

Understanding Git with Alloy

Milestone 2

Cláudio Lourenço Renato Neves

University of Minho
Formal Methods in Software Engineering

June 9, 2012



Table of contents

Where were we?

Progress

Current Model

The operations

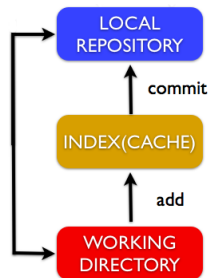
The properties

Future work

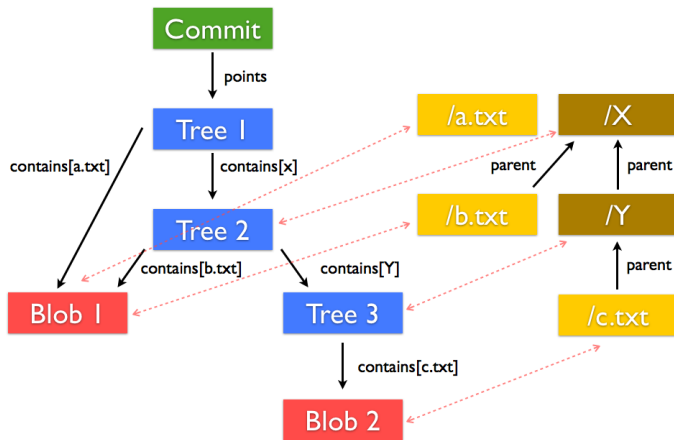


Where were we?

- Focusing on Index + Object Model
- No problem with add and rm operations
- Commit - Big Problem



The problem



How did we solve it?

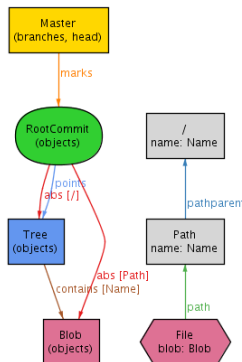
Abstraction

- In each commit there is a relation between Objects and Paths
- Facts to reflect the parent relationship
 - The Tree pointed by the Commit corresponds to the root
 - A contains relations exists, if and only if, the converse parent relation exists

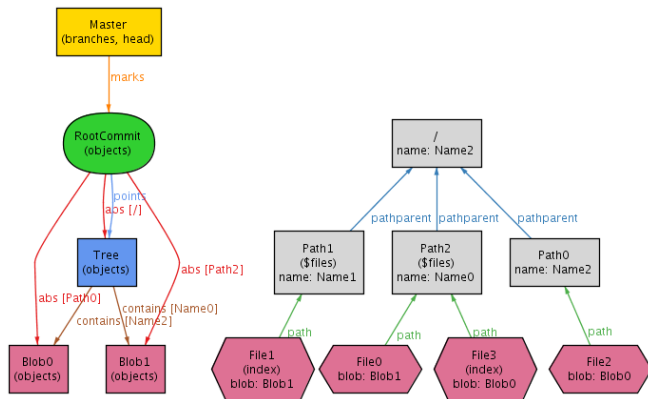
```
sig Commit extends Object {  
  points : Tree,  
  parent : set Commit,  
  abs: Path -> Object  
}
```



The abstraction relation



The abstraction relation



Model

```

abstract sig Object {
  objects: set State
}

sig Blob extends Object {}

sig Tree extends Object {
  contains : Name  $\rightarrow$  lone (Tree+Blob)
}

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

sig RootCommit extends Commit {}

sig Branch{
  marks: Commit one  $\rightarrow$  State,
  branches: set State,
  head: set State
}

lone sig Master extends Branch{}

```

```

sig Path {
  pathparent : lone Path,
  name : Name
}

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

```



Operations - add and rm

add

Add a file with the current content to the index

```
...
index.s' = index.s +
          f - ((f.path) ~ path - f)
...
```

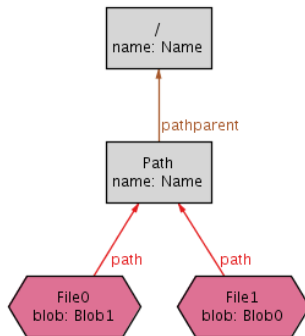
rm

Remove the file from the index

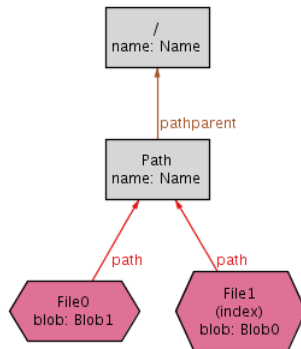
```
...
index.s' = index.s - f
...
```



Operations - add and rm



Operations - add and rm



Operations - commit

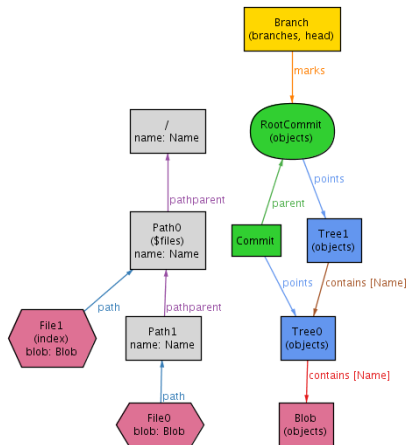
commit

Creates a commit, from the index

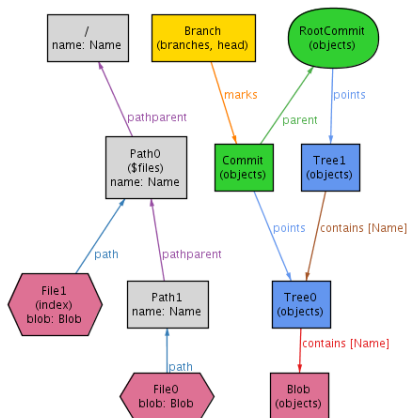
```
...  
(head.s').(marks.s').parent = (head.s).(marks.s)  
...  
(index.s).path.*pathparent = (head.s').(marks.s').abs.univ  
all f:index.s | f.path → f.blob in (head.s').(marks.s').abs  
...
```



Operations - commit



Operations - commit



Operations - branch

branch

Creates a new branch pointing to the current one

```
...  
branches.s' = branches.s + b  
marks.s' = marks.s + b  $\rightarrow$  (head.s).(marks.s)  
...
```

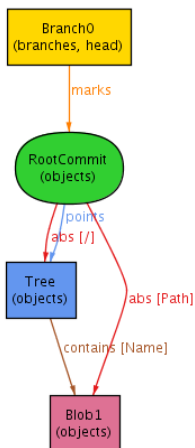
branch -d

Removes a branch if it is not pointed by the head

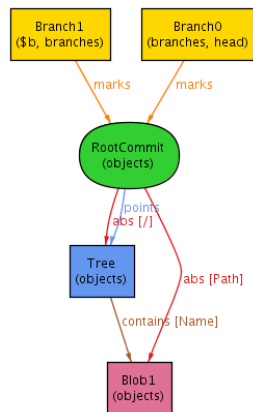
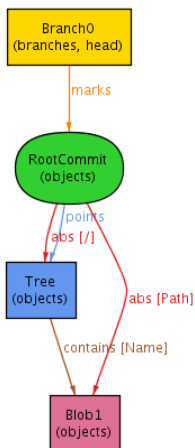
```
...  
branches.s' = branches.s - b  
marks.s' = marks.s - b  $\rightarrow$  Commit  
...
```



Operations - branch



Operations - branch



Operations - checkout

Updates the head and the index to reflect a certain commit pointed by a branch

Problems

- Difficulty to understand the pre-conditions
- There are no specifications

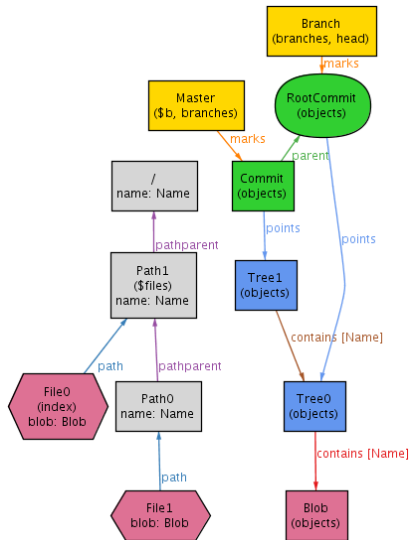


Pre-conditions found relatively to the index

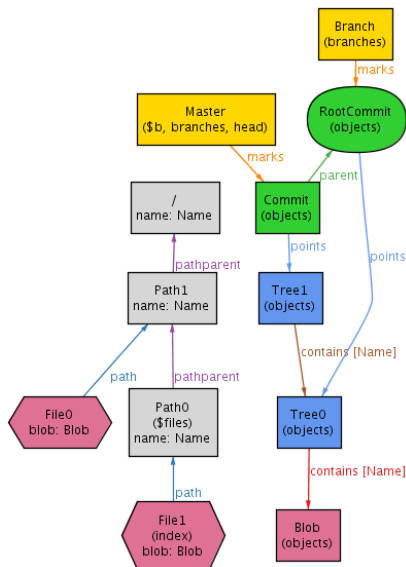
- Removed files are ignored
- Non modified files since the last commit are ignored
- New files or Modified files:
 - if file exists on the destination commit then:
 - the content is the same as in the index, or
 - the content in the destination commit is the same as in the current commit
 - if the file does not exist in the destination commit
 - it also does not exist on the current commit



Operations - checkout



Operations - checkout



Properties

Invariant preservation

All operations preserve the invariant

all $s, s': \text{State} \mid \text{invariant}[s] \text{ and } \text{operation}[s, s', \dots] \text{ implies } \text{invariant}[s']$

```
pred invariant[s:State]{
  some Commit & objects.s <=> some marks.s && one head.s
  head.s in branches.s & (marks.s).Commit
  (objects.s - Commit) in (objects.s).points.*(contents.Name)
  all t : objects.s & Tree | t.contents.Name in objects.s
  all c:objects.s & Commit | c.points in objects.s and c.parent in objects.s
  marks.s in branches.s -> one objects.s
  all s:State, t : objects.s & Tree | some t.contains
}
```



Properties

Idempotence

- After performing an operation, repeating it does not change the state
- Add, commit and checkout are idempotent

```
all s0,s1,s2 : State | operation[s0,s1,...] and operation[s1,s2,...]  
    implies dynamicRelations[s1] = dynamicRelations[s2]
```



Future work

- Merge operation
- Document operations and properties



Understanding Git with Alloy

Milestone 2

Cláudio Lourenço Renato Neves

University of Minho
Formal Methods in Software Engineering

June 9, 2012

