

Understanding Git with Alloy

Milestone 1

Cláudio Lourenço Renato Neves

University of Minho
Formal Methods in Software Engineering

May 2, 2012



Table of contents

Version Control System

Git in a nutshell

Project Goals

Progress so far

Future Work



Version Control System

What is a VCS?

- Records changes on files over time
- Recall old versions of files

Local VCS

No collaboration with others users - RCS

Centralized VCS

All files are stored on a central server - CVS,
Subversion, Perforce



Distributed VCS

Each client has a mirror of the repository - Git, Mercurial, Bazaar,
Darcs

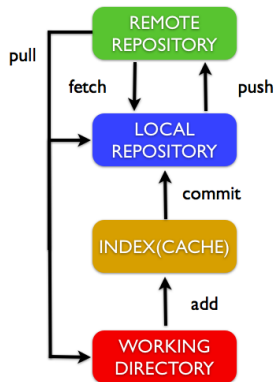


Git in a nutshell

- It was created in 2005 by Linus Torvalds
- Distributed Version Control System
- Simple, Fast, Efficient
- It keeps snapshots, not differences
- Operations with branches are very cheap

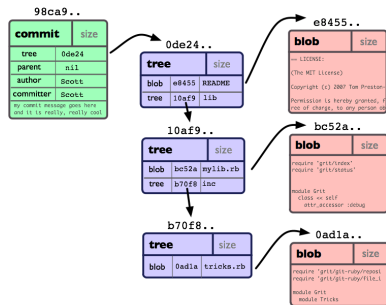


Git simplified workflow



The Git Object Model

- Similar to a filesystem
- Each git object is named by a sha
- Blob stores file data
- Tree references a set of others trees and blobs
- Commit points to a single tree
- Commit can have more than one parent



Project Goals

- Build a precise model of how Git works
- Analyze the model
- Check which properties the model does (not) guarantee
- Compare to other systems
- Build a concise user manual based on the model



What has been done so far

First Approach

- Model Working Directory
- Model Index
- Model Object Model
 - Object hash are modeled implicitly



First Approach - Object Model

```
sig Sha{}

abstract sig Object {
  namedBy : Sha
}

sig Blob extends Object{
}

sig Tree extends Object {
  references : some (Tree+Blob)
}

sig Commit extends Object{
  points : Tree,
  parent : set Commit
}

sig RootCommit extends Commit{}
```



First Approach - Working Directory

```
abstract sig WDOBJECT{  
    wdparent: lone Dir  
}  
  
sig File extends WDOBJECT{  
    content: Sha  
}  
  
sig Dir extends WDOBJECT{  
  
one sig Root extends Dir{}
```



First Approach

Index

```
one sig Index{  
  stage: Sha one -> File  
}
```

Problems of the first approach

- Model got too complex when adding operations
- We need the name of the files and directories



What has been done so far

Second Approach

- Focus on Object Model and Index
- Files are associated with a path and a blob
- Object hash are the alloy atom's name



Second Approach - Object Model

```
sig Name {}  
sig State {}  
  
abstract sig Object {  
  objects: set State  
}  
  
sig Blob extends Object {}  
  
sig Tree extends Object {  
  contains : Name -> one (Tree+Blob)  
}  
  
sig RootCommit extends Commit {}  
  
sig Commit extends Object {  
  points : one Tree,  
  parent : Commit set  
}
```



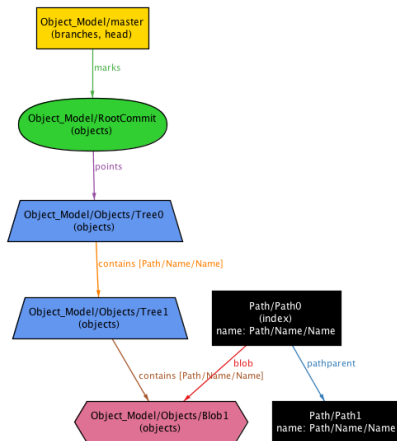
Second Approach

Index and Path

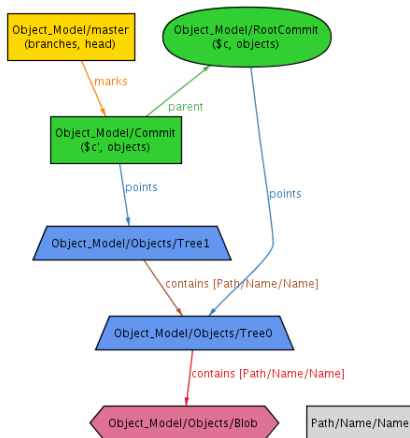
```
sig Path {  
  pathparent : lone Path,  
  name : Name,  
  blob:lone Blob,  
  index: set State  
}
```



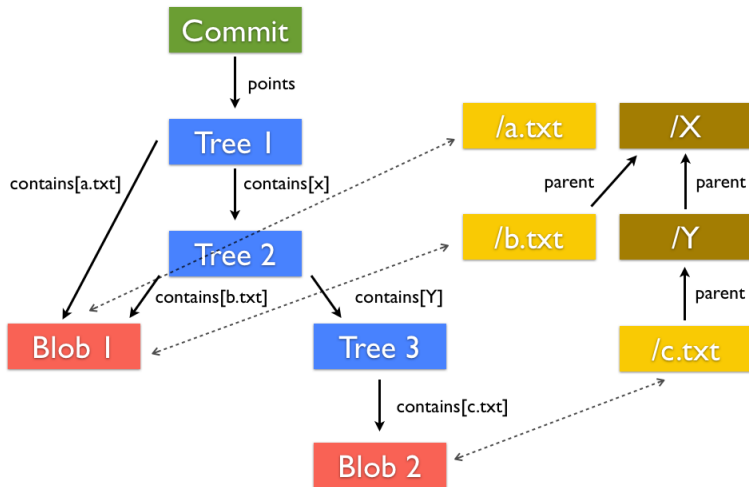
Instance - A single commit corresponding to an index



Instance - Commits sharing objects



Problems we are facing



Possible Solution

```
sig Commit extends Object {  
  points : one Tree ,  
  parent : Commit set ,  
  abs: Object lone -> some Path  
}
```



Future work

- Find a solution for the current problem (Suggestions?)
- Model some operations relatively to the remote repository
- Check some properties



Understanding Git with Alloy

Milestone 1

Cláudio Lourenço Renato Neves

University of Minho
Formal Methods in Software Engineering

May 2, 2012

