

Learning Graph-based Heuristics for Pointer Analysis without Handcrafting Application-Specific Features

1 Requirement

- A 64-bit Ubuntu system
- A Java 8 distribution
- A Python 2.x interpreter

Please set your `JAVA_HOME` environment variable to point to your Java installation directory.

1.1 Installing Datalog Engine

To run our pointer analyzer DOOP, you need to install a LogicBlox engine for interpreting the Datalog rules used in DOOP. The download link and installation instructions of PA-Datalog can be found on this link (We recommend .deb package installation).

2 Getting Started Guide

2.1 Verifying Installation (Basic Testing)

Move to the directory “`Ctx_Sensitivity/`”. Then, run the following command, which analyzes a program `luindex` with `GRAPHICK`:

```
$ ./run.py graphick luindex
```

If the artifact is successfully installed, you will see the following results:

```
Running graph_ci pointer analysis for luindex ...
...
Pointer analysis START
analysis time: 23.49s
Pointer analysis FINISH
...
loading graph heuristic ...
elapsed time: 14.46s
...
Running 2obj-Graphick pointer analysis for luindex ...
...
Pointer analysis START
analysis time: 34.94s
Pointer analysis FINISH
...
#may-fail casts          297
#poly calls              682
#call edges              29,045
```

The above results illustrate that the pre-analysis(e.g., `graph_ci`) and extracting an abstraction from the pre-analysis results take 23.49 and 14.46 seconds, respectively. The sum of these two costs corresponds to the analysis cost in the parentheses in Table 1 and 2. The main analysis (e.g., `2obj-Graphick`) takes 34.94 seconds. The results also show the results for each client (`#may-fail casts`, `#poly-call sites`, and `#call-graph-edges` metric). Note that it may report the different analysis time from the paper due to the differences in the experiment environments. In our paper, all the experiments are conducted on a machine with 64GB of RAM.

3 Step-by-Step Instructions

Following the instructions below reproduces Table 1, 2, 3, and 4. The command for running pointer analysis is as follows:

```
$ ./run.py <analysis> <pgm>
```

If your current directory is “Ctx_Sensitivity/”, <analysis> can be one of the following analyses:

```
graphick, scaler, zipper, data, 2objh, insens.
```

If you are in “Heap_Abstraction”, <analysis> can be:

```
graphick, mahjong, alloc_based, type_based.
```

<pgm> can be:

```
luindex, lusearch, antlr, pmd, fop, chart, bloat, pmdm, eclipse  
xalan, JPC, checkstyle, findbugs, soot, jython, briss, jedit
```

For example, if you want to analyze antlr with GRAPHICK, type:

```
$ ./run.py graphick antlr
```

4 Run

4.1 How to Transform Programs into Graphs

Move to the directory “Ctx_Sensitivity/” or “Heap_Abstraction/”. Then, type the following command:

```
$ ./run.py getGraph <pgm>
```

It will produce <pgm>-Nodes.facts and <pgm>-Edges.facts, which contains the nodes and edges in the graph, respectively.

4.2 How to Run the Analyzer with an Abstraction (Classified Nodes)

- If you are in “Heap_Abstraction/”, modify “CanHeap.facts” to contain nodes that need to be analyzed precisely, where the set of nodes in “CanHeap.facts” is a subset of that in <pgm>-Nodes.facts. Then, type the following:

```
$ ./run.py heuristic <pgm>
```

Please run the above command after you run \$./run.py getGraph <pgm>.

- If you are in “Ctx_Sensitivity/”, modify “CanHeap2obj.facts”, “CanHeap2type.facts”, and “CanHeap1type.facts” to contains nodes that need to be analyzed under 2-object-sensitivity (very precise), 2-type-sensitivity (precisely), 1-type-sensitivity(coarsely), respectively. Then, type the following:

```
$ ./run.py heuristic <pgm>
```

4.3 Oracles

In “oracles/” folder, you can find oracles for small programs (e.g., luindex, lusearch, antlr, pmd) that we used in our training. Using the oracles as clusters produce a cost-effective analysis. For example, type the following command

```
$ ./run.py heuristic antlr
```

after you change “CanHeap2obj.facts”, “CanHeap2type.facts”, and “CanHeap1type.facts” with “oracles/antlr-Minimal-2obj.facts”, “oracles/antlr-Minimal-2type.facts”, and “oracles/antlr-Minimal-1type.facts”, respectively. It will analyze antlr cost-effectively.