Jediah Conachan (jediah6)
Alva Wei (alvawei)

Automating Merge Conflict Resolution

Vision

We will try to solve the problem of merge conflicts. Currently, merge conflicts require a developer to manually look at the conflicting lines and resolve the conflict by hand. In small projects, this is not an issue, but in anything larger, this can mean a lot of extra work.

A lot of merge conflicts aren't actually "conflicts"; they can simply be an issue of extra whitespace, different variable names, a changed file name, etc. There do exist tools that deal with some of these types of conflicts: Git has an "ignore whitespace" option, and Team Foundation Server only auto-resolves certain categories of merge conflicts. In these kinds of tools, the conflicts that can't be easily auto-resolved are still left up to the developer to manually fix. Bitbucket is an example of a tool that tries to auto-resolve all conflicts. It does this by relying on the branches having a defined order, with priority given to the "newer" branch.

Approach

Our approach will attempt to combine previous approaches: non-conflicts will be categorized and handled accordingly, and all other conflicts will be auto-resolved by some algorithm (without necessarily relying on branch ordering like Bitbucket).

To resolve a merge conflict, we must first determine whether the changes truly conflict. One attempt involves creating abstract syntax trees (AST) from the code changes. If the syntax trees are the same, then the conflicts are superficial and we only need to decide which new variable name or extra whitespace to keep.

Otherwise, the two branches have conflicts that are not easily resolvable and we must determine which changes to keep or how to combine the changes. This could involve some form of ordering branches (like Bitbucket), prioritizing certain types of changes, or looking at historical data to see how past conflicts have been resolved.

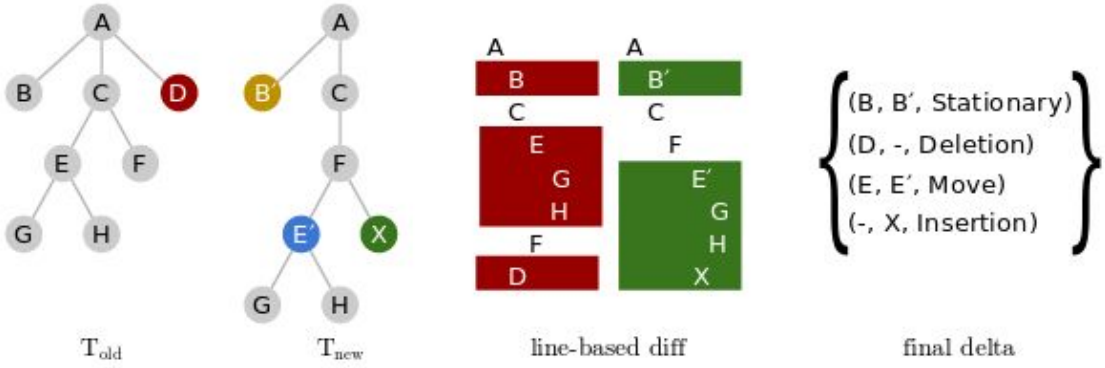

Image from [“Precise Version Control of Trees with Line-based Version Control Systems.”]

Risks

Our goal is to automatically resolve merge conflicts, minimizing the amount of work done by programmers. A problem, however, is the quality of how the merge conflict is automatically resolved. We could automate merge conflicts by simply choosing the changes of a randomly selected branch. This is probably not what the programmers want. If we create new code from “solving” a merge conflict that is unsatisfactory to the programmers, we have created more work for them. They would still need to manually solve the original merge conflict, with our “solution” code in their way.