

Understanding Git with Alloy

Milestone 3

Cláudio Lourenço Renato Neves

University of Minho
Formal Methods in Software Engineering

July 10, 2012



Table of contents

Git as VCS

Project motivation and objectives

Git internals

Specification of operations

Documentation

Conclusion



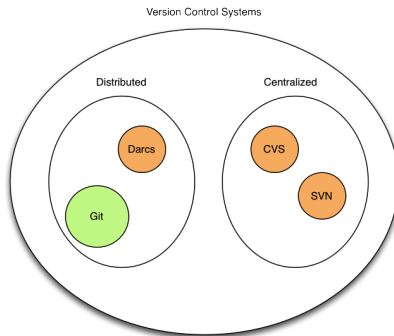
Git as VCS

Git is one of many Version Control Systems

- Fast
- Efficient
- Oriented to snapshots, not differences
- Widely used



Git as VCS



Motivation for this project

Gap in the understanding of Git

- Lack of precise descriptions
- Contradictions in some manuals

An opportunity appears

- Developers could benefit from a manual that is precise and rigorous



The dark world of Git

The common (and above average) knowledge of Git

- "if there are any uncommitted changes when you run git checkout, Git will behave very strangely." ¹
- "When you create a branch, it will contain everything committed on the branch you created it from at that given point. So if you commit more things on the master branch like you have done (after creating b), then switch to branch b, they won't appear. This is the correct behavior. Does that answer your question?" ²

¹"Understanding Git" Manual

²An user of Git development mailing list



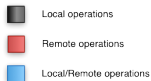
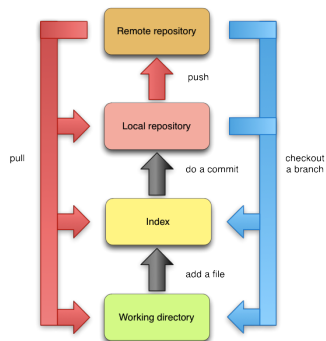
Objectives of this project

Shine some light in Git internals

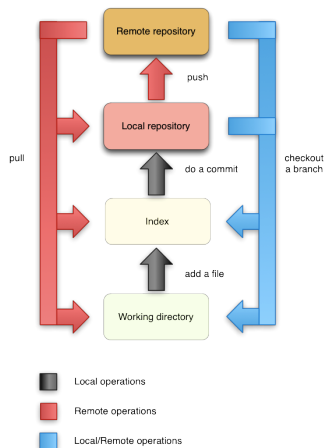
- Build a precise model of how Git works, using Alloy
- Analyze the model and verify which properties does (not) guarantee
- Help others understand Git, building a manual with public access



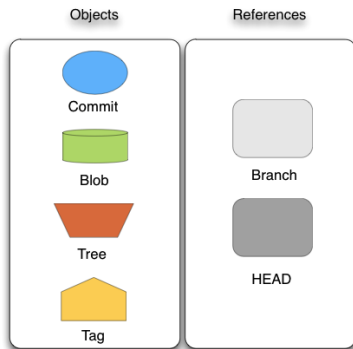
The Git Structure



Repository



Repository



Blob and Tree

Blob

- Represents the content of a file
- The identifier is calculated from its content, thus files with same content will share the same blob

```
sig Blob extends Object {}
```

Tree

- Contains Blobs or other Trees
- Used to represent the file system structure
- The identifier is calculated in a similar way to the Blobs. Thus, for Trees sharing also exists

```
sig Tree extends Object {  
  contains: Name -> lone(Tree+Blob)  
}
```



Commit

- A snapshot of the project on a certain moment in time
- Has a set of parents, that are considered the previous commits
- Points to a Tree that is equivalent to the root folder in the working directory
- An auxiliar relation named "Abstract" was specified to be easy to model the commit operation

```
sig Commit extends Object {  
  points : Tree,  
  parent : set Commit,  
  abs: Path  $\rightarrow$  Object,  
  merge : set State  
}  
  
sig RootCommit extends Commit {}
```



Branch and HEAD

Branch

- Pointer to a commit, unlike other VCS where a branch is a full copy of the repository

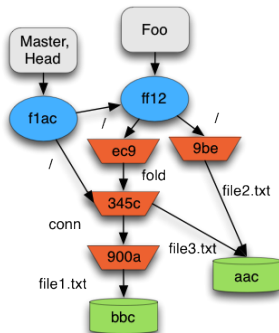
HEAD

- Special reference that identifies the current Branch

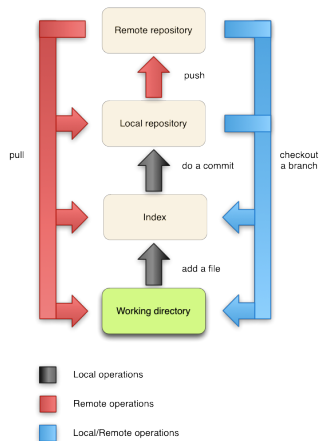
```
sig Branch{  
  marks: Commit lone → State ,  
  branches: set State ,  
  head: set State  
}  
  
lone sig Master extends Branch{}
```



Repository



Working Directory



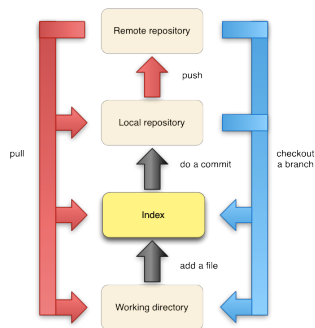
Working Directory




- Subset of a file system
- Has the content of a project
- Files can be the current files or files retrieved from the repository

```
sig Path {  
  pathparent: lone Path,  
  name: Name,  
  unmerge: set State  
}  
  
one sig Root extends Path{}
```



Repository



-  Local operations
-  Remote operations
-  Local/Remote operations



Index

- Contains all files that are going to be committed on the next commit
- The index is not necessarily equal to the working directory
- If an user wants to commit a new file or a newly modified file, first he must add it to the index

```
sig File{  
  path: Path,  
  blob: Blob,  
  index: set State  
}
```

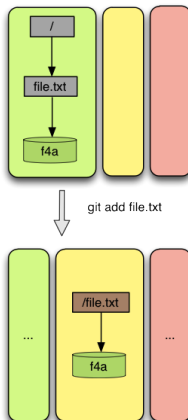


Modeled Operations

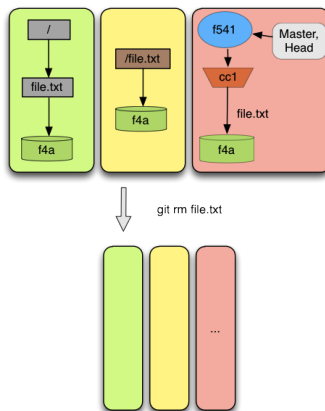
- Add and Remove
- Commit
- Branch and Branch Remove
- Checkout
- Merge (2-way and fast-forward)



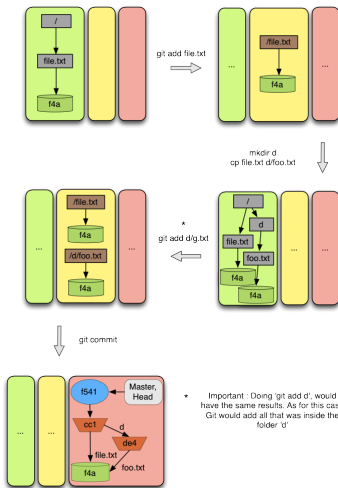
Add



Remove

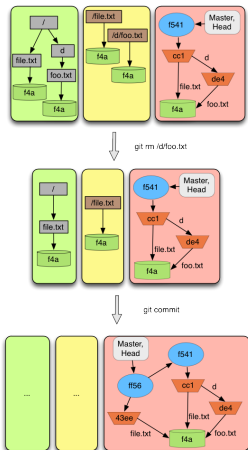


Commit

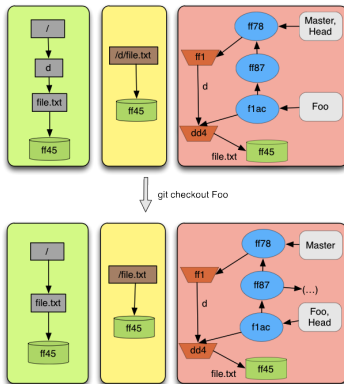


* Important: Doing 'git add d', would have the same results. As for this case Git would add all that was inside the folder 'd'.

Commit



Checkout



Checkout

The most difficult operation to specify

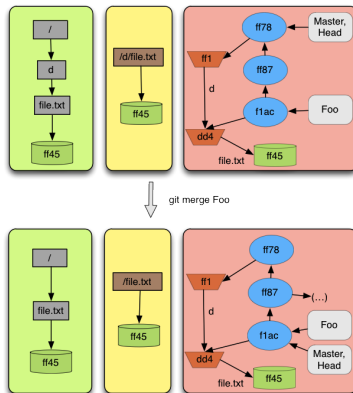
- Expected pre-conditions were not found
- Instead we found weaker pre-conditions
- Strange behaviour caught. Discussed about it with the Git mailing list. Concluded that was a bug



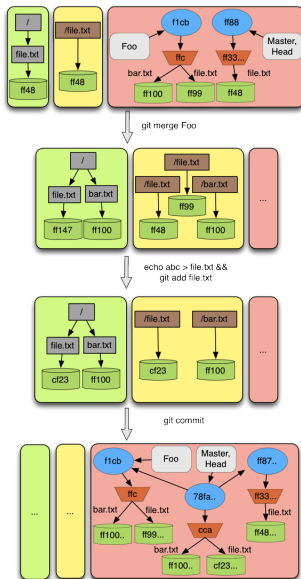
Pre-conditions found



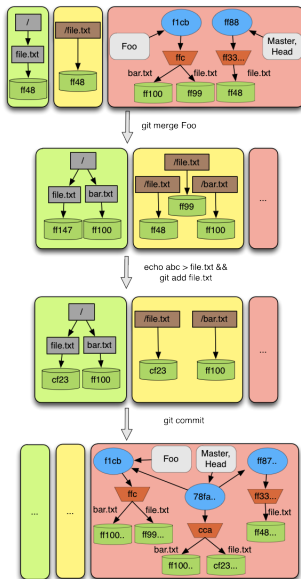
A fast-forward Merge



A 2-way Merge



A 3-way Merge



Merge

The la la

- aaawith the Git mailing list. Concluded that was a bug



Merge

The la la

- aaawith the Git mailing list. Concluded that was a bug



Website

- Website created based on the manual
- http://nevrenato.github.com/CSAIL_Git



Future Work

- Model more operations (rebase, fetch, 3-way merge...)
- Specify more properties that the model does (not) guarantee
- Build interactive diagrams of concrete examples of operations



Conclusions

