USAGE METHODS OF COMPARISONS FOR CODE ANALYSE

1 Formulation of problem

Goal of this work is to search out the most optimal ways to compare different pieces of code. So far there are two techniques of code comparison: a normal text comparison and graph compare. Two, similar but not same, pieces of source code must be selected and researched to detect any distinction. These steps can be approached by following:

1. Research on source code using existing methods to compare:
   (a) normal text compare
   (b) spanning trees transformed from control flow graphs
2. Research on source code using existing methods to compare:
   (a) abstract trees
   (b) spanning trees transformed from control flow graphs
3. Research on byte code using existing methods to compare:
   (a) normal text compare
   (b) spanning trees transformed from control flow graphs

Play around with patterns of code changing variables, names, sequences of commands and use simple ”text to text” compare. In parallel create a control flow graphs and using implemented algorithms TopDown and BottomUp figure out the difference.Compare these both differences and declare received results. The results can be:

1. same difference
2. similar difference
3. full difference

In case similar or full difference a combination or new implementation of algorithms can be further developed.

ABSTRACT

This paper presents graph algorithms to solve isomorphism problem under directed non-labeled non-weighted graphs. Combinatorial algorithms are presented in this paper for testing isomorphism of unordered trees, finding one of all isomorphisms of a tree as a subtree of another tree. Depending from type of tree, can be ordered/unordered, labeled/non-labeled, the first step would be reasonable to solve the isomorph problem using clear graph theory. Thus, the following algorithms were implemented under unordered and non-labeled trees as a requisites for further development. The main concept of the paper is to define and to visualize isomorphic trees for tree comparison.

KEY WORDS
control flow graph comparison, graph isomorphism, top down, bottom up.

2 Introduction

The Control Flow Graph Comparison is important in several fields of application such as efficient testing programs, merging between two versions of software, testing compiler optimization and code instrumentation tools.

To compare the structure of two control flow graphs it is essential to know if one graph contains the structure of the second graph. A computational task in which two graphs G_1 and G_2 are given as input, and one must determine whether G_1 contains a subgraph that has the same structure as the graph G_2, is called the subgraph isomorphism problem. The subgraph isomorphism problem is a fundamental problem in graph theory and it is known to be NP-complete [1]. Fortunately polynomial-time algorithms for the subgraph isomorphism problem are known for trees [2], two-connected outerplanar graphs [3], and two-connected series-parallel graphs [4].

In this paper we present the implementation of algorithms for comparison two graphs built from code in Dr. Garbage project. Combination of the algorithms Top-Down and Bottom-Up is suitable instrument to define isomorphism of trees constructed from graphs.As a first step, the clear graph theory is being considered, thereby the content of nodes and order of edges are not important to solve graph isomorphism problem. Therefore the unordered tree isomorphism says following: two unordered trees are isomorphic if there is a bijective correspondence between their node sets which preserves and reflects the structure of the trees - that ism such that the node corresponding to the root of one tree is the root of other tree, and a node v_1 is the parent of a node v_2 if and only if the node corresponding to v_1 is the parent of node corresponding to v_2 in the other tree [10]. In other words when nodes in two unordered trees are permuted and have different structure, but the connections among the nodes are same, it means the trees are isomorph.

All algorithms presented in this paper have been implemented in the context of the Dr. Garbage tool suite [11] and we present some experimental results and statistics of our implementation in section 4.

3 Graph Comparison Algorithms

TODO: briefly explain the used subtree isomorphism algorithms.

The input for the comparison algorithms are control flow graphs (CFG) of a method. The CFG is defined as a tuple $G = (V, A)$, where V is a nonempty set of *vertices* representing instructions of a method, A is a (possibly empty) set of *arcs* (or edges) representing transitions between the instructions. Formally, A is the finite set of ordered pairs of vertices (a, b), where $a, b \in V$.

TODO: briefly explain the used comparison algorithms from the previous paper.

TODO: here comes the new content.

4 Experimental Evaluations

TODO: review this section

To evaluate our algorithms, we selected class files ...

The test case ??? is particularly interesting because the result of the algorithm ??? is worse than the result of the algorithm ???.

The most important conclusion is that the algorithm...

5 Conclusion

TODO: review and extend this section

In this paper we presented algorithms for comparing ...

Experimental results show that proposed algorithms ...

6 Acknowledgements

The authors would like to thank the Dr. Garbage Project Community for supporting the implementation of the proposed algorithms and tools [**?**].

References

[1] Julian R Ullmann, An algorithm for subgraph isomorphism, *Journal of the ACM*, 23(1), 1976, 31-42.

[2] D. W. Matula. An algorithm for subtree identification, *SIAM Review*, 10, 1968, 273-274.

[3] A. Lingas, Subgraph isomorphism for biconnected outerplanar graphs in cubic time, *Theoretical Computer Science*, 1989, 63:295-302.

[4] A. Lingas and M. M. Sys lo, A polynomial-time algorithm for subgraph isomorphism of two-connected series-parallel graphs, *In Proc. 15th Int. Colloq. Automata, Languages and Programming*, LNCS 317, Springer-Verlag, 1988, 394-409.

[5] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, (Addison-Wesley, 1974).

[6] W. Yang, Identifying syntactic differences between two programs, *Software - Practice and Experience*, 21(7), 1991, 739755.

[7] S. M. Selkow, The tree-to-tree editing problem, *Information Processing Letters*, 6(6), 1977, 184186.

[8] G. Valiente, Simple and efficient subtree isomorphism, *Technical Report LSI-00-72-R*, Technical University of Catalonia, Department of Software, 2000.

[9] G. Valiente.,Simple and efficient tree comparison, *Technical Report LSI-01-1-R*, Technical University of Catalonia, Department of Software, 2001.

[10] Gabriel Valiente, *Algorithms on Trees and Graphs*, (Berlin: Springer-Verlag, 2002).

[11] Sergej Alekseev, Victor Dhanraj, Sebastian Reschke, and Peter Palaga, Tools for Control Flow Analysis of Java Code, *Proceedings of the 16th IASTED International Conference on Software Engineering and Applications*, 2012.

[12] H. Ehrig, Introduction to graph grammars with applications to semantic networks, *Computers and Mathematics with Applications*, 1992, Vol. 23, pp. 557572.

[13] X. Jiang, A. Mnger and H. Bunke, Synthesis of representative graphical symbols by generalized median graph computation, *Proc. of 3rd IAPR Workshop on Graphics Recognition*, Jaipur, 1999.

[14] E. K. Wong, Model matching in robot vision by subgraph isomorphism, *Journal Pattern Recognition*, 1992, Vol. 25, pp. 287304.

[15] T. Sager, Coogle A Code Google Eclipse Plug-in for Detecting Similar Java Classes, *Department of Informatics, University of Zurich*, 2006

[16] J. Laski, W. Szermer, Identification of Program Modifications and its Applications to Software Maintenance, *IEEE Conference on Software Maintenance*, 1992, pages 282-290, Nov.

[17] T. Apiwattanapong, A. Orso, M. J. Harrold, A Differencing Algorithm for Object-Oriented Programs, *IEEE International Conf. on Automated Software Engineering*, 2004, pages 2-13.

[18] J. Ferrante, K. J. Ottenstein, and J. D. Warren, The program dependence graph and its use in optimization, *ACM Transactions on Programming Languages and Systems*, July 1987, 9(3):319?349.

[19] G. Stiege, An Algorithm for Finding the Connectivity Structure of Undirected Graphs, *Technical Report Institut fr Betriebssysteme und Rechnerverbund*, May 1997

[20] G. Stiege, *Graphen und Graphalgorithmen*, (Shaker; *Auflage: 1*, 2006).

[21] G. Valiente, C. Martinez, An algorithm for graph pattern-matching, *In Proc. 4th South American Workshop on String Processing*, 1997, volume 8 of Int. Informatics Series