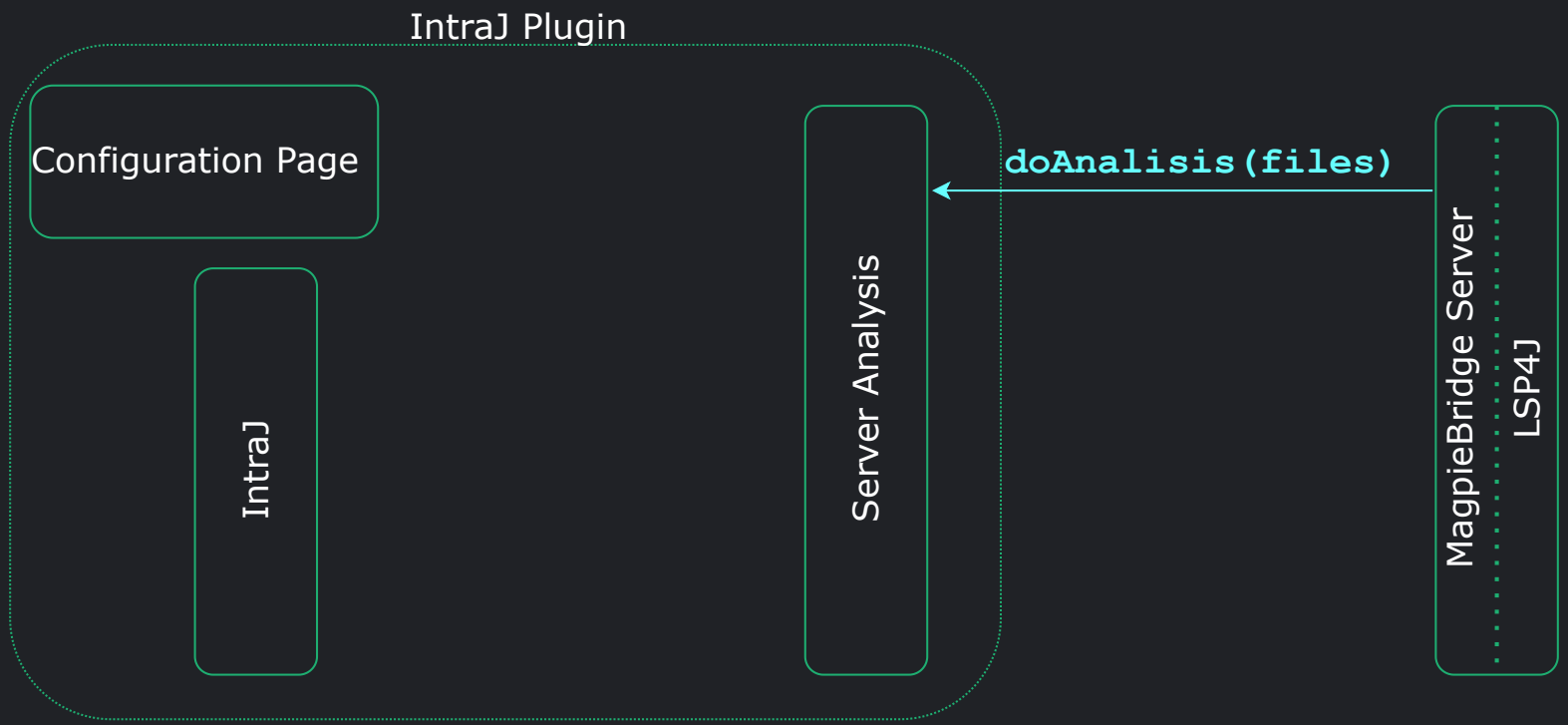
THE BIG PICTURE, AGAIN



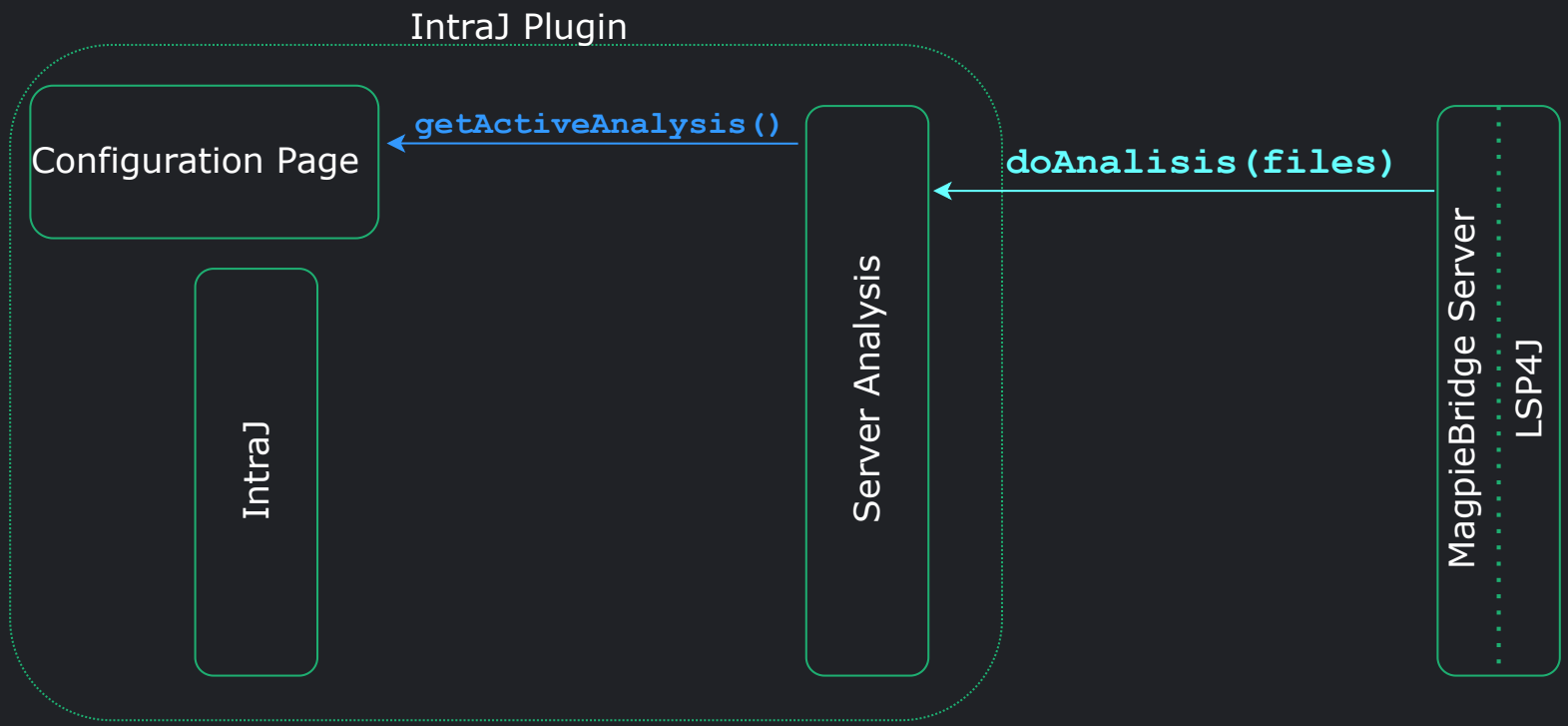
ZOOM-IN


IntraJ Plugin
Configuration Page
IntraJ
Server Analysis
MagpieBridge Server
LSP4J

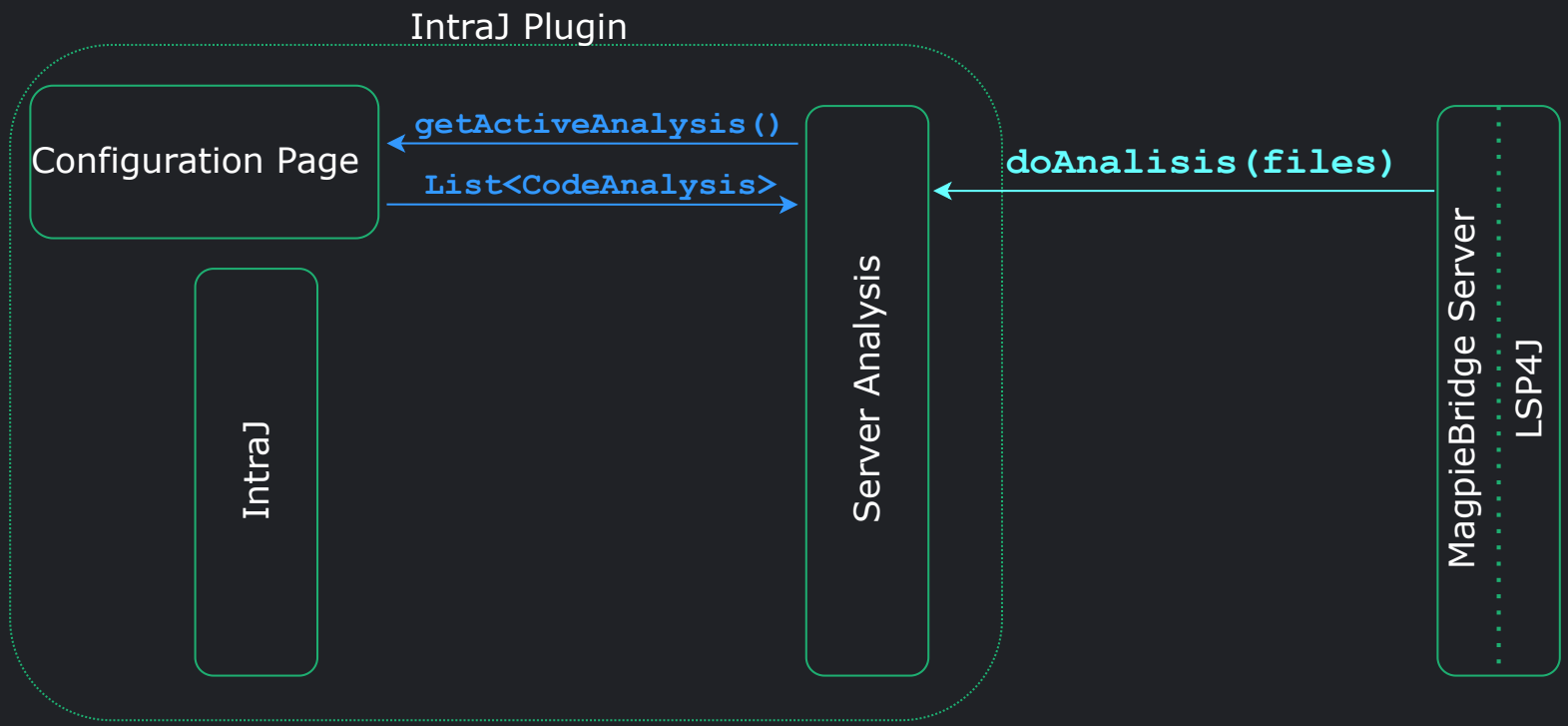
ZOOM-IN


IntraJ Plugin
Configuration Page
IntraJ
Server Analysis
doAnalisis(files)
MagpieBridge Server
LSP4J

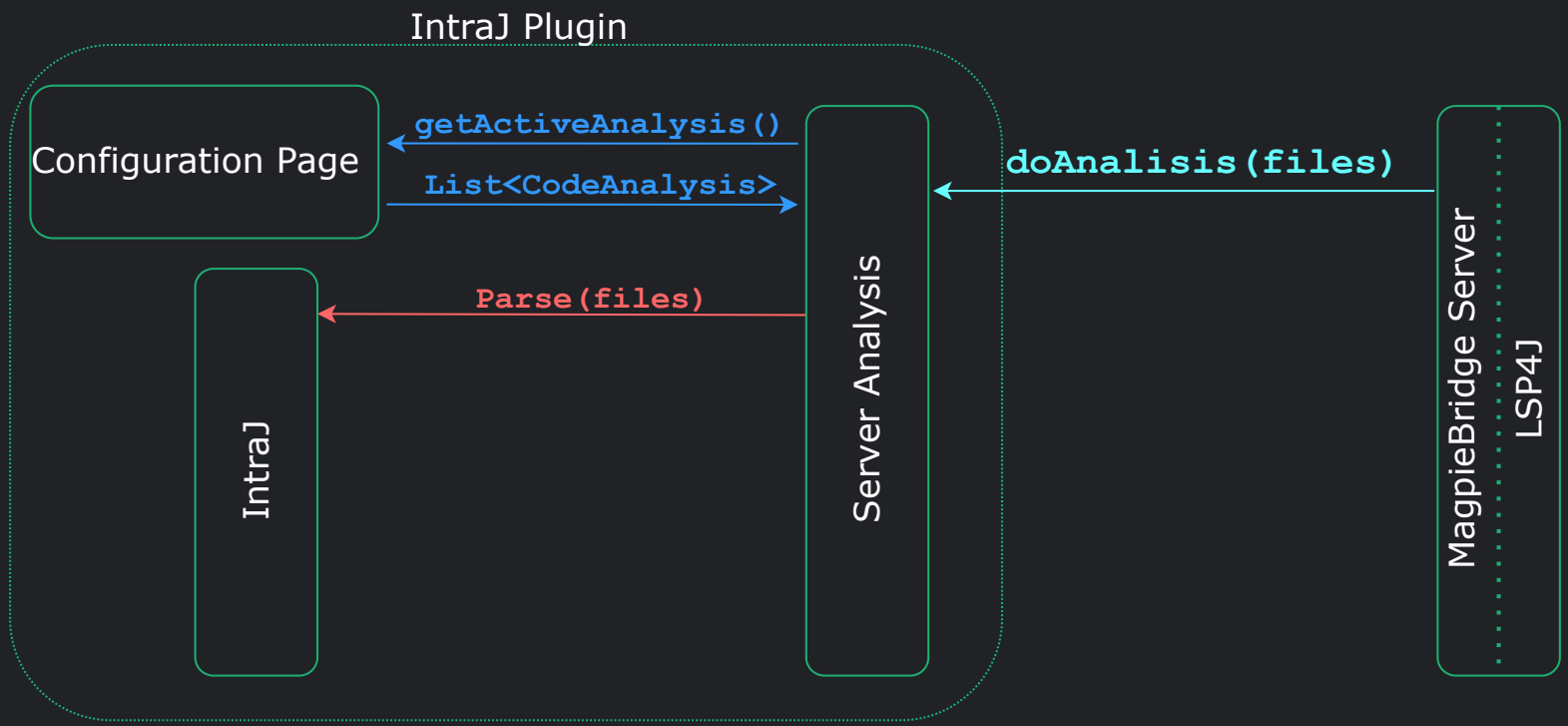
ZOOM-IN


IntraJ Plugin
Configuration Page
getActiveAnalysis()
doAnalisis(files)
IntraJ
Server Analysis
MagpieBridge Server
LSP4J

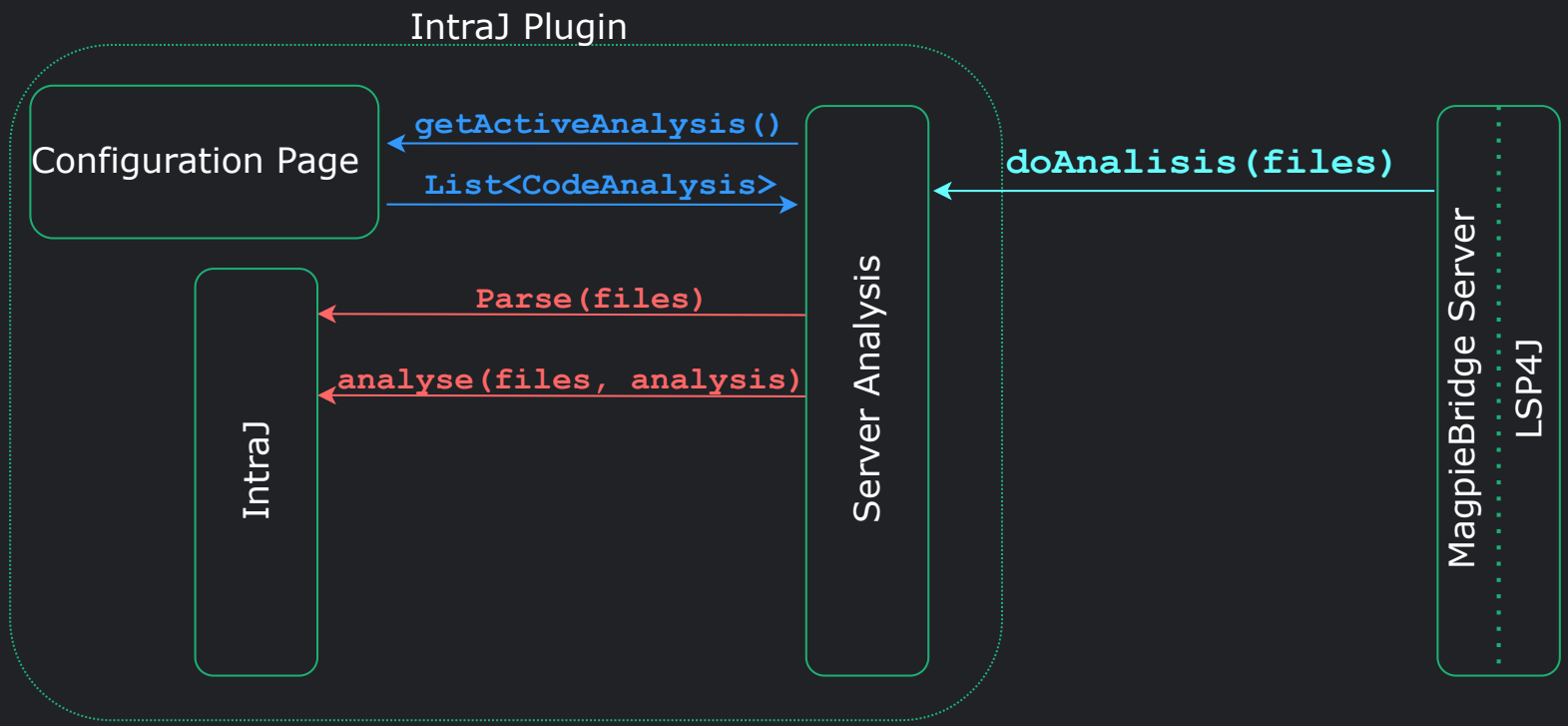
ZOOM-IN


IntraJ Plugin
Configuration Page
getActiveAnalysis()
List<CodeAnalysis>
IntraJ
Server Analysis
doAnalisis(files)
MagpieBridge Server
LSP4J

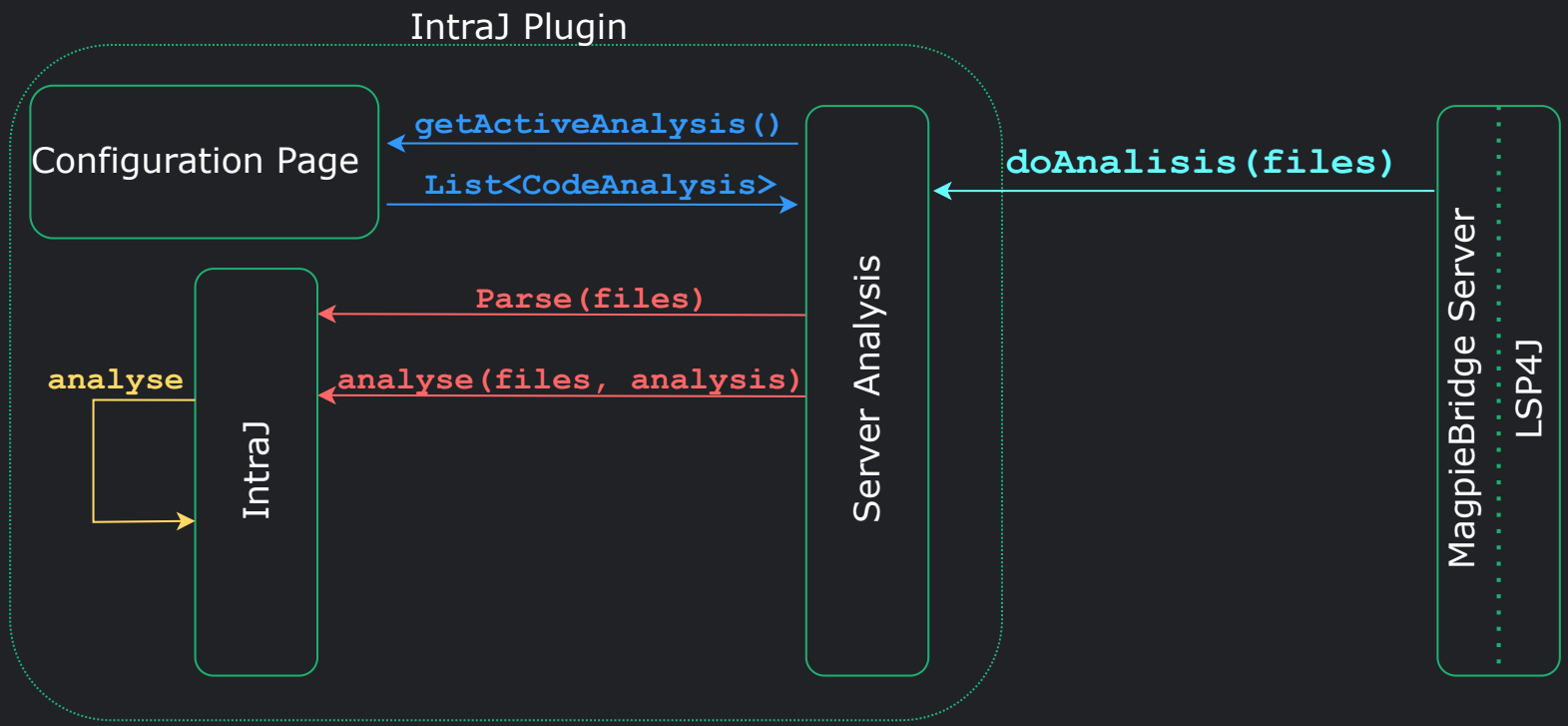
ZOOM-IN


IntraJ Plugin
Configuration Page
getActiveAnalysis()
List<CodeAnalysis>
Parse(files)
IntraJ
Server Analysis
doAnalisis(files)
MagpieBridge Server
LSP4J

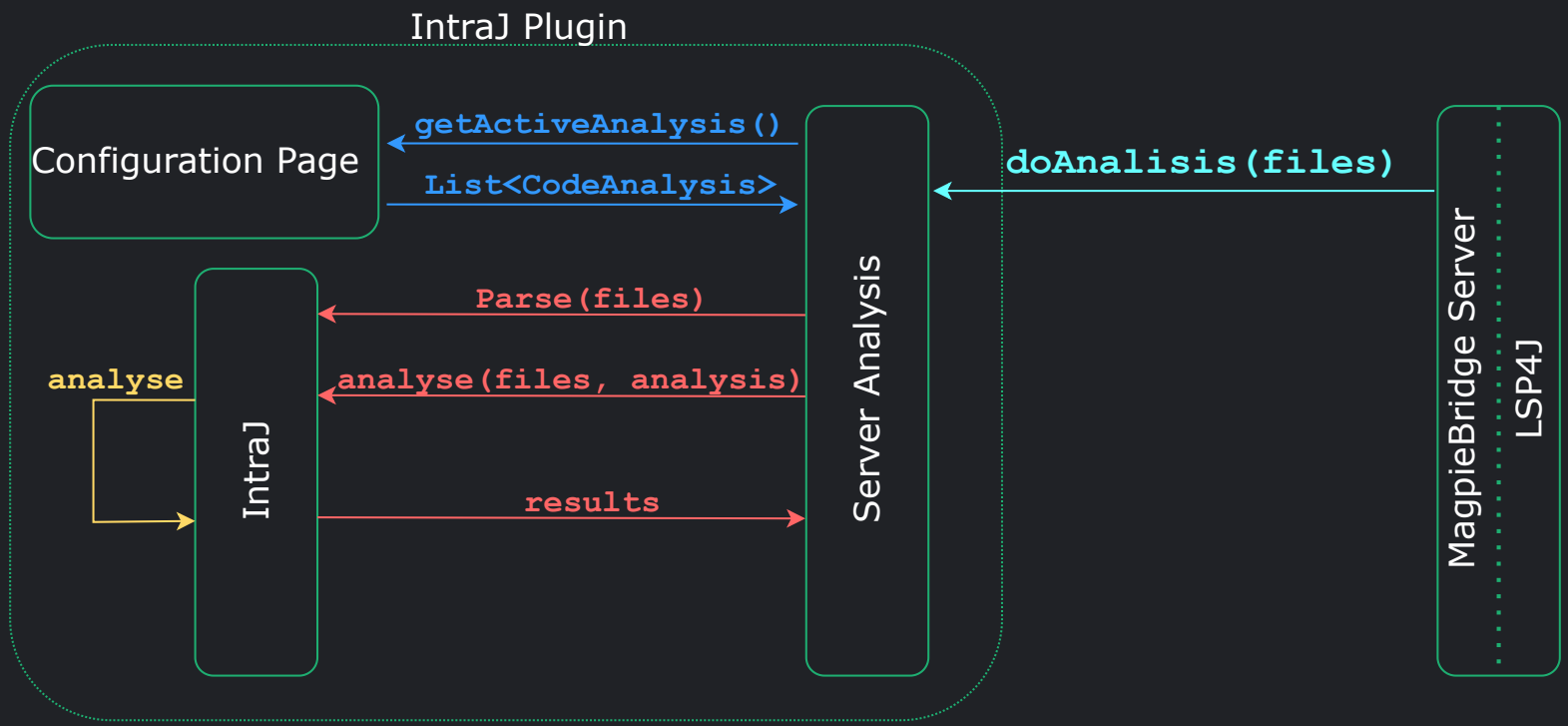
ZOOM-IN


IntraJ Plugin
Configuration Page
getActiveAnalysis()
List<CodeAnalysis>
Parse(files)
analyse(files, analysis)
IntraJ
Server Analysis
doAnalisis(files)
MagpieBridge Server
LSP4J

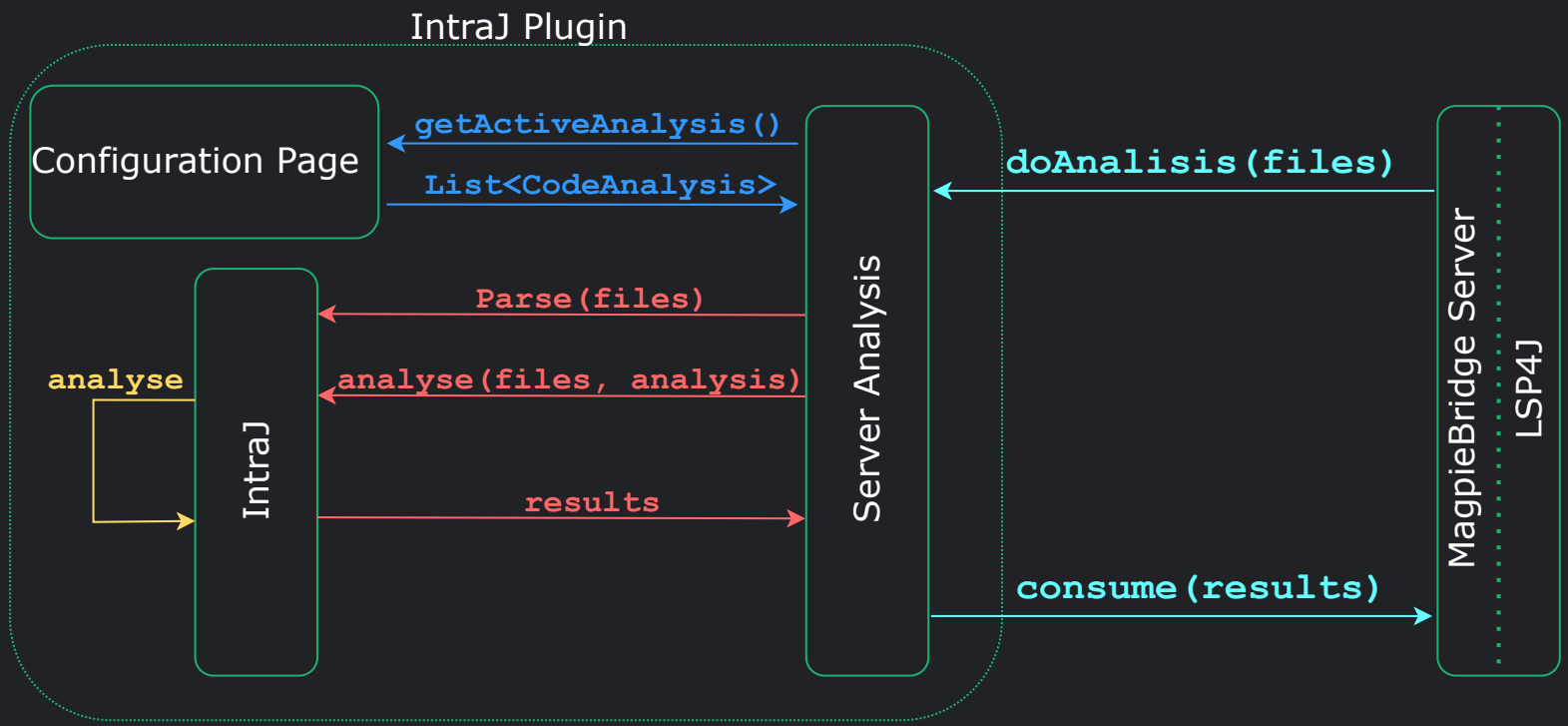
ZOOM-IN


IntraJ Plugin
Configuration Page
getActiveAnalysis()
List<CodeAnalysis>
Server Analysis
doAnalisis(files)
MagpieBridge Server
LSP4J
Parse(files)
analyse
analyse(files, analysis)
IntraJ

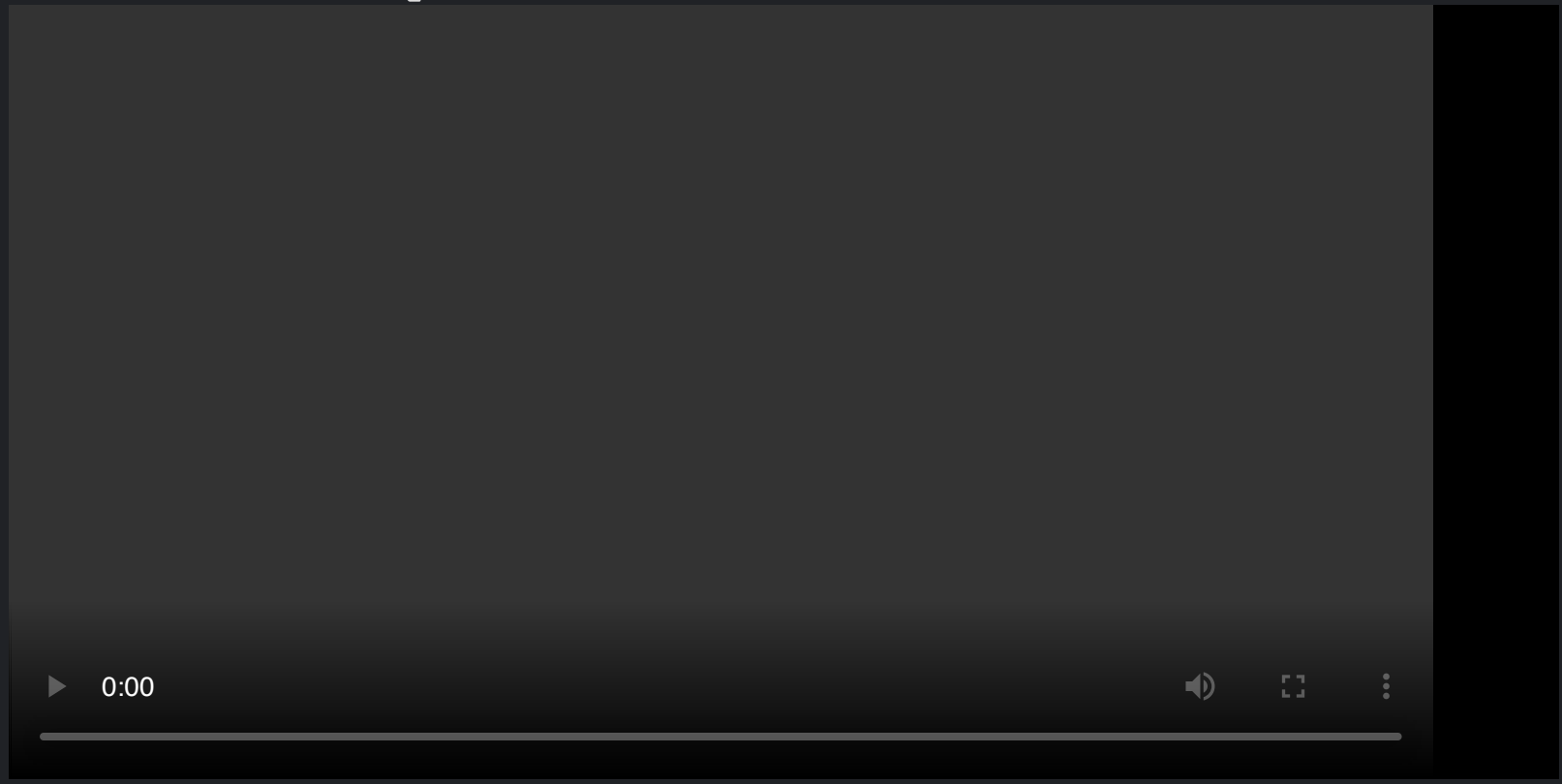
ZOOM-IN


IntraJ Plugin
Configuration Page
getActiveAnalysis()
List<CodeAnalysis>
Server Analysis
doAnalisis(files)
MagpieBridge Server
LSP4J
Parse(files)
analyse
analyse(files, analysis)
IntraJ
results

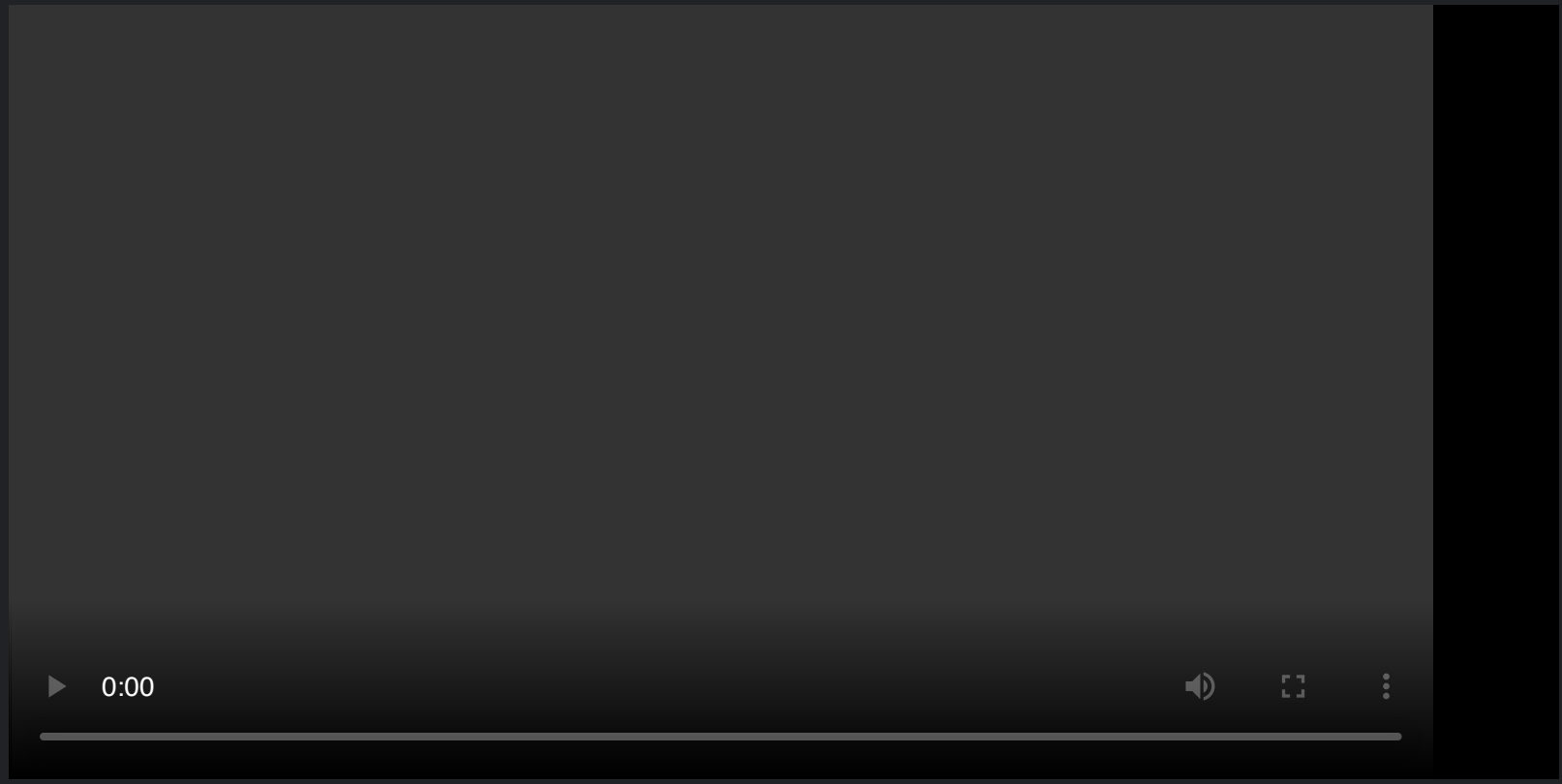
ZOOM-IN


IntraJ Plugin
Configuration Page
getActiveAnalysis()
List<CodeAnalysis>
Server Analysis
doAnalisis(files)
MagpieBridge Server
LSP4J
Parse(files)
analyse
analyse(files, analysis)
IntraJ
results
consume(results)

EXAMPLE: QUICK FIX



EXAMPLE: BUG EXPLANATION



TIP OF THE ICEBERG

OVERALL EXPERIENCE

- Intuitive and easy to use
- Concise specification of the server
- Well documented
- With the scaffolding we provide, adding support for a new analysis is trivial:

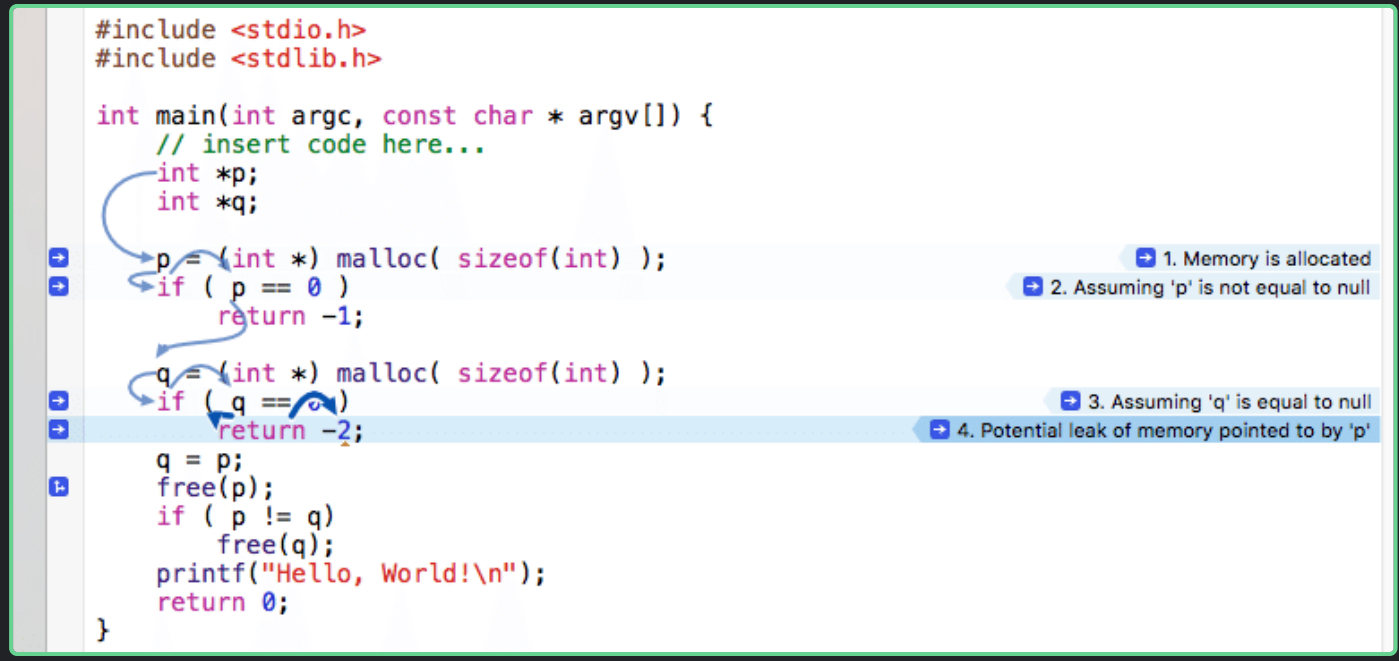
```
1  public class YourAnalysis extends CodeAnalysis {
2    public String getName() { return "YourAnalysis"; }
3    protected Set<Warning> getWarnings(CompilationUnit cu)
4      { return cu.yourAnalysis(); }  //⟵  Property triggered by the analysis
5  }
```

```
1  activeAnalyses.put(new YourAnalysis(), true); //Register the analysis
```

- Plugin V 0.0.1 made by Charlie Mrad (Master Student @ LU)

LOOKING FORWARD FOR ...

Not only warnings



THANK YOU FOR YOUR ATTENTION !

GitHub

Paper

Extension

MOTIVATIONS: SOURCE-LEVEL

```
1  void foo(boolean b){
2    String x = null;
3    if(b) x = "Hello World";
4    x.toString();
5  }
```

Advantages

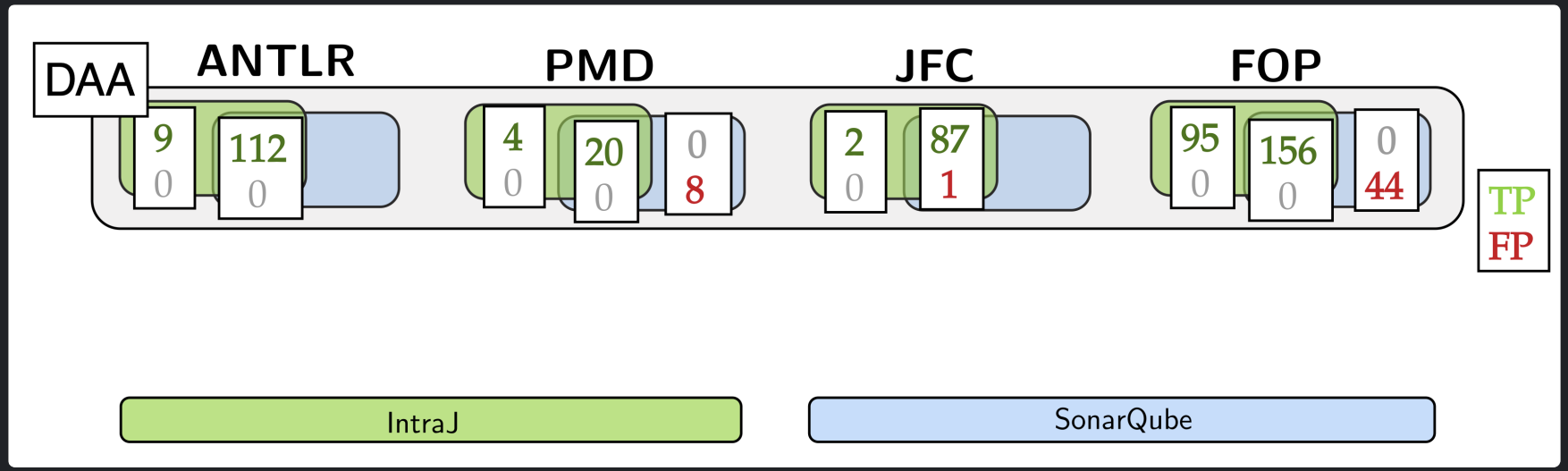
1. Error are directly linked to the source code
2. Works with broken code
3. Easier integration with IDEs

Disadvantages

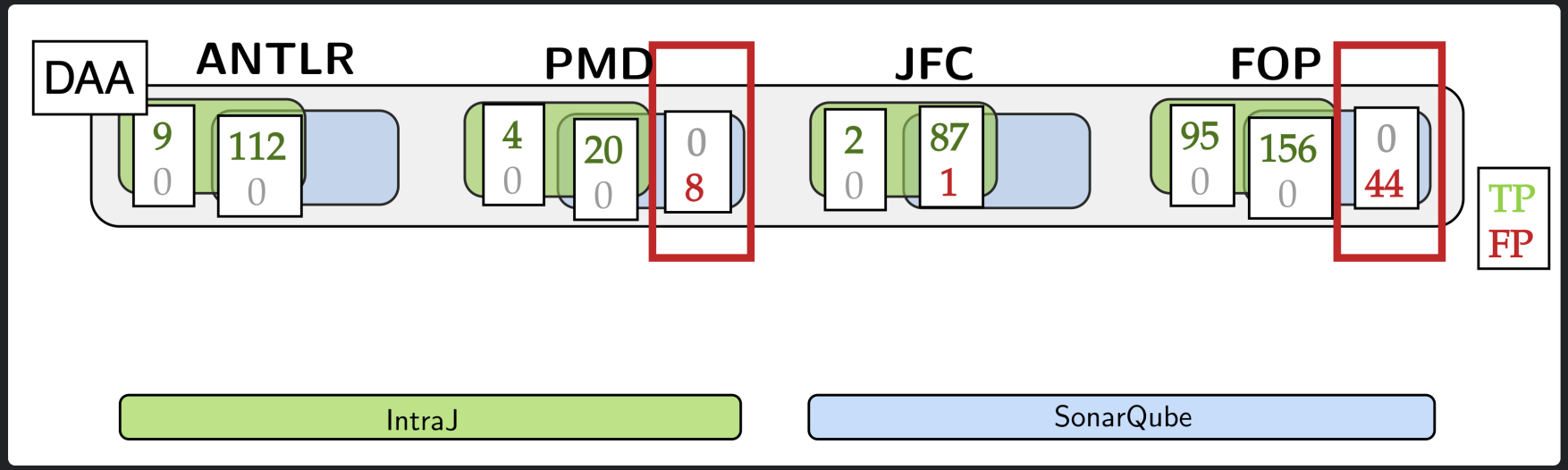
1. Bigger language
2. Source-code contains implicit facts

```
1   void foo(java.lang.boolean);
2     Code:
3        0: aconst_null
4        1: astore_2
5        2: aload_1
6        3: invokevirtual #2
7        6: ifeq          12
8        9: ldc           #3
9       11: astore_2
10      12: aload_2
11      13: invokevirtual #4
12      16: pop
13      17: return
```

PRECISION: NUMBERS

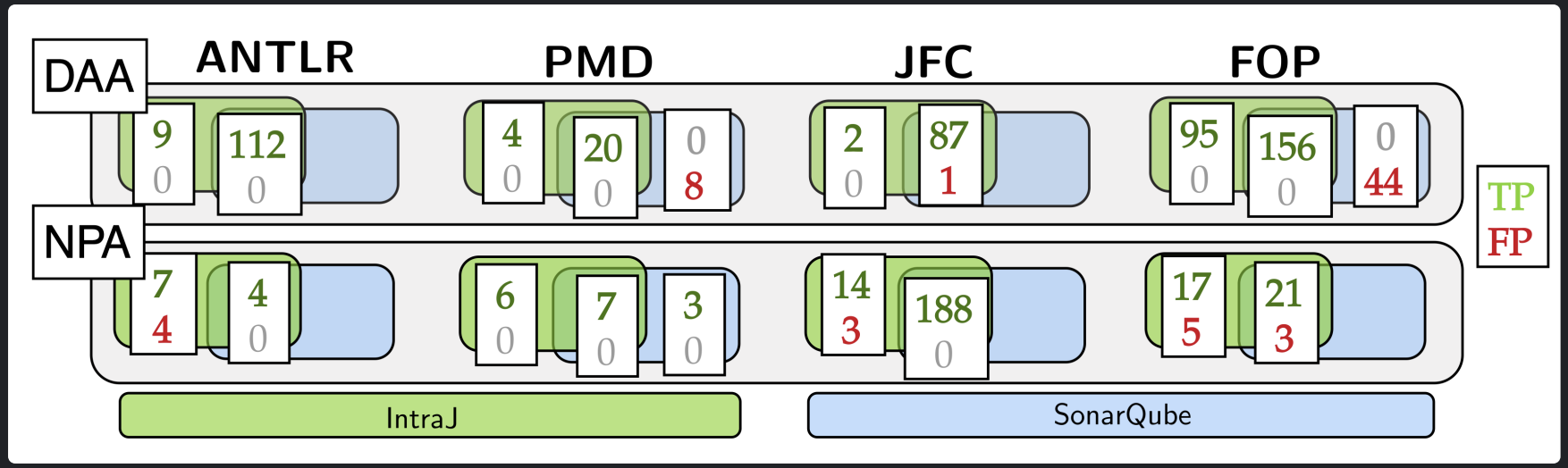


PRECISION: NUMBERS



DeadAssignmentAnalysis: IntraJ detects everything that SonarQube detects

PRECISION: NUMBERS

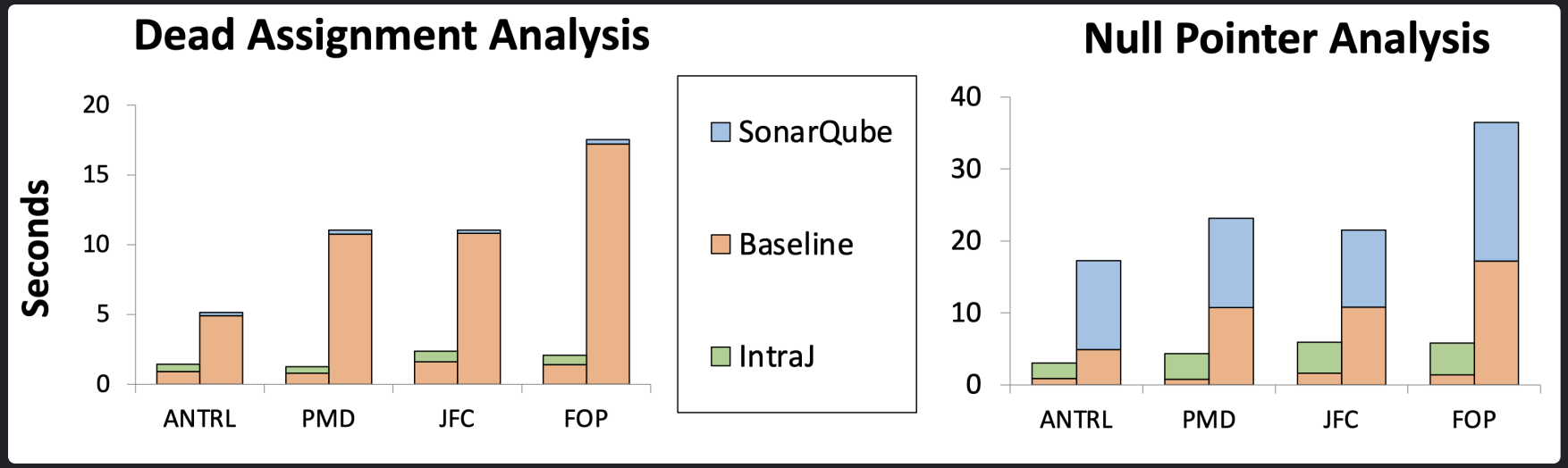


DeadAssignmentAnalysis: IntraJ detects everything that SonarQube detects

NullPointerAnalysis: SonarQube is more precise but IntraJ remains competitive

PERFORMACE

1. No dealy in the previous demo



INTRAJ - A USE CASE