Understanding Git with Alloy

Milestone 2

Cláudio Lourenço Renato Neves

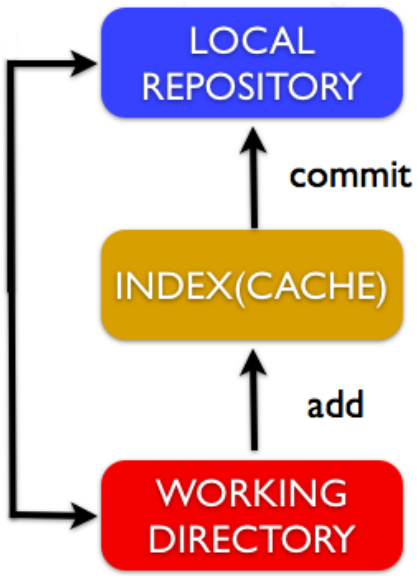
University of Minho
Formal Methods in Software Engineering

June 9, 2012

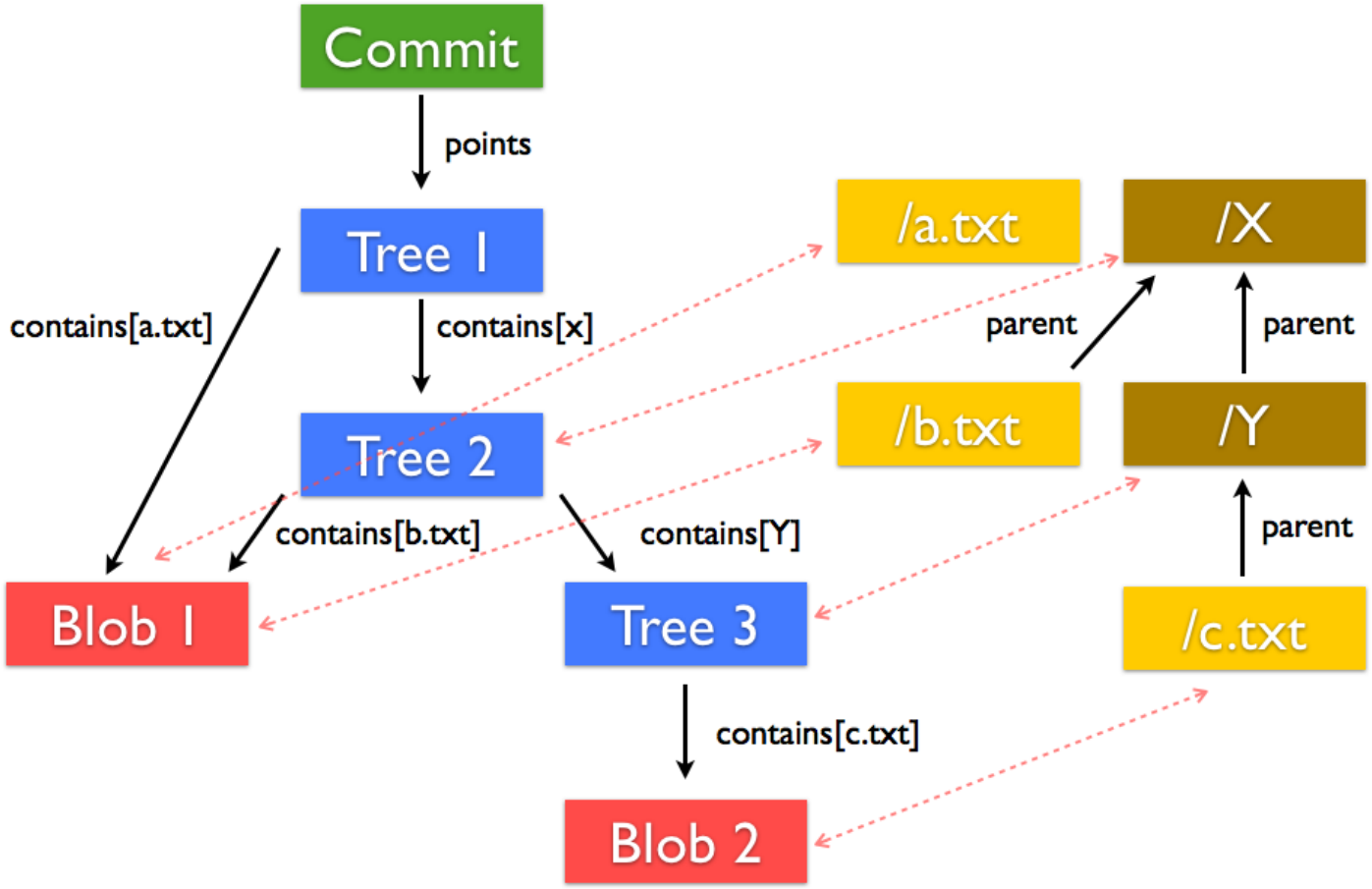
Table of contents

Where were we?

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



The problem


Commit
points
Tree 1
contains[a.txt]
contains[x]
Tree 2
contains[b.txt]
contains[Y]
Blob 1
Tree 3
contains[c.txt]
Blob 2
/a.txt
/X
/b.txt
/Y
/c.txt
parent
parent
parent

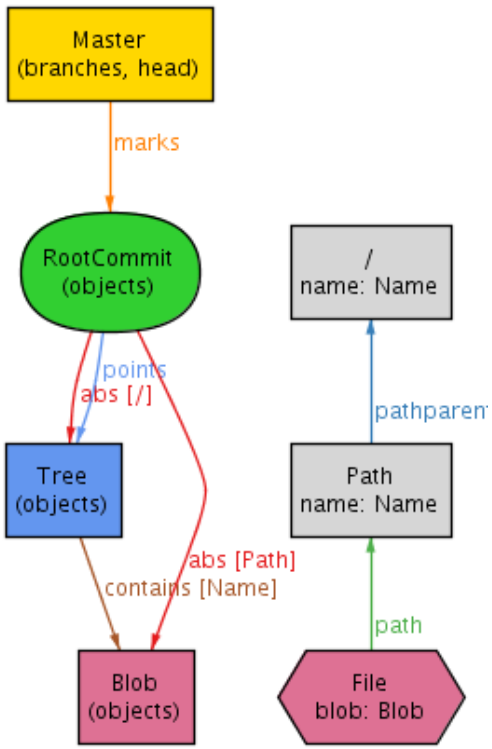
How did we solve it?

Abstraction

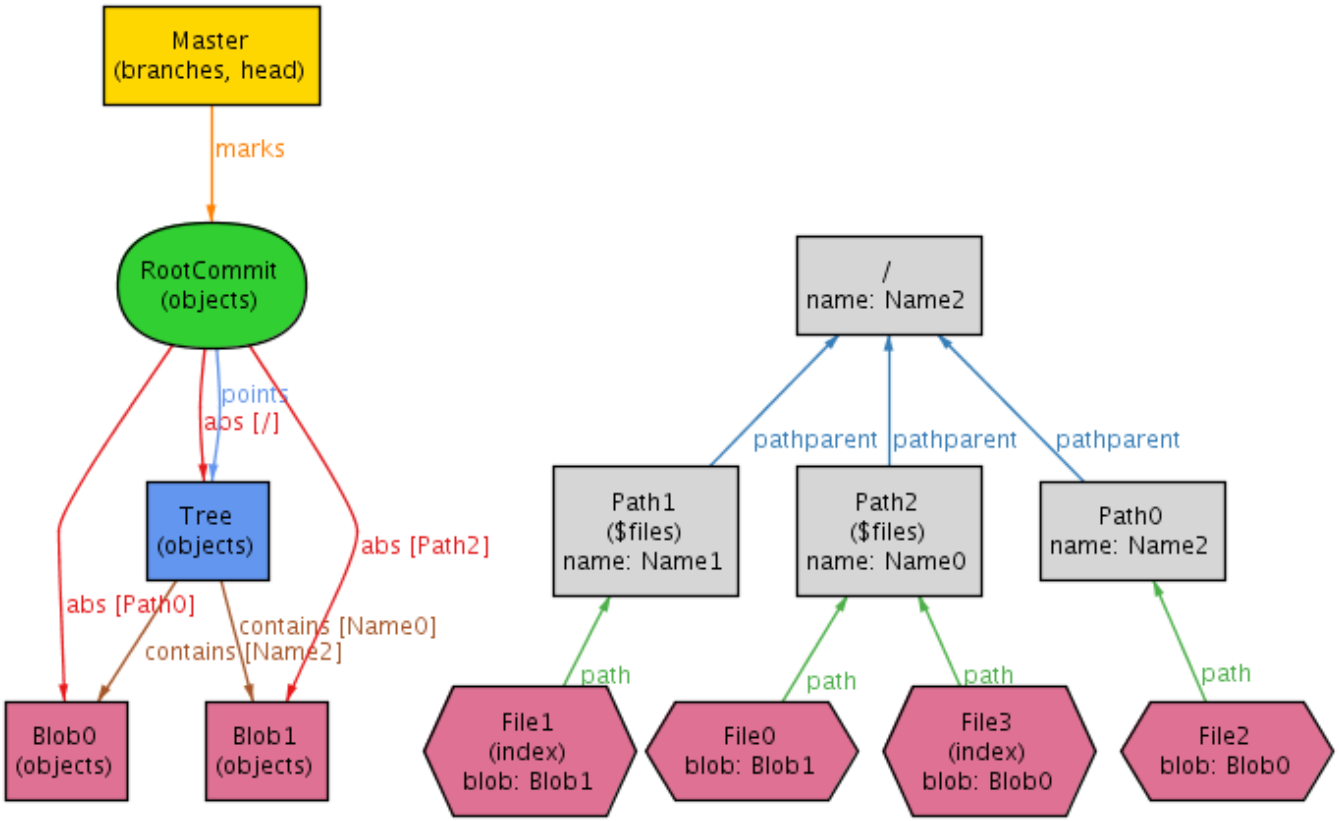
- In each commit there is relation between Objects and Paths
- Facts to reflect the parent relationship
  - The Tree pointed by the Commit corresponds to the root
  - A contains relations exists, if and only if, the converse parent relation exists

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

The abstraction relation



The abstraction relation



CSAIL

Model

```
abstract sig Object {
   objects: set State
}

sig Blob extends Object {}

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

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

sig RootCommit extends Commit {}

sig Branch{
   marks: Commit one -> 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

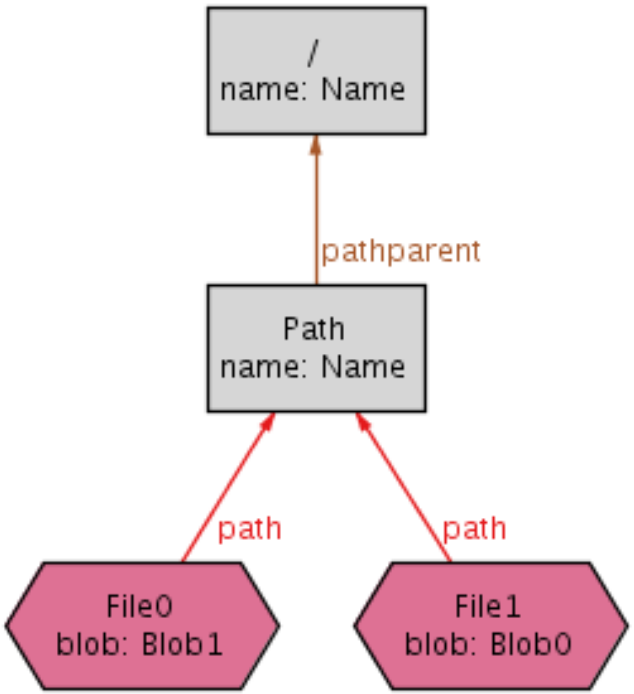
```
...
index.s' = index.s +
           f - ((f.path).~path -f)
...
```

rm

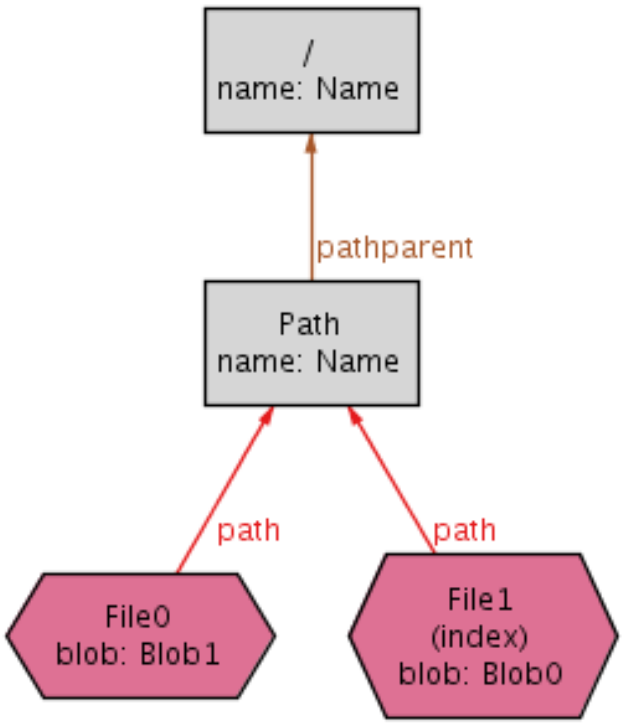
Remove the file from index

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

Operations - add and rm



Operations - add and rm



CSAIL

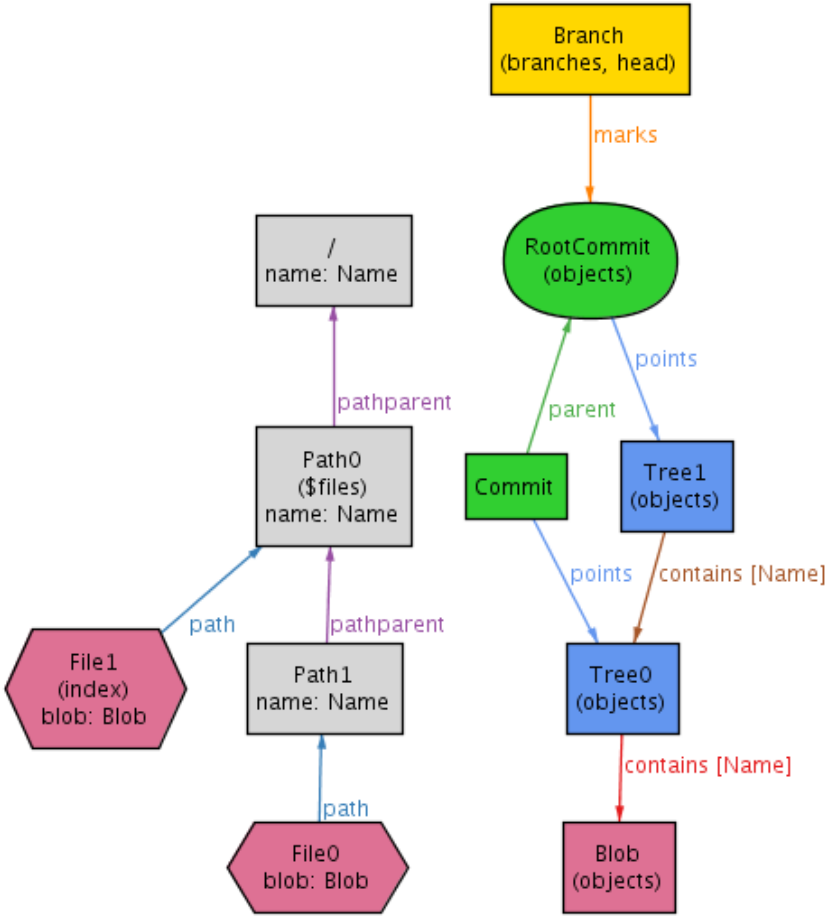
Operations - commit

commit

Creates a commit, from the index

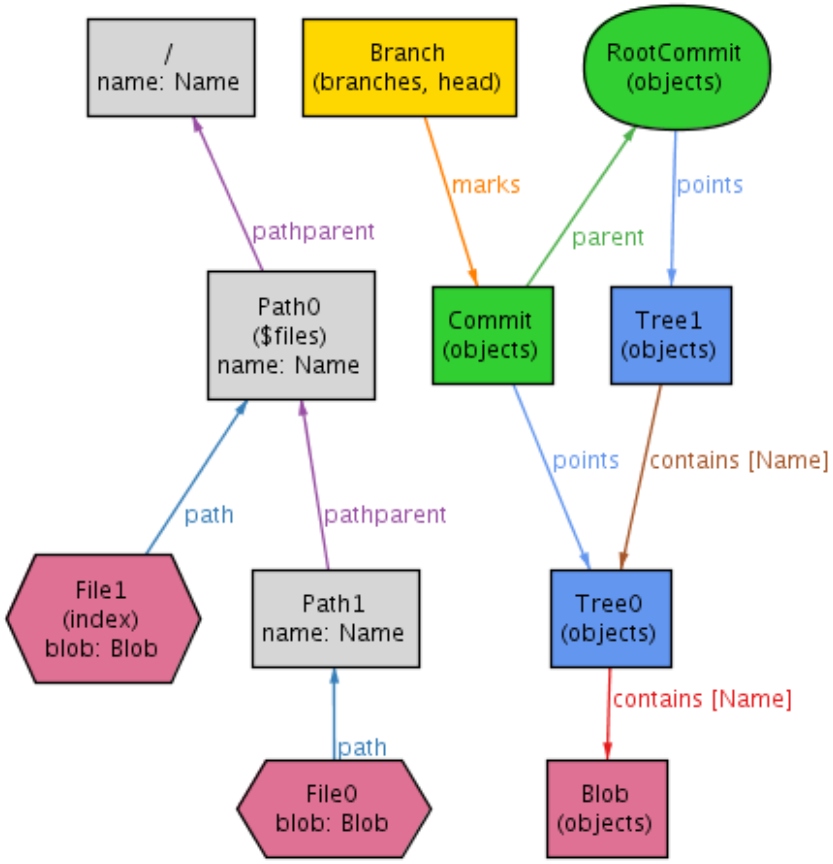
```
...
(head.s').(marks.s').parent = (head.s).(marks.s)
...
(index.s).path.*pathparent = (head.s').(marks.s').abs.univ
all f:index.s | f.path -> f.blob in (head.s').(marks.s').abs
...
```

Operations - commit



CSAIL

Operations - commit



Operations - branch

branch

Creates a new branch pointing to the current one

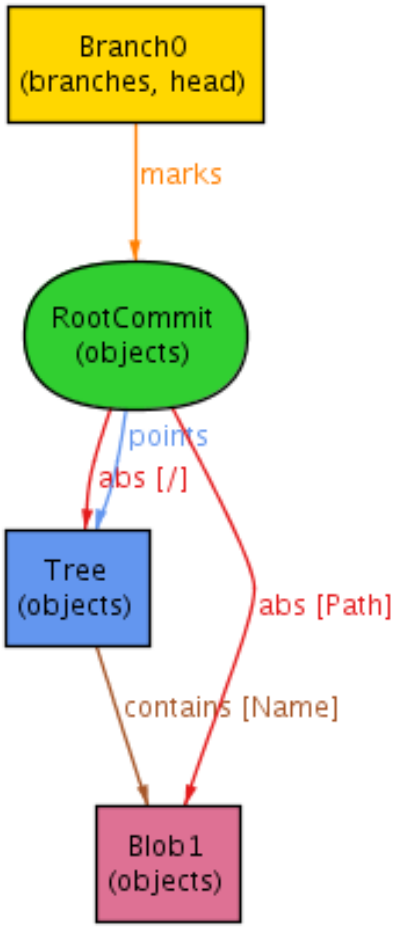
```
...
branches.s' = branches.s + b
marks.s' = marks.s + b -> (head.s).(marks.s)
...
```

branch -d

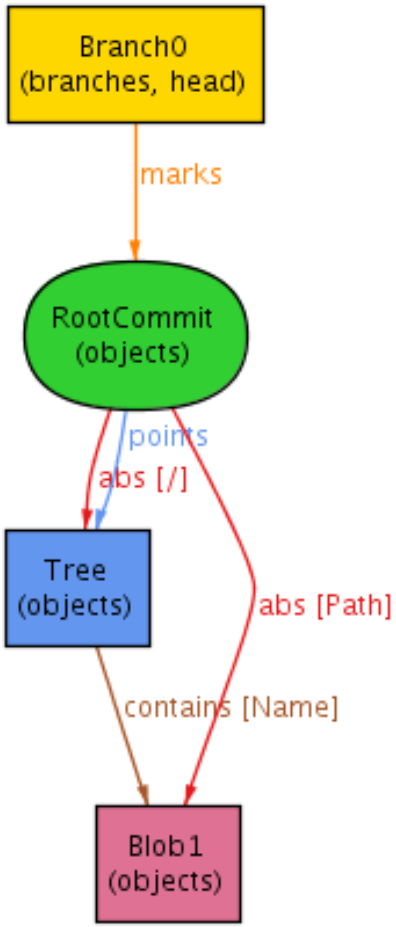
Removes a branch if it is not pointed by the head

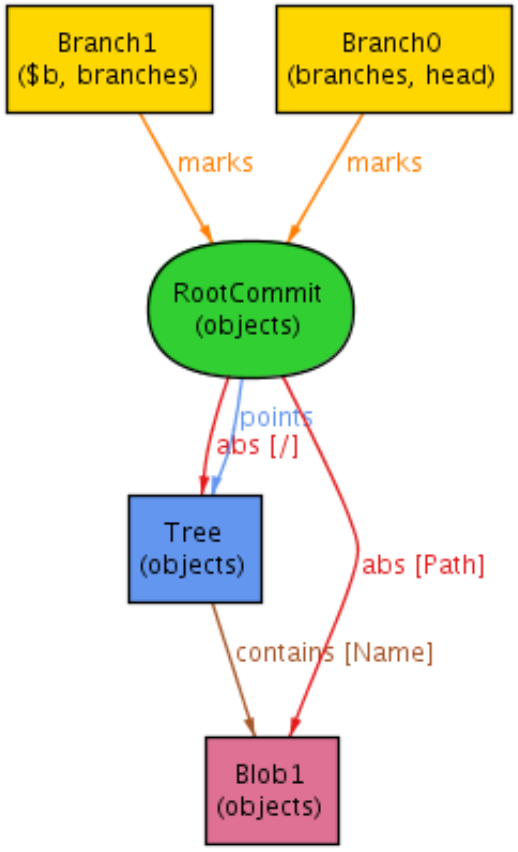
```
...
branches.s' = branches.s - b
marks.s' = marks.s - b -> Commit
...
```

Operations - branch



Operations - branch





Operations - checkout

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

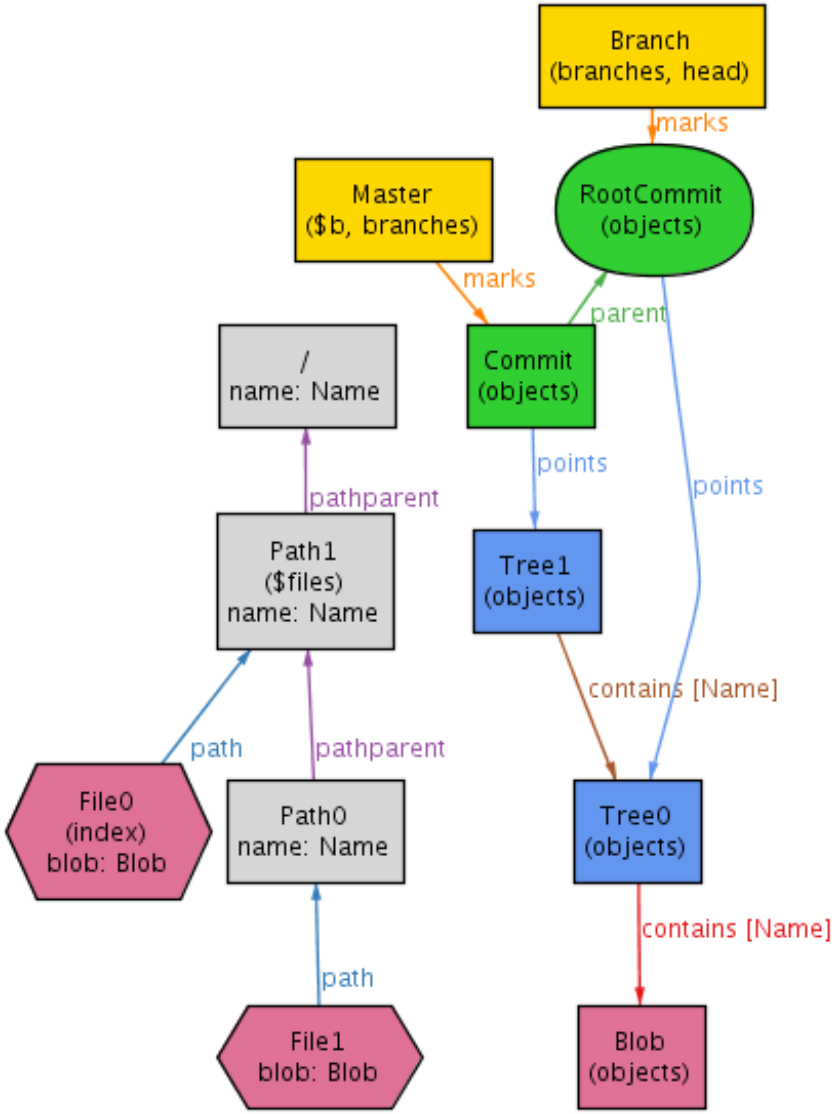
Problems

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

Pre conditions found relatively to the index

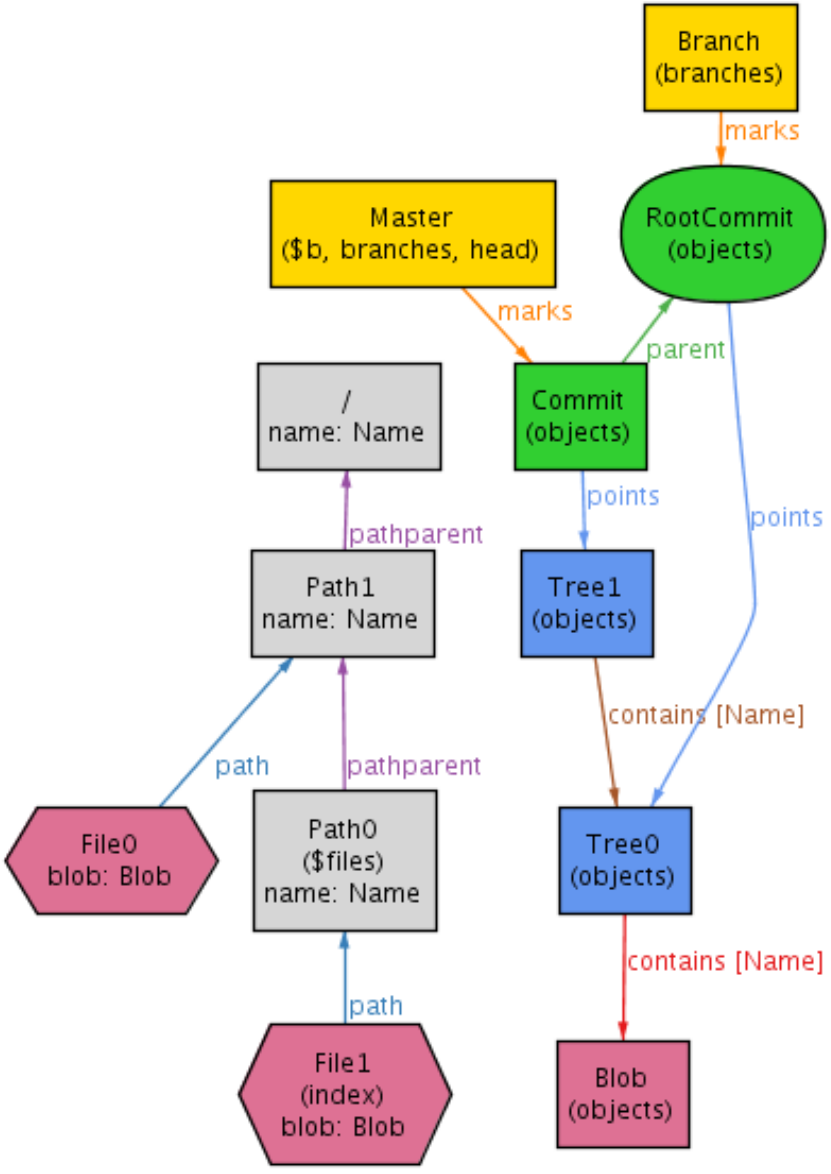
- Removed files are ignored
- Not modified files since last commit are ignored
- New files or Modified files:
  - if file exists on destination commit then:
    - the content is the same as in index, or
    - the content in the destination commit is the same as in the current commit
  - if the files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

Operations - checkout



CSAIL

Operations - checkout



CSAIL

Properties

Invariant preservation

All operations preserve the invariant

all s,s':State | invariant[s] and operation[s,s',...] implies 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 doing an operation repeating 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