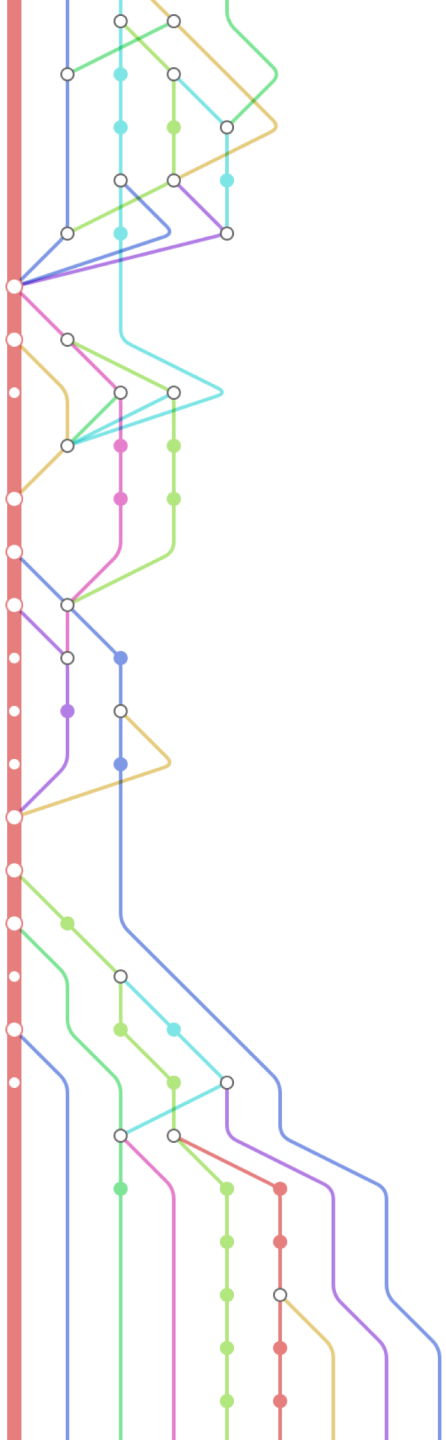


Git Planner

EAAI-17

Joshua Eckroth

CSCI 431 / Artificial Intelligenc ×

Joshua

csci431.artifice.cc

Notes

- Syllabus
- Turing Test
- Searching for a solution
  - Uninformed search
  - Informed search
- Finding a plan
  - PDDL
- Beating an adversary
- Evolving solutions
- Representing expert knowledge
  - Drools rules engine
- Logic programming with Prolog
  - Prolog unification
  - Prolog resolution
  - Prolog examples

Assignments

- A01: Organic pathfinding
  due Sep 7, 11:59pm
- A02: Git planner
  due Sep 12, 11:59pm
- A03: Connect Four AI
  due Sep 19, 11:59pm
- A04: Wedding seat assignment
  due Sep 26, 11:59pm
- A05: Student advisor
  due Oct 3, 11:59pm
- A06: Prolog Pokédex
  due Oct 26, 11:59pm
- A07: Automated jury
  due Nov 9, 11:59pm
- A08: NFL play-by-play
  due Nov 21, 11:59pm
- A09: Recipe classification
  due Nov 30, 11:59pm

CSCI 431

Fall 2016

jeckroth@stetson.edu

Office hours:
Mon/Wed 12-2:30

Goals

- Expand their knowledge of search states to partial state descriptions
- Practice translating a real-world domain into a planning domain
- Practice with a planning engine (and possibly search heuristics)
- Practice learning a new language quickly


git

TUTORIALS / Git

Git Tutorial: 10 Common Git Problems and How to Fix Them

#Git – October 21st 2014 Like 5 Share Tweet

Learning Git? This Git tutorial covers the 10 most common Git tricks you should know about: how to undo commits, revert commits, edit commit messages, discard local files, resolve merge conflicts, and more.

1. Discard local file modifications

Sometimes the best way to get a feel for a problem is diving in and playing around with the code. Unfortunately, the changes made in the process sometimes turn out to be less than optimal, in which case reverting the file to its original state can be the fastest and easiest solution:

```
git checkout -- Gemfile  # reset specified path
git checkout -- lib bin  # also works with multiple arguments
```

In case you're wondering, the double dash (--) is a common way for command line utilities

3250 votes · 21 answers · 1.4m views
How to resolve merge conflicts in Git?
Is there a good way to explain how to resolve merge conflicts in Git?
git · git-merge · merge-conflict-resolution · git-conflict-resolution
asked Oct 2 '08 at 11:31 Spoike 48.6k ● 33 ● 116 ● 141

2890 votes · 28 answers · 743k views
How to clone all remote branches in Git?
I have a master and a development branch, both pushed to GitHub. I've cloned, pulled, and fetched, but I remain unable to get anything other than the master branch back. I'm sure I'm missing ...
git · git-branch · git-clone · remote-branch
asked Sep 15 '08 at 22:42 Peter Coulton 18.9k ● 8 ● 39 ● 64

2726 votes · 26 answers · 1.3m views
How do I discard unstaged changes in Git?
How do I discard changes in my working copy that are not in the index?
git
asked Sep 9 '08 at 19:33 Readonly 91.4k ● 83 ● 176 ● 190

2618 votes · 29 answers · 479k views
How can I add an empty directory to a Git repository?
How can I add an empty directory (that contains no files) to a Git repository?
git · directory · git-add
asked Sep 22 '08 at 16:41 Laurie Young 55.4k ● 11 ● 36 ● 50

2547 votes · 8 answers · 383k views
Move the most recent commit(s) to a new branch with Git
I'd like to move the last several commits I've committed to master to a new branch and take master back to before those commits were made. Unfortunately, my Git-fu isn't strong enough yet, any help? ...
git · git-branch · branching-and-merging
asked Oct 27 '09 at 3:07 Mark A. Nicolosi 22.5k ● 9 ● 35 ● 45

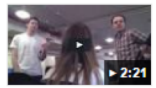
git is hard

All · Images · Videos · News · Shopping · More · Search tools

About 5,460,000 results (0.59 seconds)

Probably Done Before: Why the Heck is Git so Hard?
merrigrove.blogspot.com/2014/02/why-heck-is-**git**-so-**hard**-places-model-ok.html
Feb 3, 2014 - The post will show that one of the reasons (there are other) why going from the SVN model of source-control to **Git** can be so complicated ...

Why is Git so popular despite it being more complicated and harder to ...
https://www.quora.com/Why-is-**Git**-so-popular-despite-it-being-more-complicated-an...
Git, despite its arcane syntax and mercilessly pedantic and obtuse error handling, is popular for ... I didn't really expect anyone to use it because it's so **hard** to use, but that turns out to be its big appeal. No technology can ever be too arcane or ...

GIT IS HARD on Vimeo


https://vimeo.com › Meagan Fisher › Videos
Feb 28, 2013
I accidentally did something so freakishly wrong with **Git** that half the dev team had to come stare at it.

10 things I hate about Git | Steve Bennett blogs
https://stevebennett.me/2012/02/24/10-things-i-hate-about-**git**/
Feb 24, 2012 - **Git** is the source code version control system that is rapidly becoming ... I suspect people are confusing **git** being **hard**, with an absolute inability ...

Can we please stop pretending that Git is simple and easy to learn? If ...
https://news.ycombinator.com/item?id=4200492
Jul 4, 2012 - posts every other day. The fact of the matter is that **Git** is incredibly powerful but also complex and **hard** to learn. This isn't a bash on **Git** at all.

Why the Heck is Git so Hard? | Hacker News
https://news.ycombinator.com/item?id=7174554
Feb 4, 2014 - I agree. Just based on the number of tutorials and posts about how **Git** is difficult / **hard** / weird. Seems to me that **Git** is more complicated than it ...

GIT: a Nightmare of Mixed Metaphors – Nature, Brain, Technology
https://ventrellathing.wordpress.com/2013/01/.../**git**-a-nightmare-of-mixed-metaphors/
Jan 25, 2013 - The answer to why **git is hard** to learn, confusing and inconsistent is history .... consistency, and that this makes Git's UI needlessly hard to learn.

Git - Undoing changes | Atlassian Git Tutorial
https://www.atlassian.com/**git**/tutorials/undoing-changes/
Without the **--hard** flag, **git** reset is a way to clean up a repository by unstaging changes or uncommitting a series of snapshots and re-building them from scratch.
You visited this page on 11/18/16.

How to undo last commit(s) in Git?
I committed the wrong files to Git. How can I undo that commit?
git · git-rebase · git-commit · git-reset · git-revert
community wiki
44 revs, 27 users 23%
Peter Mortensen

RandyFay.com

Home » Blogs » rfay's blog

Avoiding Git Disasters: A Gory Story

Submitted by rfay on Sun, 2011-01-30 12:10

Planet Drupal, git

Edit 2015-08-30: The bottom line years later: Use the (Github's) Pull Request methodology, with a responsible person doing the pulls. You'll never have any of these problems.

I learned the hard way recently that there are some unexpectedly horrible things that can happen to a project in the Git source control management system due to its distributed nature... that I never would have thought of.

There is one huge difference between Git and older server-based systems like Subversion and CVS. That difference is that there's no server. There's (usually) an authoritative repository, but it's really fundamentally just a peer repository that gets stuff sent to it. OK, we all knew that. But that has some implications that aren't obvious at first. In Subversion, when you make a change, you just push that change up to the server, and the server handles applying just that change to the master copy of the project. However, in Git, and especially when using the default "merge workflow" (I'll write about merge workflow versus rebase workflow in another article), there are times when a single developer may be in charge of (and able to unintentionally break) the entire codebase all at once. So here I'm going to describe two ways that I know of that this can happen.

Disaster 1: git push --force

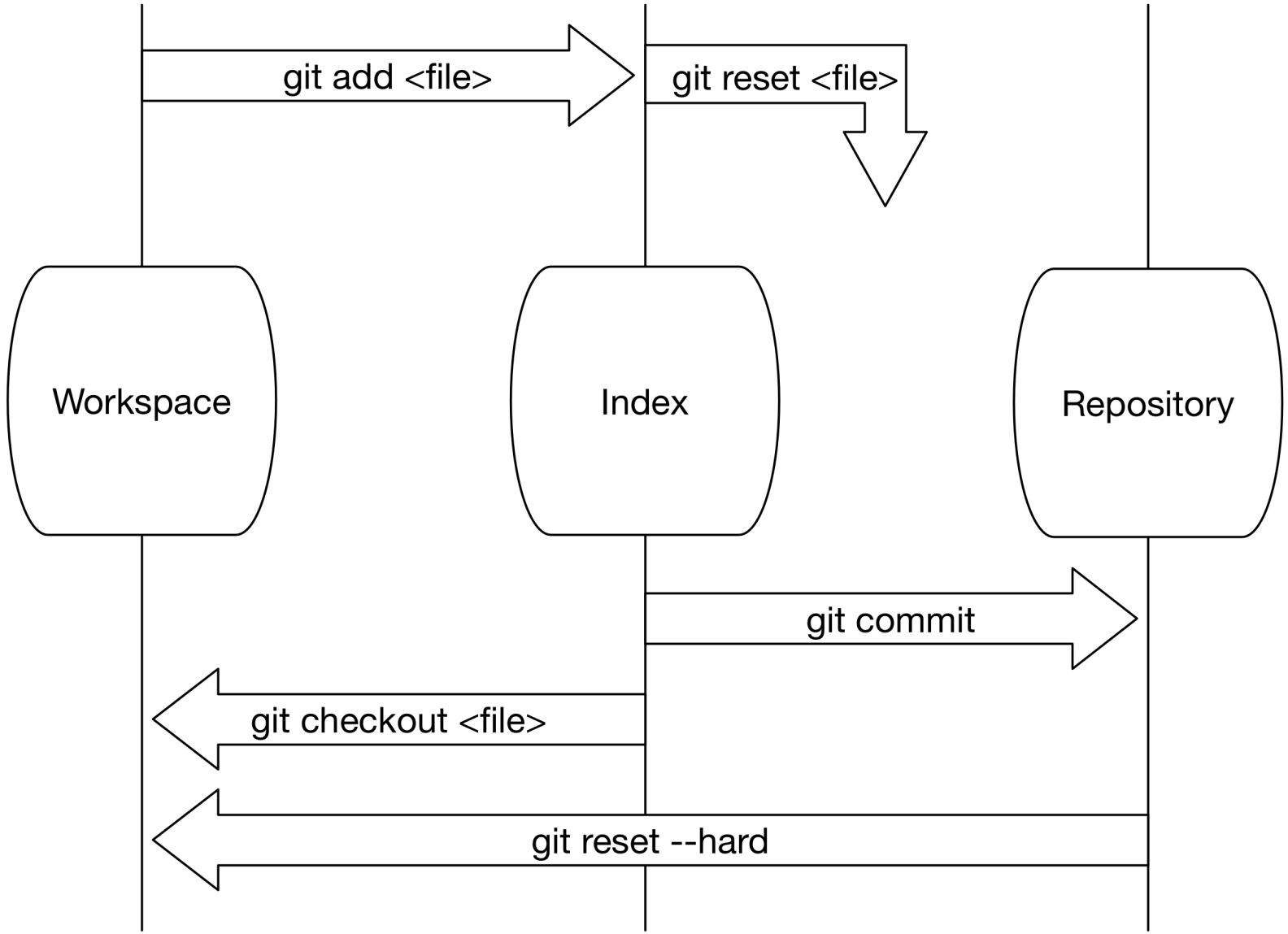
A normal push to the authoritative repository involves taking your new work as new commits and plopping those commits as-is on top of the branch in the repository. However, when a developer's local Git repository is not in sync with (or up-to-date with) the authoritative repository (the one we normally push to), then it can't do a fast-forward merge, and it will balk with an error message.

The right thing to do in this case is to either merge your code with a `git pull` or to rebase your code onto the HEAD with `git pull --rebase`, or to use any number of other similar techniques. The absolutely worst and wrong-est thing in the whole world is something that you can do with the default configuration: `git push --force`. A forced push *overwrites* the structure and sequence of commits on the authoritative repository, throwing away other people's commits. Yuck.

The default configuration in git, that `git push --force` is allowed. In most cases you should not ever allow that.

8967 votes · 31 answers · 3.4m views
How to delete a Git branch both locally and remotely?
I want to delete a branch both locally and on my remote project fork on GitHub. Failed Attempts to Delete Remote Branch $ git branch -d remotes/origin/bugfix error: branch 'remotes/origin/bugfix' ...
git · git-branch · git-remote
asked Jan 5 '10 at 1:12 Matthew Rankin 165k ● 25 ● 89 ● 130

7688 votes · 27 answers · 1.8m views
How to modify existing, unpushed commits?
I wrote the wrong thing in a commit message. Alternatively, I've forgotten to include some files. How can I change the commit message/files? The commit has not been pushed yet.
git · git-commit · git-rewrite-history · amend
asked Oct 7 '08 at 15:44 Laurie Young 55.4k ● 11 ● 36 ● 50


git add <file>
git reset <file>
Workspace
Index
Repository
git commit
git checkout <file>
git reset --hard

PDDL: Planning Domain Definition Language

Domain file:
- name of domain
- list of available predicates
- list of actions; for each:
  - name
  - parameters
  - preconditions
  - effects

Problem file:
- name of domain
- list of objects
- initial state:
  - list of predicates
- goal state

Predicates

(untracked ?f)

(modified-in-workspace ?f)

(deleted-in-workspace ?f)

(updated-in-index ?f)

(added-to-index ?f)

(deleted-from-index ?f)

(committed ?f)

Actions

(add ?f)

(checkout ?f)

(reset ?f)

(reset-hard)

(commit)

Example

Setup

```
mkdir prob1
cd prob1
git init
echo "aaa" > a.txt
```

Goal

a.txt containing just “aaa” is committed.

Manual solution

```
git add a.txt
git commit
```

Problem file

```
(define (problem p)
  (:domain git)
  (:objects "a.txt")
  (:init
    (untracked "a.txt"))
  (:goal
    (committed "a.txt")))
```

Example

Plan found by Fast Downward

```
$ fast-downward.py prob1.pddl --search "astar(blind())"
$ cat sas_plan
(add "a.txt")
(commit )
; cost = 2 (unit cost)
```

Example

Plan found by Fast Downward

```
$ fast-downward.py prob1.pddl --search "astar(blind())"
$ cat sas_plan
(add "a.txt")
(commit )
; cost = 2 (unit cost)
```

Heuristic	Admissible?	Syntax
Additive	No	`add()`
Blind	Yes	`blind()`
Context-enhanced additive	No	`cea()`
Causal graph	No	`cg()`
Constant	No	`const()`
FF	No	`ff()`
Goal count	No	`goalcount()`
Max	Yes	`hmax()`

Example

Setup

```
mkdir prob4
cd prob4
git init
echo "aaa" > a.txt
echo "bbb" > b.txt
git add a.txt
git add b.txt
git commit -m "msg...”
echo "xxx" > a.txt
rm b.txt
```

Manual solution

```
git add a.txt
git commit –m "msg..."
git add b.txt
```

Problem file

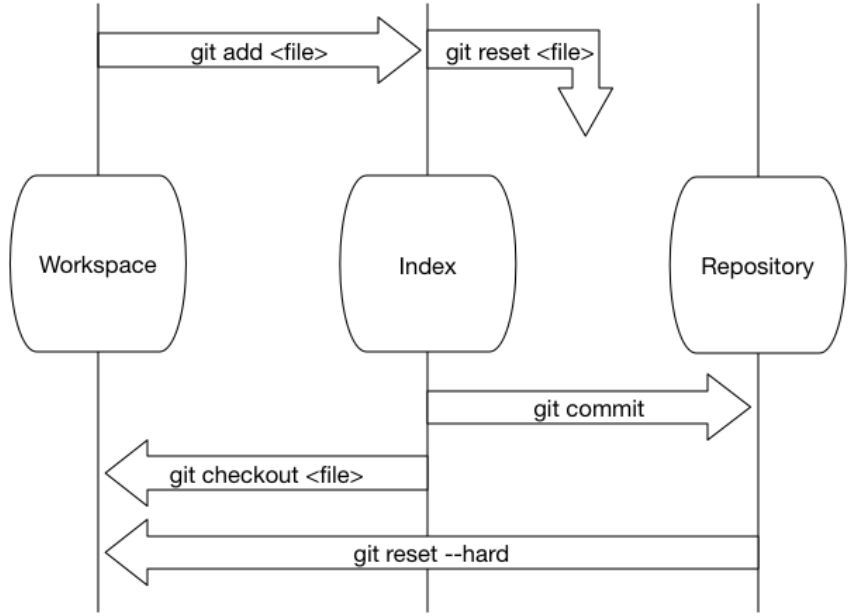
```
(define (problem p)
  (:domain git)
  (:objects "a.txt" "b.txt")
  (:init
    (modified-in-workspace "a.txt")
    (deleted-in-workspace "b.txt"))
  (:goal
    (and
      (deleted-from-index "b.txt")
      (committed "a.txt"))))
```

Goal

a.txt changes are committed
b.txt deletion is recorded in index but not committed

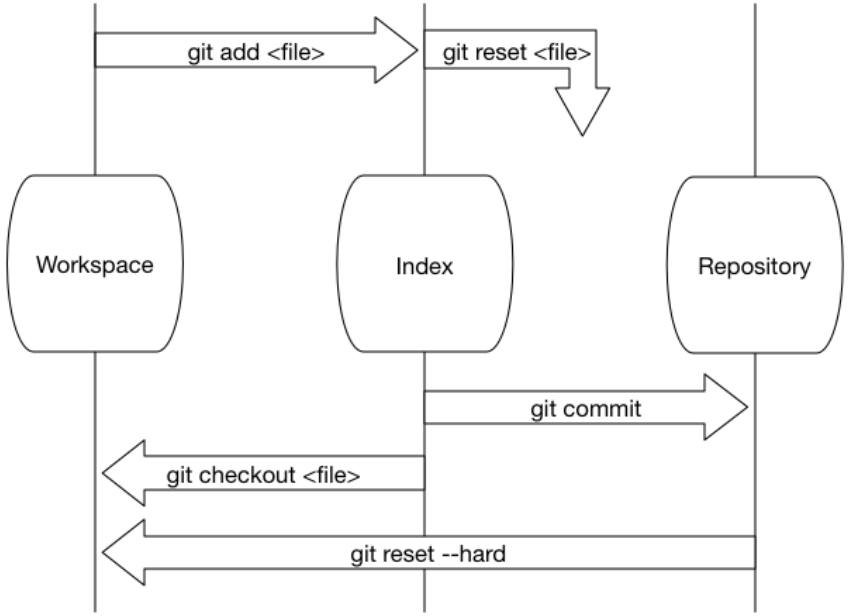
Partial solution: “reset” action

```
(:action reset
  :parameters (?f)
  :precondition (or (added-to-index ?f)
                    (updated-in-index ?f)
                    (deleted-from-index ?f))
  :effect (and (when (added-to-index ?f)
                 (and (not (added-to-index ?f))
                      (untracked ?f)))
               (when (updated-in-index ?f)
                 (and (not (updated-in-index ?f))
                      (modified-in-workspace ?f)))
               (when (deleted-from-index ?f)
                 (and (not (deleted-from-index ?f))
                      (deleted-in-workspace ?f)))))
```



Partial solution: “reset --hard” action

```
(:action reset-hard
  :parameters ()
  :precondition (exists (?f)
                  (and (not (untracked ?f))))
  :effect (forall (?f)
            (when (and (not (untracked ?f))
                       (added-to-index ?f)
                       (not (committed ?f)))
                  (and (not (added-to-index ?f))
                       (not (modified-in-workspace ?f))
                       (not (deleted-in-workspace ?f))
                       (not (updated-in-index ?f))
                       (not (deleted-from-index ?f))))))
```



Experience in the classroom

7 of 11 students completed it

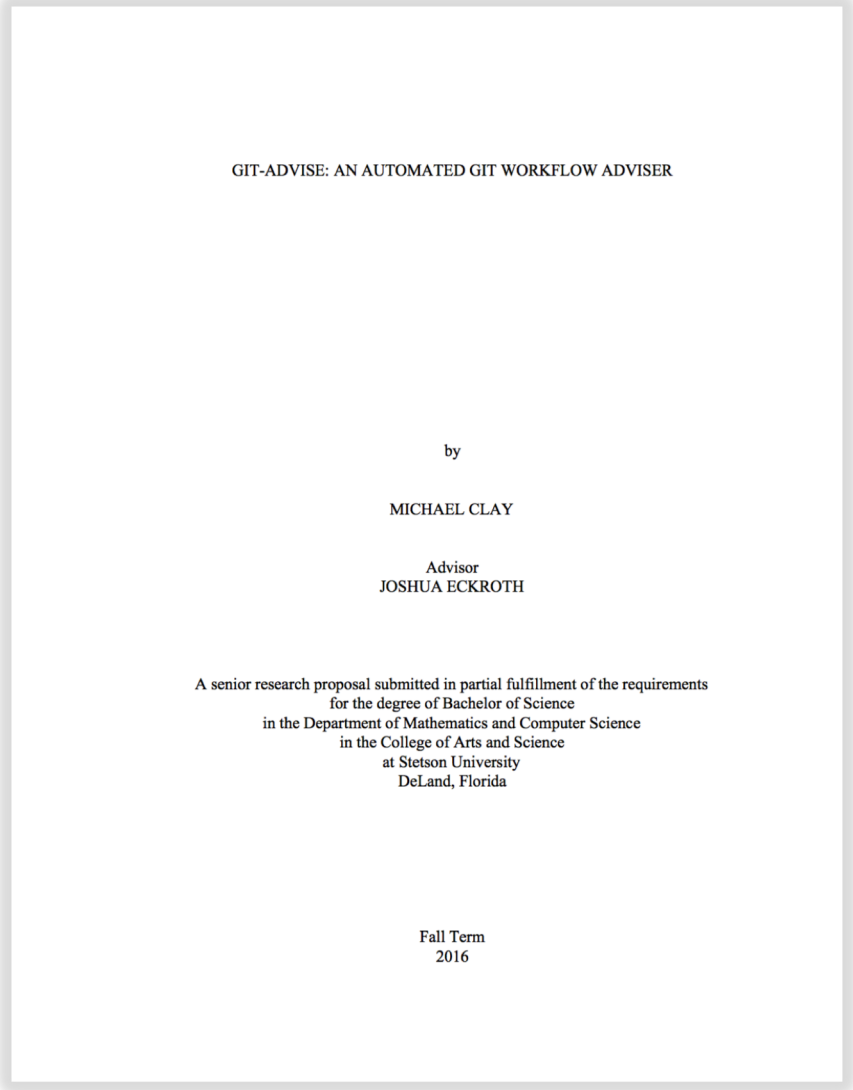
One student attempted to use OR in an effect

Three students did not solve examples 4 & 5

Fast Downward has terrible error messages (like all planning engines)

Git-Advise: An Automated Git Workflow Advisor

Michael Clay



GIT-ADVISE: AN AUTOMATED GIT WORKFLOW ADVISER

by

MICHAEL CLAY

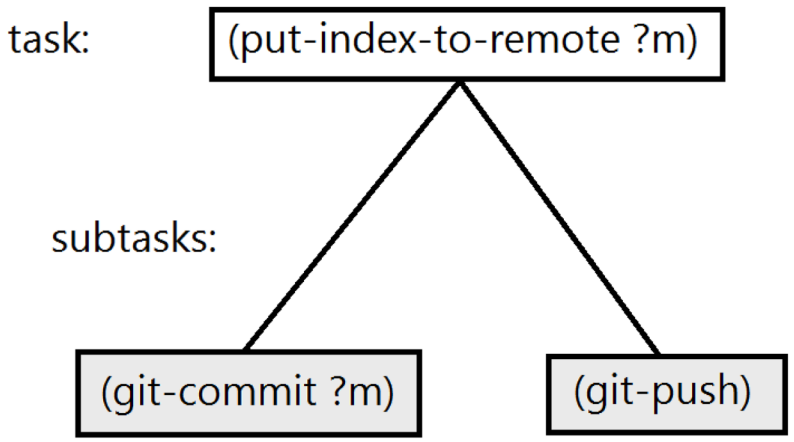
Advisor
JOSHUA ECKROTH

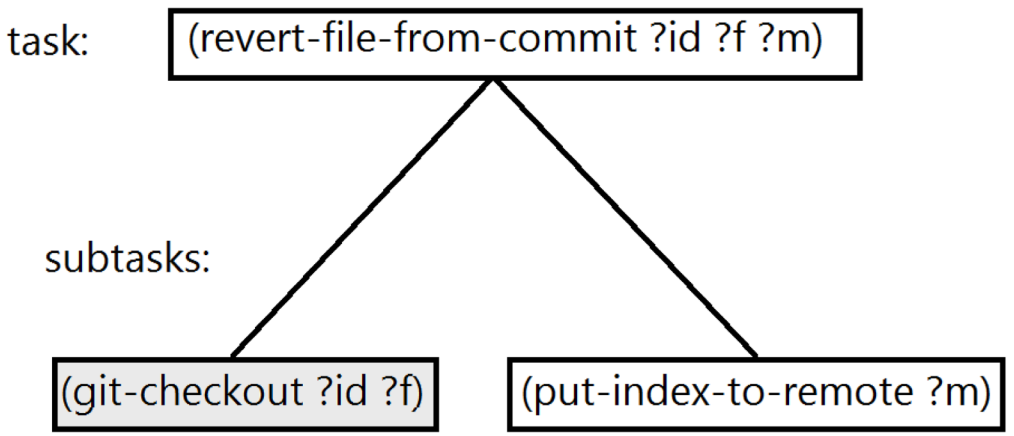
A senior research proposal submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science
in the Department of Mathematics and Computer Science
in the College of Arts and Science
at Stetson University
DeLand, Florida

Fall Term
2016

Git-Advise: An Automated Git Workflow Advisor

Michael Clay





Questions?