PROGEX (Program Graph Extractor)

PROGEX is a cross-platform tool for extracting well-known graphical program representations (such as CFGs, PDGs, ASTs, etc.) from software source code. PROGEX is written in Java, based on the ANTLR parser generator.

PROGEX reads program source code files as input, and is able to generate various graphical program representations and export them into well-known file formats for graphs; such as DOT and JSON.

The structure of this repository is a Maven project. If you are interested in contributing to this project, you can easily get up and running using your favorite Java IDE and Maven.

This is an internal project by members of our lab from Amirkabir University of Technology, as a sub-project of the PhD thesis of Seyed Mohammad Ghaffarian.

Why We Developed PROGEX?

There are key aspects about PROGEX which makes it different and relevant in the presence of other well-known program analysis frameworks (such as SOOT, Chord, Spoon, etc.):

- PROGEX is designed to operate at the source code level (or more precisely, at Abstract Syntax Tree (AST) level); not any other intermediate code or machine code level. To the best of our knowledge, all the above-mentioned frameworks operate at byte-code level. To explain the benefits of analysis at this level, we provide the following quote:
- Control-flow and data-flow analyses are key elements in many static analyses, and useful for a variety of purposes, e.g. code optimization, refactoring, enforcing coding conventions, bug detection, and metrics. Often, such analyses are carried out on a normalized intermediate code representation, rather than on the abstract syntax tree (AST). This simplifies the computations by not having to deal with the full source language. However, doing these analyses directly at the AST level can be beneficial, since the high-level abstractions are not compiled away during the translation to intermediate code. This is particularly important for tools that are integrated in interactive development environments, such as refactoring tools and tools supporting bug detection and coding convention violations.

- -- Extensible Intraprocedural Flow Analysis at the Abstract Syntax Tree Level
- PROGEX is not a general-purpose program analysis, transformation, or rewriting framework. In contrast, it is only focused on the task of extracting well-known graphical program representations, and the ability to export them in well-known formats for further analysis by external programs. While all the above-mentioned frameworks are able to extract well-known graphical program representations, they are only internally usable, and none provide convenient mechanisms for exporting these graphs and making them available to external programs.
- While the current state of PROGEX only supports the Java programming language, there is no limitation to support other programming languages. Any programming language that can be parsed using the ANTLR parser generator, can be supported by PROGEX. This virtually means all programming languages can be supported!

After doing many searches online, we were unable to find any suitable tool with the above-mentioned properties. Since such a tool was a requirement for our research projects, hence we developed PROGEX.

Usage Guide

```
USAGE:

   java -jar PROGEX.jar [-OPTIONS...] /path/to/program/src


OPTIONS:

   -help       Print this help message
   -outdir     Specify path of output directory
   -format     Specify output format; either 'JSON' or 'DOT'
   -lang       Specify language of program source codes

   -ast        Perform AST (Abstract Syntax Tree) analysis
   -cfg        Perfomt CFG (Control Flow Graph) analysis
   -icfg       Perform ICFG (Interprocedural CFG) analysis
   -info       Analyze and extract detailed information about program source
code
```

```
   -pdg       Perform PDG (Program Dependence Graph) analysis

   -debug     Enable more detailed logs (only for debugging)
   -timetags  Enable time-tags and labels for logs (only for debugging)


DEFAULTS:

   - If not specified, the default output directory is the current working
directory.
   - If not specified, the default output format is DOT.
   - If not specified, the default language is Java.
   - There is no default value for analysis type.
   - There is no default value for input directory path.


EXAMPLES:

   java -jar PROGEX.jar -cfg -lang java -format dot  /home/user/project/src

      This example will extract the CFG of all Java source files in the given
path and
      will export all extracted graphs as DOT files in the current working
directory.

   java -jar PROGEX.jar -outdir D:\outputs -format json -pdg  C:\Project\src

      This example will extract the PDGs of all Java source files in the given
path and
      will export all extracted graphs as JSON files in the given output
directory.


NOTES:

   - The important pre-assumption for analyzing any source code is that the
     program is valid according to the grammar of that language. Analyzing
```

```
    invalid programs has undefined results; most probably the program will
    crash!

  - Analyzing large programs requires high volumes of system memory, so
    it is necessary to increase the maximum available memory for PROGEX.

    In the example below, the -Xmx option of the JVM is used to provide PROGEX
    with 5 giga-bytes of system memory; which is required for the PDG analysis
    of very large programs (i.e. about one million LoC). Needless to say, this
    is possible on a computer with at least 8 giga-bytes of RAM:

        java -Xmx5G -jar PROGEX.jar -pdg ...
```

Installation and Requirements

PROGEX is a fully portable tool and requires no installation. Just download the latest stable release from our releases page.

Installing a Java Runtime Environment (JRE, version 8 or newer) is the only requirement for running PROGEX. To acquire the latest JRE version for your platform, visit java.com

Visualizing Output Graphs

One use of the exported graphs is to visually analyze the resulting graphs for program comprehension, and other analysis purposes. For this purpose there are several options:

1. xdot

PROGEX can export the extracted graphs into DOT format. This format can be visualized using the xdot program. To install `xdot` on Ubuntu, use the following command:

```
sudo apt install xdot
```

And visualize the resulting graph as below:

```
xdot graph.dot
```

An alternative way is to create an image file. This can be done as follows:

```
dot -Tpng -o graph.png graph.dot
```

2. PROGVIZ

While `xdot` is a fine program and fulfills many basic requirements, it has some limitations. PROGVIZ is a related project by the developers of PROGEX, which aims to overcome some of xdot's limitations and provide an improved modern graph visualization tool for program analysis applications. This tool is still under active development and supports both the DOT and JSON output formats of PROGEX. For more information, refer to its dedicated GitHub repository:

https://github.com/ghaffarian/progviz/