Using concepts of text based plagiarism detection in source code plagiarism analysis

Michal Ďuračík, Emil Kršák, Patrik Hrkút

University of Žilina, Slovakia

Motivation

- Increasing number of plagiarism in students' assignments
- Lack of complex tools
- Usage not only in plagiarism detection
  - Improving code quality (bugs), finding clones, refactoring
- Similarities between text and source code plagiarism detection

Text documents processing

- Conversion of the documents into plain text
- Tokenization
- Removing stop-words
- Stemming and lemmatization
- Representation of the document
- Searching for similarities
- Reporting

Differences between text and source code processing

- Source code preprocessing
  - Purification (eliminating generated code)
- Tokenization
  - Revealing a structure of code
- No lemmatization, no stemming
  - The grammar is not as complicated as in natural language
- Different methods of representations
  - Text and token based
  - Abstract syntax tree

String and token based detection

- String based
  - Treating source code as strings (substrings)
  - Task: find matching substrings (lines, code snippets)
- Token based
  - Improved version of string based detection
  - “Words” in source codes are converted to tokens
  - Task: find the matching token sequence

Available tools (string & token based)

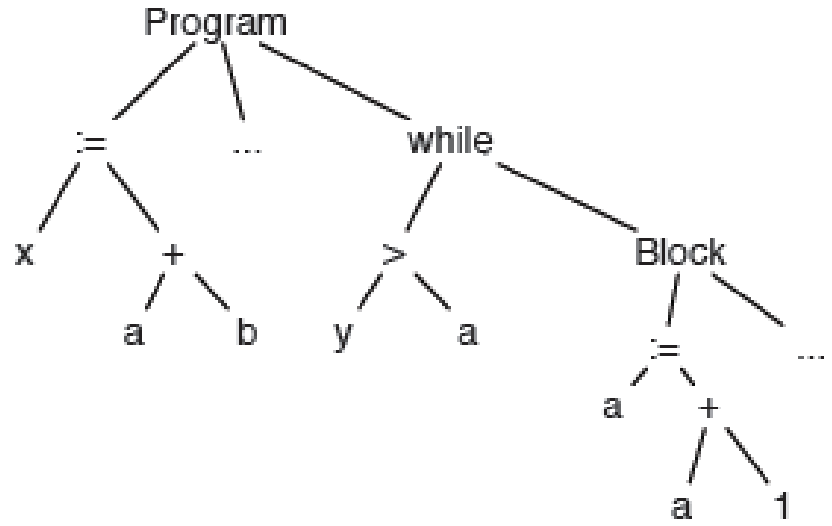
- JPlag
  - Ignores white spaces, comments, identifier names
  - Uses *Greedy String Tiling* method for the comparison and generating the similarity value
- MOSS
  - Document-based textual similarity in source code
  - Uses method of *k-grams* to detect similarity
- CCFinder
  - Token based code clone detector

Abstract syntax tree (AST)

- Source code is not a text written in any natural language
  - Source code has a strict structure
- AST is a tree structure (graph theory)
  - Vertices (nodes) connected with edges (lines)
- The tree represents structure of source code
  - Without irrelevant parts
- What is similar and what isn't?
  - Changing name of identifiers, change order of lines, …
  - Example if (a>b) … vs. if (b>=a) …

Abstract syntax tree - example

```
x := a + b;
y := a * b;
while (y > a) {
    a := a + 1;
    x := a + b
}
```



Source: https://vinaytech.wordpress.com/tag/abstract-syntax-tree/

Available tools (AST based)

- CloneDR
  - Tool for finding redundancy in software project
  - Goal: eliminate duplicate parts of code
  - Not free (commercial software)
- DECKARD
  - Free command line tool for Linux
  - Based on identifying similar subtrees
  - Computes characteristic vector of subtrees for comparison

Document similarity measurement using AST

- Clustering of similar subtrees
  - Comparing all combination of all subtrees is time and resource consuming
- Find a proper metrics
  - Hash/fingerprint of subtrees
  - Subtree characteristics vector distance measurement

How to build source code antiplagiarism system?

- System will consist of following parts:
  - Input data processing
  - Indexing
  - Similarities detection & calculation
  - Reporting

Current progress

- We are at the beginning:
  - Research of tools for building of abstract syntax trees
  - Will we need our own algorithm?
  - Choosing an efficient data structure for AST representation
  - Visualizing of AST

Thank you for your attention

Patrik.Hrkut@fri.uniza.sk