Git ADV: Improving Java & Python Source Code Merging using Abstract Data Structures

Nirmal Chaudhari, Madhur Jain, Sebastien Mosser PhD P.Eng

Department of Computing & Software, McMaster University, Hamilton, Canada.

Introduction

- With the increased use of **Git** in Software Development, developers often find themselves having to resolve numerous merge conflicts when pushing their changes to their shared repository [1]
- Modern merge tools are either unable to scale well for large number of differences, or cannot be used for multiple programming languages
- Without a better solution, source code merging is becoming an increasingly tedious and time-consuming process

Mechanics of Merges

- Modern merge algorithms perform **three-way merges** by differencing between the common ancestor (base) and the latest commits of the merging branches [1]

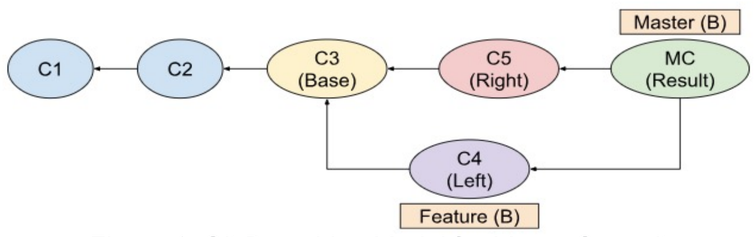

Figure 1: Git Branching Model for Merge Commits

- The differences between the three versions are analyzed using a structured or unstructured approach
  - **Unstructured Merge** tools like Git's 3-way merge rely on textual differencing using the LCS (Longest Common Subsequence) algorithm [1]
    - However, these algorithms are unable to scale well with significant changes between versions, producing many merge conflicts
  - **Structured Merge** tools like **jDime** and **Spork** rely on **AST** (abstract syntax trees) to provide a representation of the source code whose nodes can be directly compared
    - These tools are more precise than unstructured tools, but they are often slower and poorly retain the original code structure [1] These tools are also more language dependant, which makes it difficult generalize to all languages

Objectives

- **Evaluate** the accuracy of current tools using case studies extracted from the **Awesome Java & Python** Lists [2,3]
- **Develop** an Abstract Data Structure for merge conflict resolution that can be generalized to multiple languages (like Git) and maintain the accuracy of existing Structured merge tools (jDime & Spork)
- **Compare** the accuracy of developed tool against existing ones using the developer's desired version as the benchmark

Comparison Methodology

- Measure the difference between the developer's desired version and the results generated by each tool using **Gumtree** [4]
- Five measurements for comparison:
  - **Deletions:** Code present in desired, but not result
  - **Insertions:** Code present in result, but not desired
  - **Moves:** Same code, but on different lines
  - **Path Difference:** Same structure, but slightly different
  - **Conflicts:** Conflicts present after merge
- **Overall accuracy** is measured as the distance from desired version using the vector of all measurements:

$$Overall = Deletions^2 + Insertions^2 + Moves^2 + \cdots$$

Abstract Data Structure

- Focused on Java & Python code to provide the general structure for both **Statically & Dynamically Typed Languages**
- Used the **CST** (Concrete Syntax Tree) generated by **Tree-Sitter** to develop an Abstract Data Structure for Python & Java Code (figure 2)
- Data Structure is composed of:

Main Root: Root Node of File
Pack: Directory of imported package
End: Name of imported package
Class: Class Declaration
Method: Method/Process Declaration
Field: Field/Variable Declaration
Comment: Block/Line Comment

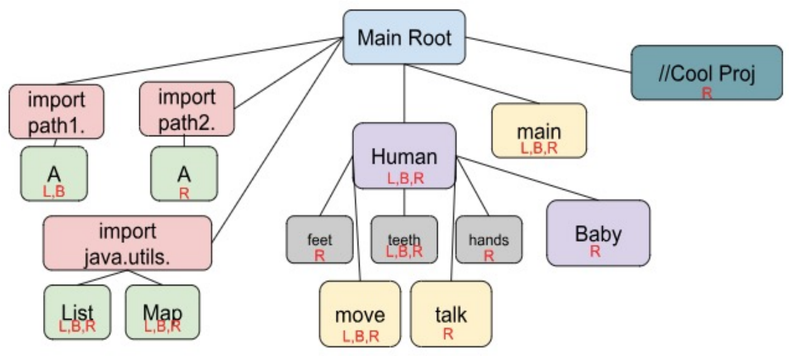

Figure 2: Structure of MethodUnion designed for Python and Java code

Selective Set Union

Different in No Versions

Versions	Selected	Cross-Check
R	R	True
L	L	True
B	None	True
R,L	R,L	True
B,L	B,L	True
B,R	B,R	True
L,B,R	L,B,R	False

Different in 2 Versions

Versions		Selected	Cross-Check
Path 1	Path 2		
R,L	B	R,L	False
B,L	R	R	False
B,R	L	L	False
B	L	L	True
B	R	R	True
L	R	max(L,R)/Git Merge	True

Different in All Versions

Versions			Selected	Cross-Check
Path 1	Path 2	Path 3		
B	L	3	max(L,R)/Git Merge	False

Results

- Compared results using 2 of our developed tools.
- Both tools run the same heuristics for import statements. **CompressedTree** uses Git for the body, whereas **MethodUnion** uses our heuristics
- Java & Python results are based off 86 and 68 case studies respectively

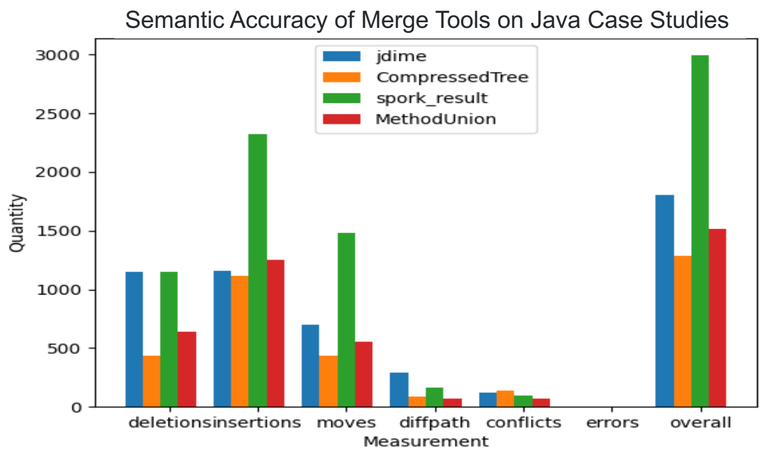

Figure 3: Comparing Structured Java merge tools using Java Cases

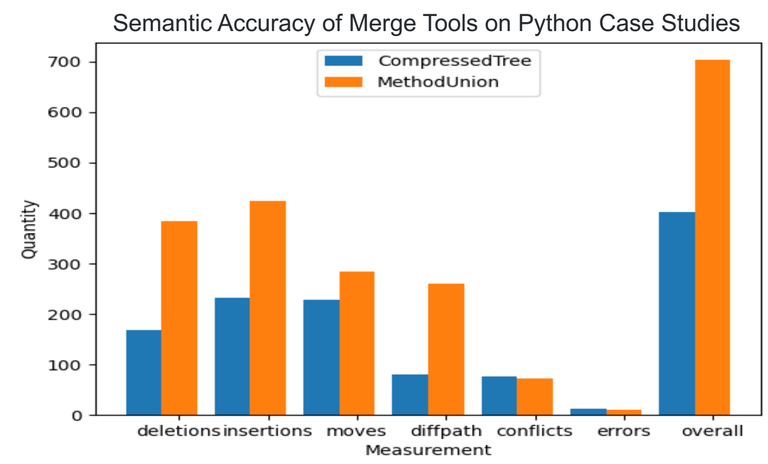

Figure 4: Comparing Structured Python merge tools using Python Cases

Conclusions

Improvement Over Existing Tools (Figure 3):

- Overall empirical results suggest that CompressedTree & MethodUnion are **more semantically equivalent** to the desired version than other tools. Results show that:
  - Tools are removing less necessary code.
  - Merged result will include less unused code.
  - Original code structure is being preserved better.
  - Tools choose correct changes more often.

Extension to Other Languages (Figure 4):

- CompressedTree and MethodUnion also demonstrate strong accuracy for Python.
  - Results for Java were from a total of 86 case studies, whereas Python a total of 68 case studies. Given this, the ratio of Overall results to case studies is better for Python.

CompressedTree vs MethodUnion:

- Overall difference is small in Java, meaning our heuristics provides approximately the same accuracy as Git Merge on the body with half the conflicts.
- Logically large different for Python makes sense for now since we have yet to include comments and maintain the code structure for it.

Future Work

- Improve MethodUnion accuracy for Python code by including comment blocks and utilizing existing attributes in code for ordering code segments using line numbers like Java.
- Extend MethodUnion to include a representation for code within methods in the general AST. Requires differencing using sequential sets. Empirically test and validate tool by testing new heuristics on method body.
- Extend tools to integrate more Statically (C, Rust) and Dynamically (Ruby, JavaScript) typed languages.

References

[1] S. LARSÉN, "Spork : Move-enabled structured merge for Java with GumTree and 3DM," thesis, School of Electrical Engineering and Computer Science, STOCKHOLM, SWEDEN , 2020

[2] akullpp, "awesome-java" https://github.com/akullpp/awesome-java (accessed August 8, 2023)

[3] V. Chen, "awesome-python" https://github.com/vinta/awesome-python (accessed August 8, 2023)

[4] J.-R. Falleri, F. Morandat, X. Blanc, M. Martinez, and M. Monperrus, "Fine-grained and accurate source code differencing," thesis, International Conference on Automated Software Engineering, Vasteras, Sweden, 2014

[5] Tree-Sitter, https://tree-sitter.github.io/tree-sitter/ (accessed Aug. 8, 2023)

Nirmal Chaudhari
B.Eng Software Engineering

McMaster University, Department of Computing & Software

Email: chaudn12@mcmaster.ca
Phone: +1(647) 619-1087

BRIGHTER WORLD