pijul – post-git SCM

https://lukas-prokop.at/talks/glt25-pijul

meisterluk
2025-04-26
Lightning Talk, GLT25

the epoch of git

(darcs)

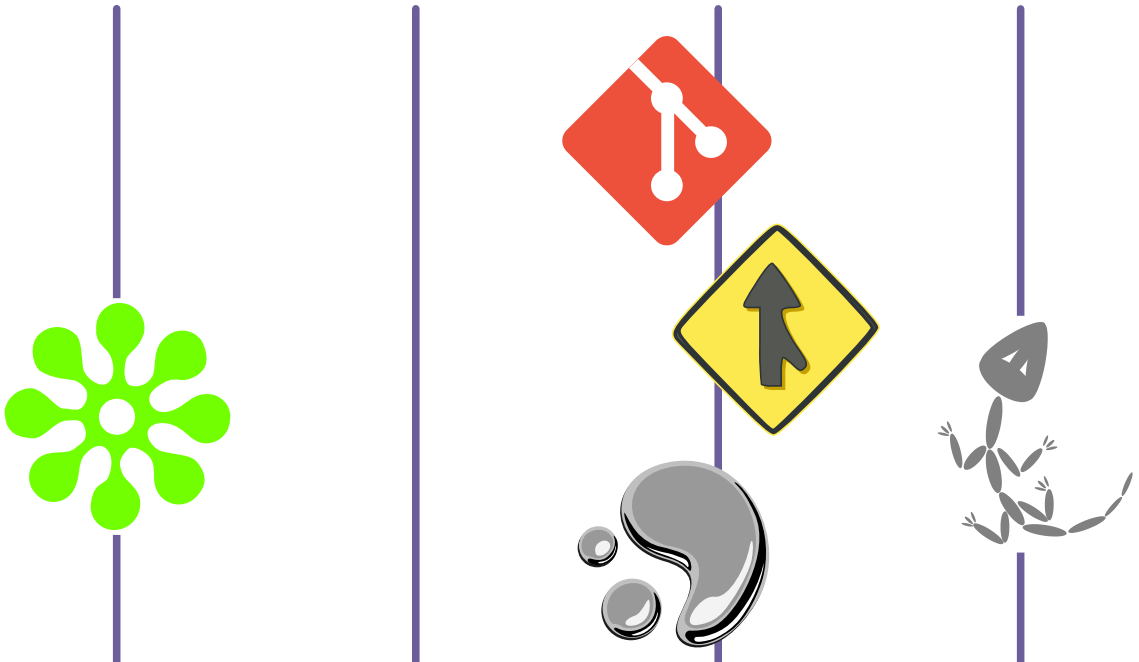
2003

2004

2005

(Fossil)

2006



(mercurial, bazaar, git)

the epoch of git

(darcs)

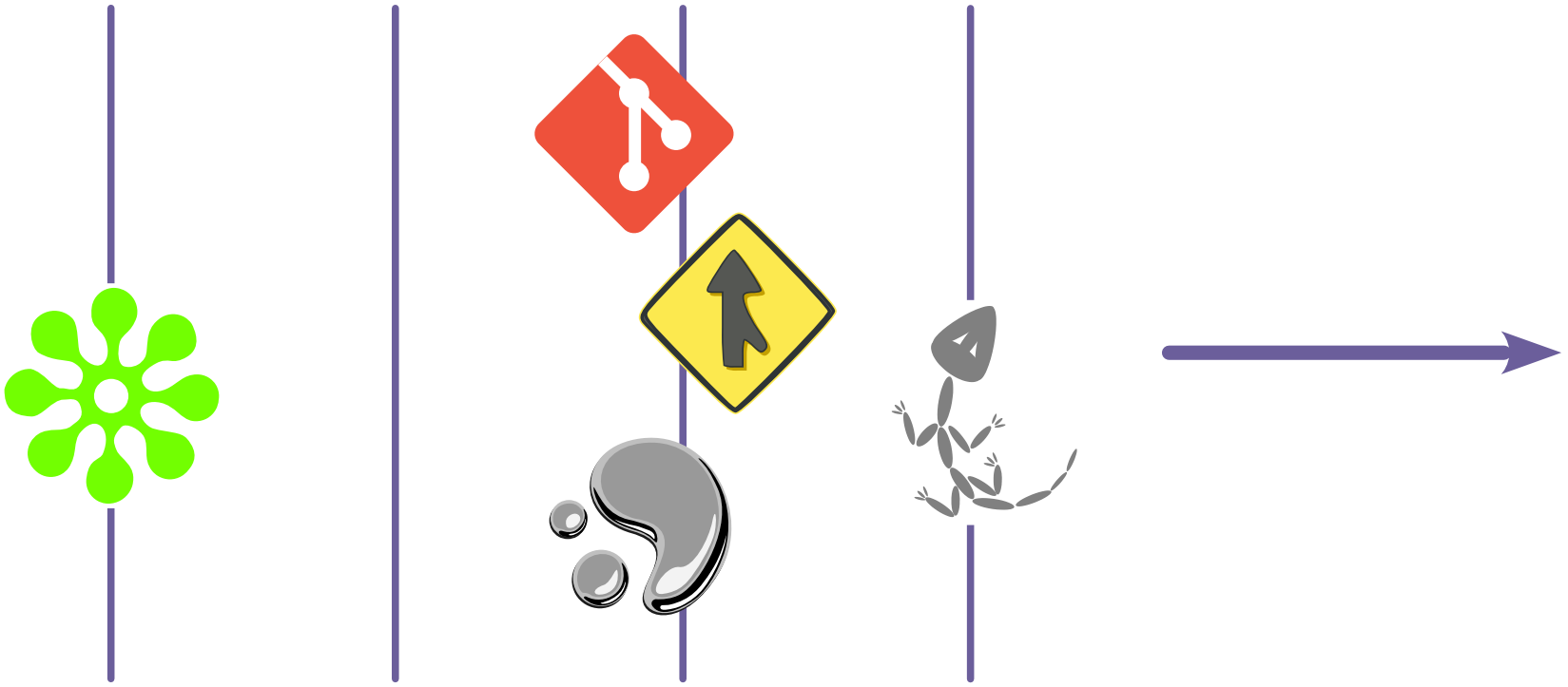
2003

2004

2005

(Fossil)

2006



(mercurial, bazaar, git)

the epoch of git

(darcs)

2003

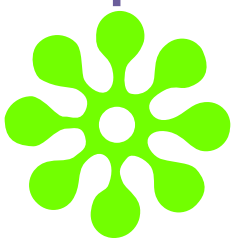
2004

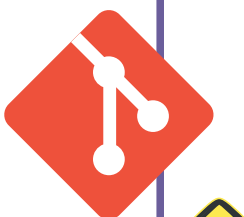
2005

(Fossil)

2006

2016





(mercurial, bazaar, git)

Pijul

git applied

```
$ git init
Initialized empty Git repository in /tmp/gitex/.git/

$ echo "In which branch am I? main" > text.txt

$ git add text.txt

$ git commit -m 'initial commit'
[main (root-commit) 51337ac] initial commit
 1 file changed, 1 insertion(+)
 create mode 100644 text.txt

$ git branch feature

$ echo "In which branch am I? feature" > text.txt

$ git commit -m 'feature branch'
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   text.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

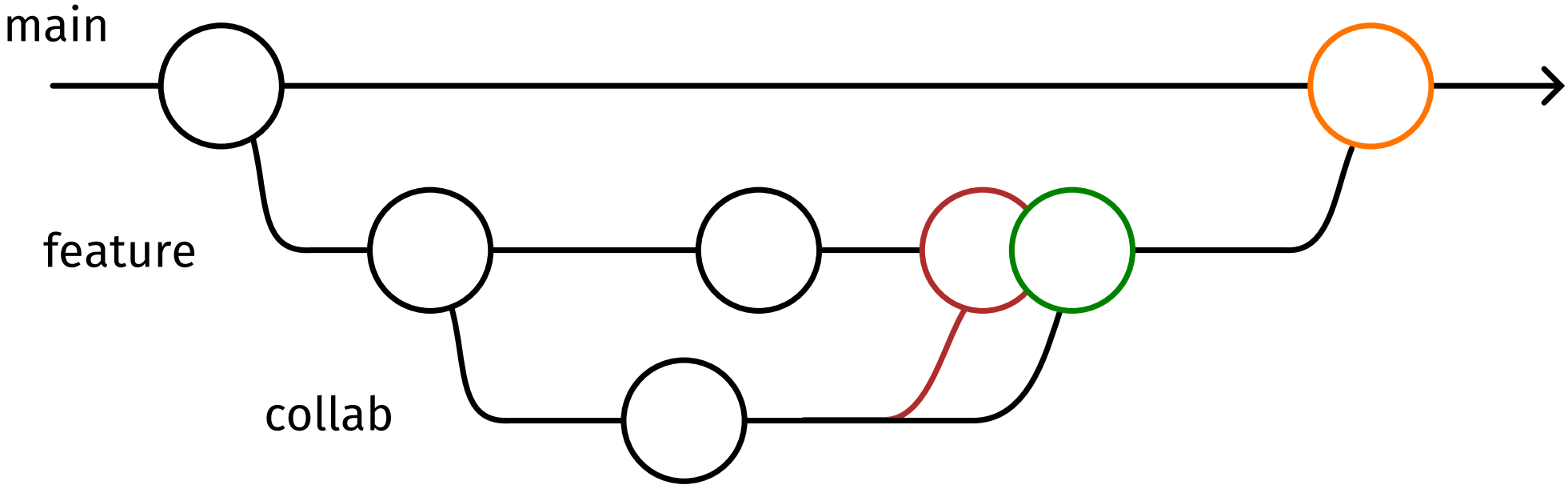
git applied

```
$ git commit -a -m 'feature branch'
[main 2ce74b7] feature branch
 1 file changed, 1 insertion(+), 1 deletion(-)

$ git branch collab

$ echo "In which branch am I? collab" > text.txt

$ git commit -a -m 'collab branch'
[main ed81b85] collab branch
 1 file changed, 1 insertion(+), 1 deletion(-)

$ git switch feature
Switched to branch 'feature'

$ echo "In which branch am I? feature. I swear!" > text.txt

$ git commit -a -m 'feature branch for sure'
[feature b256b5b] feature branch for sure
 1 file changed, 1 insertion(+), 1 deletion(-)

$ git merge collab
Auto-merging text.txt
CONFLICT (content): Merge conflict in text.txt
Automatic merge failed; fix conflicts and then commit the result.

$ echo "In which branch am I? some-merged-branch" > text.txt
```

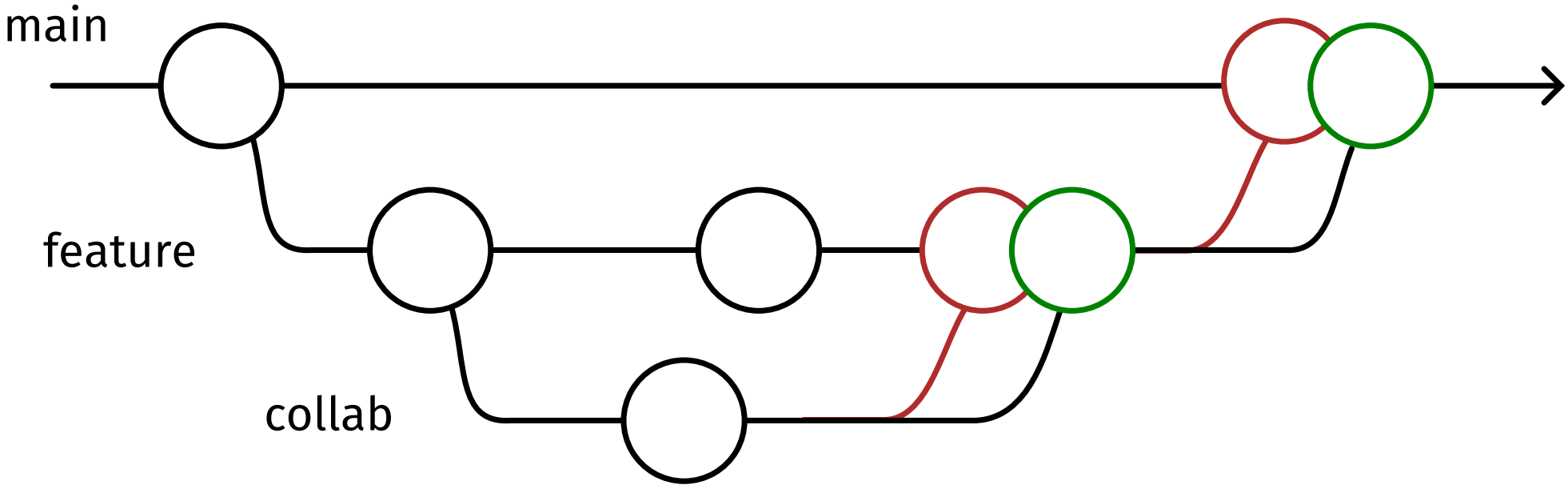
git applied

```
$ git commit -a -m 'feature and collab merged'
[feature 3a6d2ea] feature and collab merged

$ git switch main
Switched to branch 'main'

$ git merge feature
Auto-merging text.txt
CONFLICT (content): Merge conflict in text.txt
Automatic merge failed; fix conflicts and then commit the result.

$ git log --graph --oneline --all
*   3a6d2ea (feature) feature and collab merged
|\
* | b256b5b feature branch for sure
| | * ed81b85 (HEAD -> main) collab branch
| |/
| * 2ce74b7 (collab) feature branch
|/
* 51337ac initial commit
```

git visualized



git visualized



7.9 Git Tools - Rerere

Rerere

The `git rerere` functionality is a bit of a hidden feature. The name stands for "reuse recorded resolution" and, as the name implies, it allows you to ask Git to remember how you've resolved a hunk conflict so that the next time it sees the same conflict, Git can resolve it for you automatically.

There are a number of scenarios in which this functionality might be really handy. One of the examples that is mentioned in the documentation is when you want to make sure a long-lived topic branch will ultimately merge cleanly, but you don't want to have a bunch of intermediate merge commits cluttering up your commit history. With `rerere` enabled, you can attempt the occasional merge, resolve the conflicts, then back out of the merge. If you do this continuously, then the final merge should be easy because `rerere` can just do everything for you automatically.

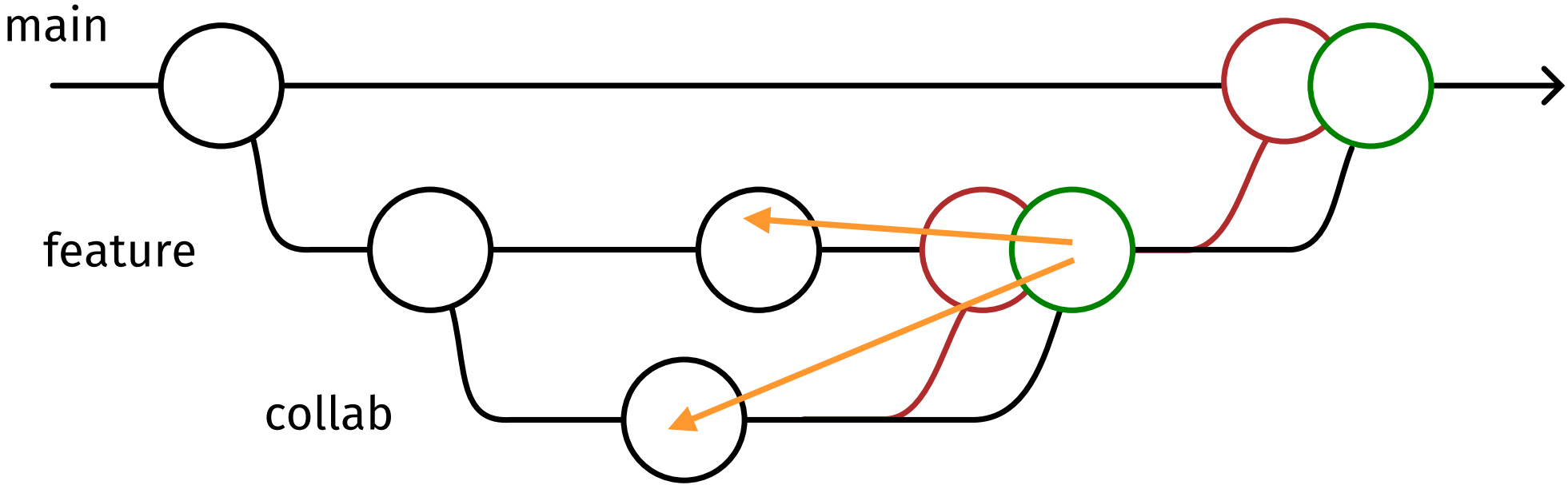
This same tactic can be used if you want to keep a branch rebased so you don't have to deal with the same rebasing conflicts each time you do it. Or if you want to take a branch that you merged and fixed a bunch of conflicts and then decide to rebase it instead — you likely won't have to do all the same conflicts again.

Another application of `rerere` is where you merge a bunch of evolving topic branches together into a testable head occasionally, as the Git project itself often does. If the tests fail, you can rewind the merges and re-do them without the topic branch that made the tests fail without having to re-resolve the conflicts again.

To enable `rerere` functionality, you simply have to run this config setting:

```
$ git config --global rerere.enabled true
```

git visualized



author, time, commit message, filepaths, file states, ...

https://stackoverflow.com/a/68806436

pijul

- developed by ...
  - Pierre Étienne Meunier
  - Florent Becker
- command-line tool written in rust
- built upon the theory of patches (like darcs)

theory of patches

- *commutativity:* independent patches commute

  $a \cdot b = b \cdot a$

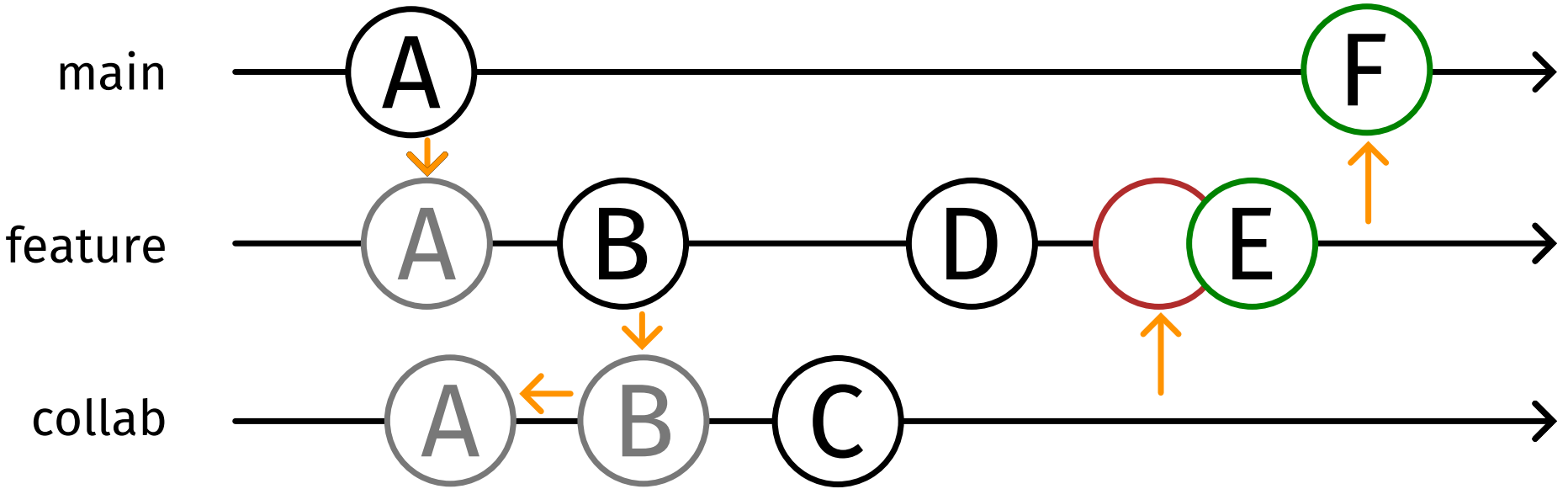
- *associativity:* repositories are sets of patches

  $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

- inversion: patches have a semantic inverse

based on: Pijul: Sane Version Control

pijul visualized



pijul applied

```
$ pijul init
Repository created at /tmp/pijul-test

$ echo "On which channel am I? main" > text.txt
$ pijul add text.txt
Tracked 1 path(s)

$ pijul record -a -m 'initial record'
Hash: KBNAAFTJ3OBGOV2TL4DDVO5DSWSJPSZT4VY6B6WECISYPCL74JOQC

$ pijul fork feature

$ pijul channel switch feature
Outputting repository... done!
Reset given paths to last recorded change

$ echo "On which channel am I? feature" > text.txt
$ pijul record -a -m 'record on feature'
Hash: 5URL4O5LX7D5Y3HOHY62HEOHD5WU4NFWSHFY6EPYZFQIYR7B4JOAC
```

pijul applied

```
$ pijul fork collab

$ pijul channel switch collab
Outputting repository... done!
Reset given paths to last recorded change

$ echo "On which channel am I? collab" > text.txt
$ pijul record -a -m 'record on collab'
Hash: DJD2OHJLWLV4NZSRBTHEEV4UDFXYHDPQLJAIYPOA4IXH746DZXPQC

$ pijul channel switch feature
Outputting repository... done!
Reset given paths to last recorded change

$ echo "On which channel am I? feature - I swear!" > text.txt
$ pijul record -a -m 'record on feature, for sure'
Hash: ZSF4ZYMUV5CDWVMGKXSHDWQDYN5BQRMR4X4HCI7EJWWJHDRBOA6QC

$ pijul apply DJD2OHJLWLV4NZSRBTHEEV4UDFXYHDPQLJAIYPOA4IXH746DZXPQC
There were conflicts:
  - Order conflict in "text.txt" starting on line 1
  - Order conflict in "text.txt" starting on line 1
Outputting repository... done!
```

pijul applied

```
$ pijul channel switch main
Outputting repository... done!
Reset given paths to last recorded change

$ pijul apply
RCCMQI3IX3ORSCOZ6VALKUX2XZHLDHNWHXU66FS5L4DPLOAGUDAAC
Outputting repository... done!
```

pijul summarized

- pijul has a simpler model & CLI than git
- pijul fixes performance problems of darcs
- no three-way merge algorithm per default
- no discussions like “merge or rebase workflow?”
- web interface for repositories? nest.

git ⟷ pijul

`git init`	`pijul init`
`git branch`	`pijul branch`
`git switch`	`pijul branch switch`
`git commit`	`pijul record`
`git blame`	`pijul credit`
`git remote`	`pijul remote`
`git push`	`pijul push`
`git pull`	`pijul pull`

git ⟷ pijul

`git init`	`pijul init`
`git branch`	`pijul branch`
`git switch`	`pijul branch switch`
`git commit`	`pijul record`
`git blame`	**`pijul credit`**
`git remote`	`pijul remote`
`git push`	`pijul push`
`git pull`	`pijul pull`

crotophaga sulcirostris

a bird known to do collaborative nest building

https://pijul.org/

¡Gracias por su atención!