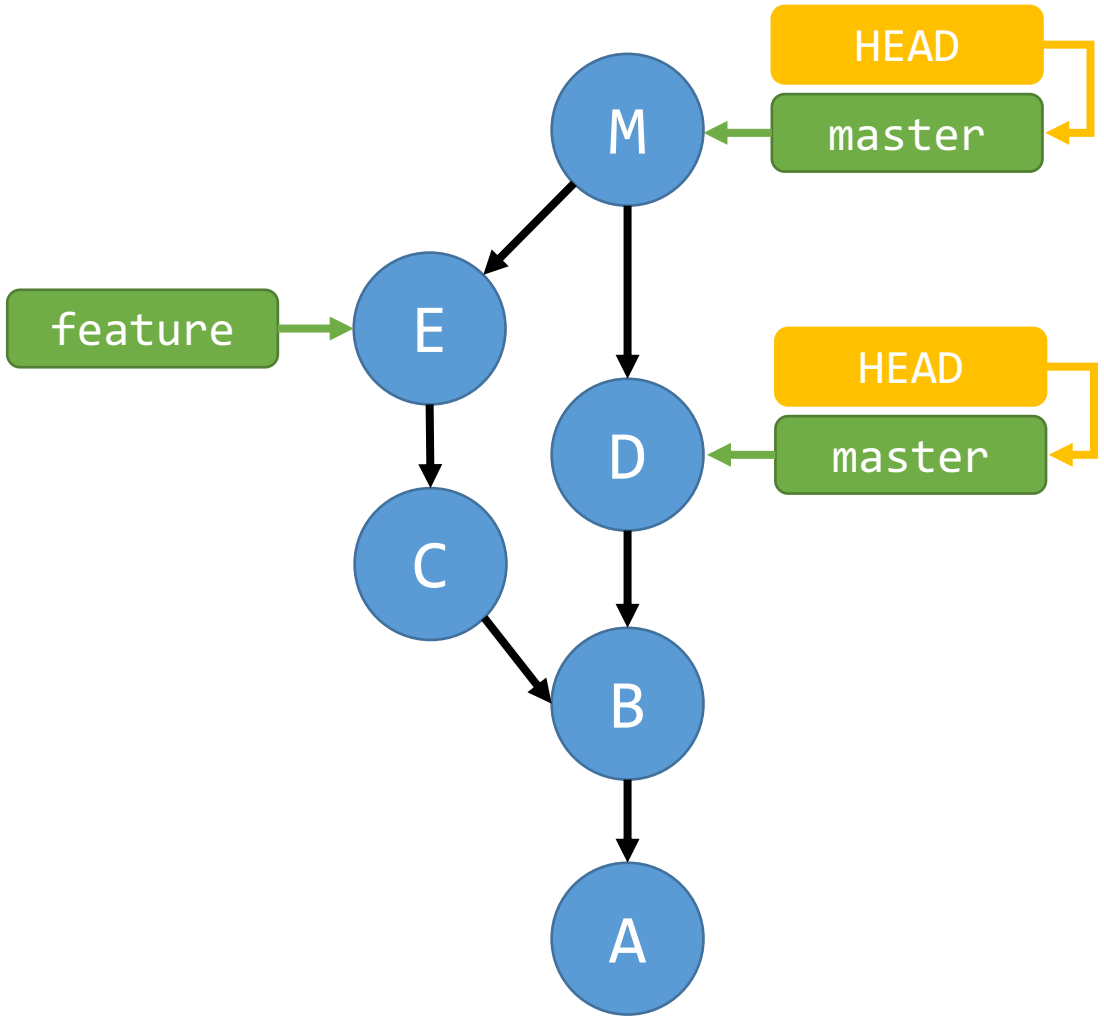
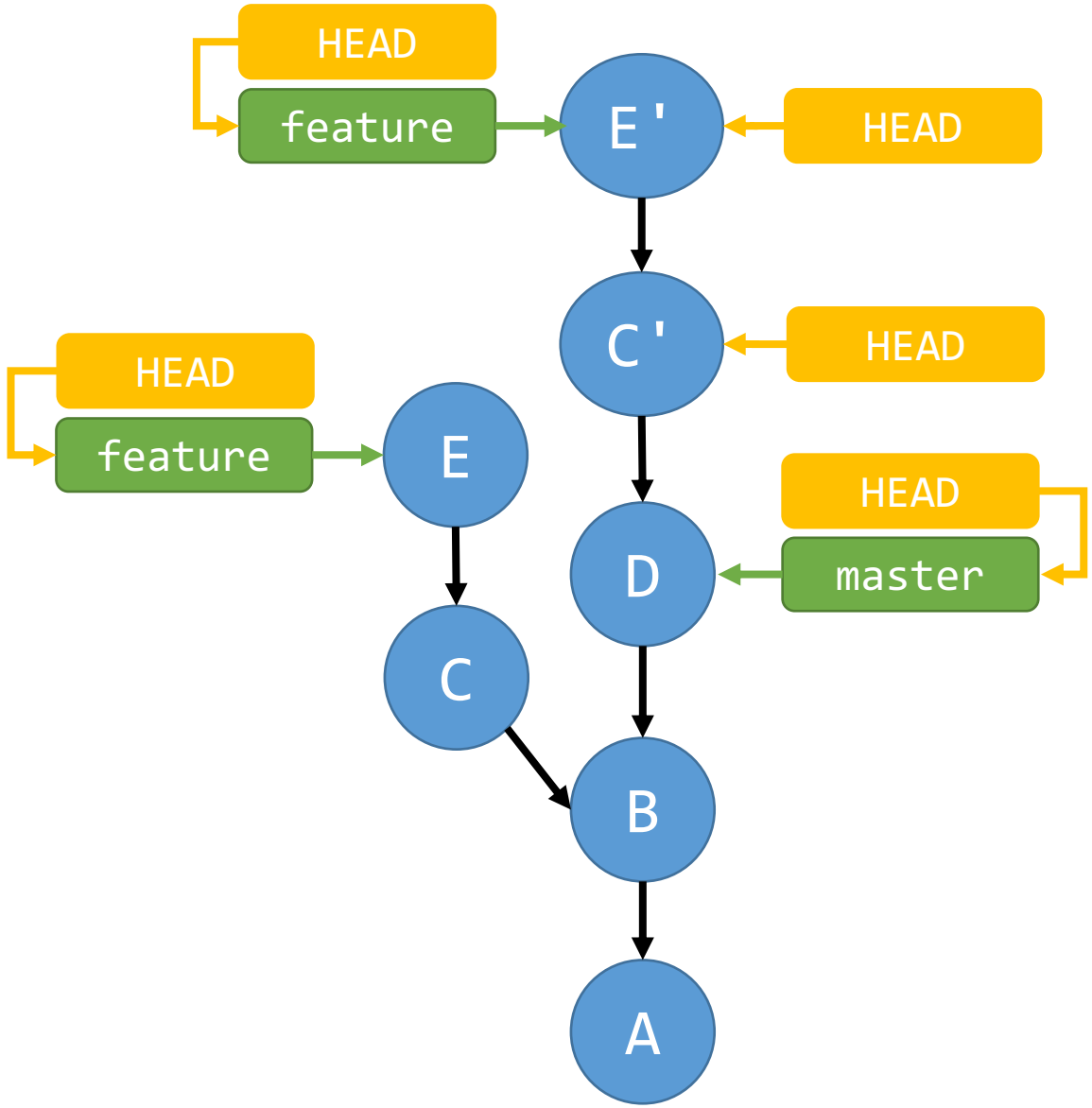
Lecture 9
Interactive Rebasing

Merge vs. Rebase



git merge feature



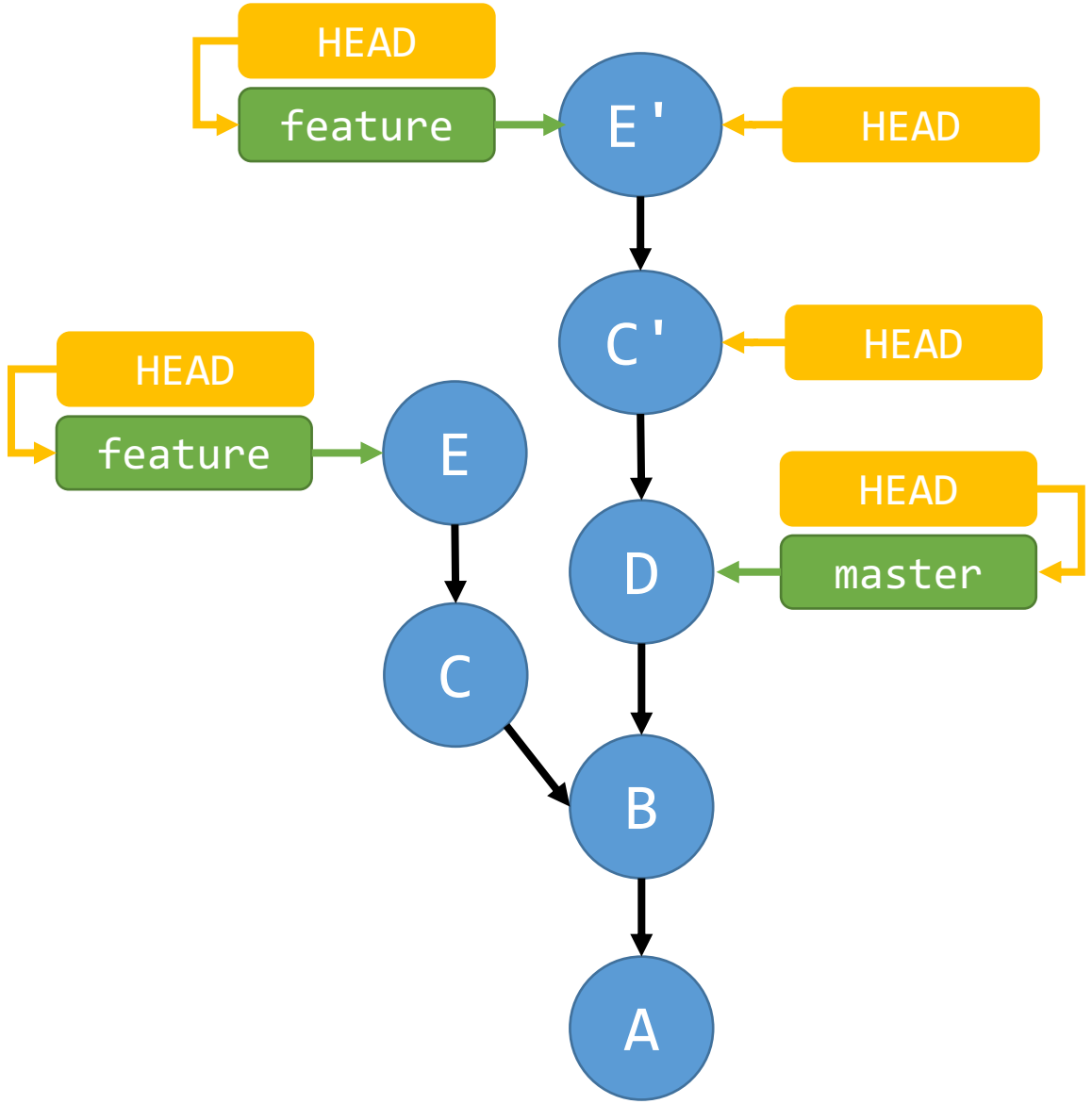
git rebase master

Today

- More powerful rebasing

git rebase master

1. Move HEAD to merge base of current branch and feature
2. Calculate diff between D and C and apply to D, creating C’
3. Calculate diff between C’ and E and apply to C’, creating E’
4. Reset feature to E’



git rebase master

What about conflicts?

- Just like merge conflicts, there can be conflicts when applying the diffs
- Resolve them like normal and use git rebase --continue

Interactive Rebasing

- Rebasing is super powerful!
- We can reorder, edit, remove, or amend commits!

git rebase -i <commitish>

Rebasing onto a commit allows you to change all of the commits in the commit history back to that commit.

Interactive Rebasing

```
pick 084db37 Add simple epilogue
pick 759fb6f Add Table of Contents
pick a4d140c Fix typos

# Rebase 690e6cc..a4d140c onto 690e6cc (3 commands)
#
# Commands:
# p, pick = use commit
# r, reword = use commit, but edit the commit message
# e, edit = use commit, but stop for amending
# s, squash = use commit, but meld into previous commit
# f, fixup = like "squash", but discard this commit's log message
# x, exec = run command (the rest of the line) using shell
# d, drop = remove commit
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
~
~
```

Git Reset

- Moves branch pointers
- git reset --soft HEAD~
- git reset –hard HEAD~2

git commit --amend

- Add files or edit the commit message of the most recent commit

Force Pushing

- Rebasing, resetting, and amending commits all move branch pointers to different commits
- Require force pushing, since the changes are not fast-forwards

Activity/Homework

Fork and clone https://github.com/ilanbiala/mergesort. Rebase all of the commits on master into better organized and named commits by using reword, squash, and fixup.

You should have between 2-3 commits at the end. Push them to your fork and submit a pull request.