AN INTRODUCTION TO SOOT

Kongjunjun @OS Lab

2008-10-23

AGENDA

- What's Soot
- Soot's IRs
- Soot's Tags
- How to set up Soot
- How to develop an analysis with Soot
- Soot's Resources

AGENDA

- **What's Soot**
- Soot's IRs
- Soot's Tags
- How to install Soot
- How to develop an analysis with Soot
- Soot's Resources

WHAT'S SOOT

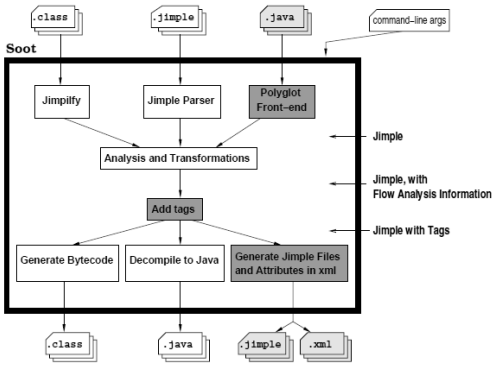
- **Soot** is a Java optimization framework
- a tool for analyzing and transforming Java bytecode
- developed at McGill University, Canada
- a free compiler infrastructure, written in Java(LGPL)
- originally designed to analyze and transform Java bytecode
- has been extended to include decompilation and visualization

SOOT'S APPLICATIONS

- Two general fields
  - compiler research: for a wide variety of applications
  - graduate course projects: used in several compiler courses

- Soot can be used as a standalone tool to optimize or inspect class files
- as well as a framework to develop optimizations or transformations on Java bytecode

SOOT'S OVERVIEW

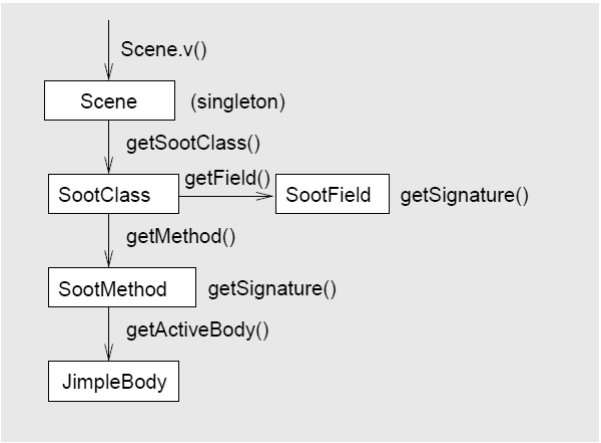


- Note:this slice from http://plg.uwaterloo.ca/~olhotak/cs744/4soot.pdf

SOOT DATA STRUCTURE

- Soot builds data structures to represent:
  - a complete environment (**Scene)** the analysis takes place in
  - classes (**SootClass)**
  - Fields and Methods (**SootField , SootMethod)**
  - bodies of Methods (come in different flavours, corresponding to different IR levels, ie. **JimpleBody)**
- These data structures are implemented using OO techniques

SOOT DATA STRUCTURE(CONT.)



- Note: this slice from: pldi03 tutorial by Laurie Hendren, Patrick Lam, Jennifer Lhot ´ak, Ond ˇ rej Lhot ´ak and Feng Qian

AGENDA

- What's Soot
- **Soot's IRs**
- Soot's Tags
- How to install Soot
- How to develop an analysis with Soot
- Soot's Resources

SOOT'S IRS

- **Baf**: is a compact rep. of Bytecode (stack-based)
- **Jimple**: is Java's simple, typed, 3-addr (stackless) representation
- **Shimple**: is a SSA variation of Jimple
- **Grimp**: is like Jimple, an aggregated version of Jimple suitable for decompilation and code inspection
- **Dava**: structured representation used for Decompiling Java

JIMPLE

- **Jimple** is:
  - principal Soot Intermediate Representation
  - 3-address code in a control-flow graph
  - a typed intermediate representation
  - stackless
- Soot can output the Jimple representation of classes in a textual format (.jimple files) and reread this textual format back into Soot

JIMPLE STMTS

- Core statements:
  - NopStmt
  - DefinitionStmt:
    - IdentityStmt,
    - AssignStmt
- Intraprocedural control-flow:
  - IfStmt
  - GotoStmt
  - TableSwitchStmt
  - LookupSwitchStmt
- Interprocedural control-flow:
  - InvokeStmt
  - ReturnStmt,
  - ReturnVoidStmt
- …

JIMPLE:AN EXAMPLE

```
for(;i<10;i++)
 {
String str;
str=cstr+i;
System.out.println(str);



}
```

```
  label0:
      nop;
      if i < 10 goto label1;

      goto label2;
    label1:
      nop;
      temp$0 = new java.lang.StringBuffer;
      specialinvoke temp$0.<java.lang.StringBuffer: void <init>()>();
      virtualinvoke temp$0.<java.lang.StringBuffer: java.lang.StringBuffer
append(java.lang.Object)>(cstr);
      virtualinvoke temp$0.<java.lang.StringBuffer: java.lang.StringBuffer
append(int)>(i);
      temp$1 = virtualinvoke temp$0.<java.lang.StringBuffer: java.lang.String
toString()>();
      str = temp$1;
      temp$2 = <java.lang.System: java.io.PrintStream out>;
      virtualinvoke temp$2.<java.io.PrintStream: void println(java.lang.String)>(str);
      nop;
      temp$3 = i;
      temp$4 = temp$3 + 1;
      i = temp$4;
      goto label0;

    label2:
      nop;
```

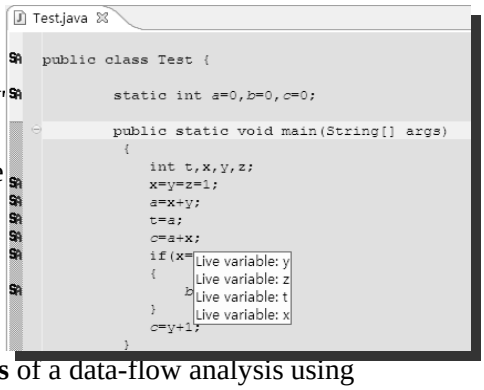

AGENDA

- What's Soot
- Soot's IRs
- **Soot's Tags**
- How to install Soot
- How to develop an analysis with Soot
- Soot's Resources

SOOT'S TAGS

- We often want to attach annotations to code
  - to **convey analysis results to humans**
  - to record profiling information
- Soot supports visualization
- Three visual representations of attribute information:
  - Text displayed in tooltips
  - Color highlighting of chunks of code
  - Pop-up links

SOOT'S TAGS(CON



- Tags are objects that can be

- Soot can display **the results** of a data-flow analysis using three different kinds of tags:
  - **StringTag** – shows a string when you hover over the tag with your mouse
  - **ColorTag** – highlights the tagged Stmt or Value with a color
  - **LinkTag** – lets you jump from the tagged Stmt or Value to the line of another Stmt by clicking on the source Stmt or Value

AGENDA

- What's Soot
- Soot's IRs
- Soot's Tags
- **How to install Soot**
- How to develop an analysis with Soot
- Soot's Resources

HOW TO INSTALL SOOT

- Command-line Soot
  - Refer to this website
    - http://www.sable.mcgill.ca/soot/tutorial/
- Soot Eclipse plugin
  - Follow installation instructions at:
  - http://www.sable.mcgill.ca/soot/eclipse/updates/
- This presentation will show you how to install Soot plugin in Eclipse

SOOT - ECLIPSE PLUGIN

- The Soot - Eclipse Plugin integrates Soot, a Java optimization framework, with Eclipse, allowing the user to:
  - optimize class files automatically and use more advanced optimization options
  - import class files and decompile them using the Dava Decompiler
  - …
- This plugin works with the Eclipse official release 3.1 and should run in any of the 3.x releases

INSTALL **THE SOOT ECLIPSE PLUGIN**

- if your Eclipse version is 3.3.x, then refer to http://www.sable.mcgill.ca/soot/eclipse/updates/..
- If you are using Eclipse 3.4.0, do as follows.
- **Step 1: set up Eclipse 3.4.0 first**
  - Goto website: http://www.eclipse.org/, and download Eclipse 3.4.0 freely
- **Step 2: Start Eclipse**

- **Step 3: left click "Help" within Eclipse then choose "Software Updates…"**
  - **See the next slice**

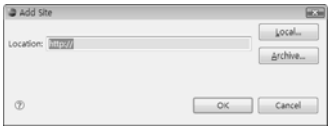
STEP 3 :INSTALL SOOT PLUGIN

- **left click "Help" within Eclipse then choose "Software Updates…"**
  - Now you get the window:

- Select the tab "Available Software"
- Left click the button "Add site…"

STEP 3 :INSTALL SOOT PLUGIN(CONT.)

- Now a small dialog window pops up:



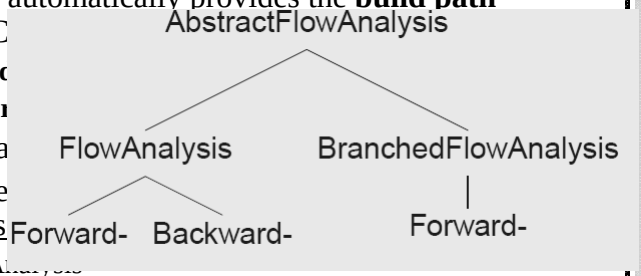
- Use this site for the URL: http://www.sable.mcgill.ca/soot/eclipse/updates/
- Click ok
- Select the row named http://www.sable.mcgill.ca/soot/eclipse/updates/
- Expand the item and select soot
- Click "Install…" on the upper right

AGENDA

- What's Soot
- Soot's IRs
- Soot's Tags
- How to install Soot
- **How to develop an analysis with Soot**
- Soot's Resources

HOW TO DEVELOP AN ANALYSIS WITH SOOT

- The Soot plugin automatically provides the **build path** variable SOOTC
  - **We can Extend**
  - For **example pr**
- Control-flow gra
- There exist three
  of FlowAnalysis
  - ForwardFlowA
  - BackwardsFlowAnalysis
  - ForwardBranchedFlowAnalysis
- What direction you want to use depends entirely on your analysis problem

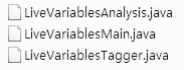


DEVELOPING WITH SOOT IN ECLIPSE

- First create a project for your analysis
  - Choose: File->New->Java Project
  - Give the project a name
  - Click Finish
- Add Java files doing analyzing or transformation
  - You can create, or import some Java files
  - See the following example
- Configure build path
  - See the following example
- Use soot plugin to execute your analysis
  - See the following example

AN EXAMPLE: LIVENESS

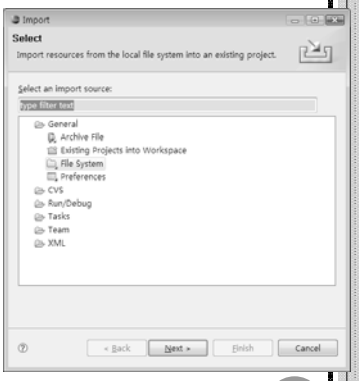
- Note: the liveness analysis codes are from another ACT course in Canada



- **Step 1: create a project within Eclipse for testing**
- **Step 2: create another project for your analysis code**
- **Step 3: do variables liveness analysis with custom codes in Eclipse.**

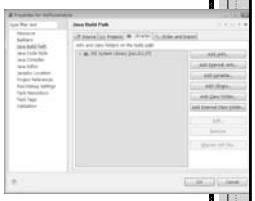
LIVENESS: STEP2

- Suppose you have created an empty project named liveness for your analysis
- Do the following step by step
- Import the source files
  - Right click the project icon
  - Choose import
  - Now you get to this point:
  - Choose file system then click Next
  - Browse to get your target java files
  - Select your files for analyzing
  - Click Finish



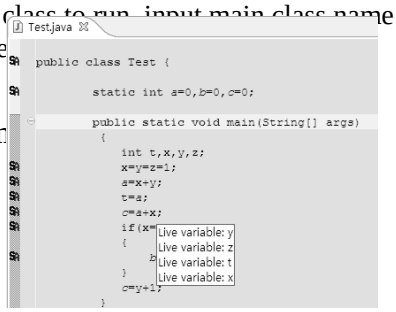
LIVENESS: STEP2(CONT.)

- Because your codes may depends on some packages from soot, you must add the soot path into the current build path
- **Configure the build path**
  - Right click the project icon
  - Choose: Build Path->Configure Build Path..
  - Select the Libraries tab
  - Now what's on the screen is such a window:
  - Click Add Variables..
  - Select SOOTCLASSES then click ok
  - Click ok



LIVENESS: STEP3

- **do variables liveness analysis with custom codes in Eclipse**
  - Right click the test file in your first project
  - Choose: Soot->Process Source File->Run soot…
  - Select “Soot Main Class”
  - Specify main class to run, input main class name and project name(eg. Live
  - Click run
- Wait for a mom



AGENDA

- What’s Soot
- Soot’s IRs
- Soot’s Tags
- How to install Soot
- How to develop an analysis with Soot
- **Soot’s Resources**

SOOT'S RESOURCES

- **Soot's home**:
  - (1)McGill University,http://www.sable.mcgill.ca/soot/
  - (2)Another wonderful site http://www.brics.dk/SootGuide/
- **download soot**:
  - http://www.sable.mcgill.ca/soot/soot_download.html
- **soot tutorial:**
  - http://www.sable.mcgill.ca/soot/tutorial/index.html
- **Soot Eclipse plugin**:
  - General intro
  - http://www.sable.mcgill.ca/soot/eclipse/index.html
  - Installation guide
  - http://www.sable.mcgill.ca/soot/eclipse/updates/
  - Developing With Soot in Eclipse
  - http://www.sable.mcgill.ca/soot/soot_in_eclipse_howto.html

THE END

Thanks

kongjunjun@os lab

2008-10-23