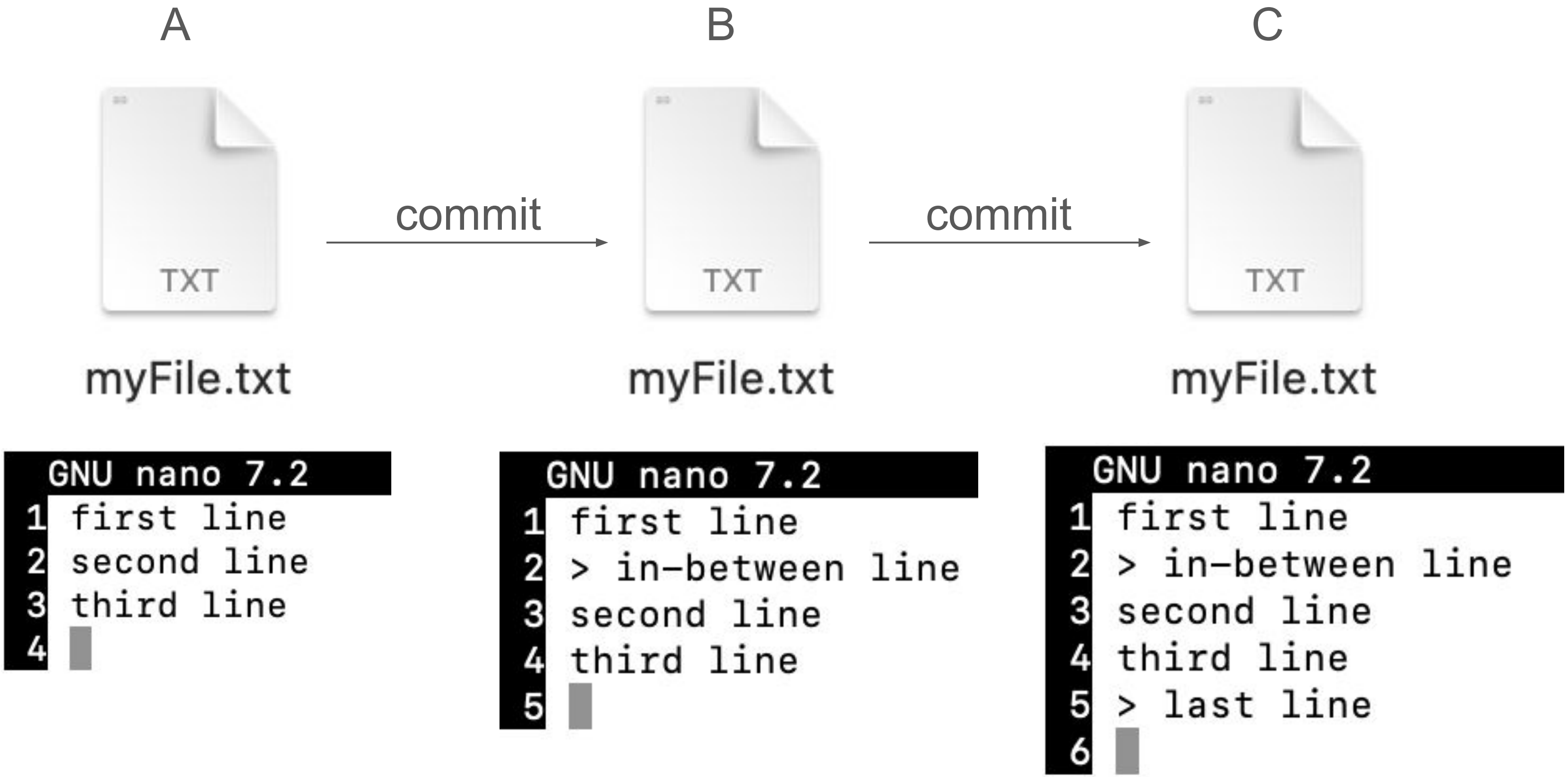
Categorical Version Control

Dylan Wallace

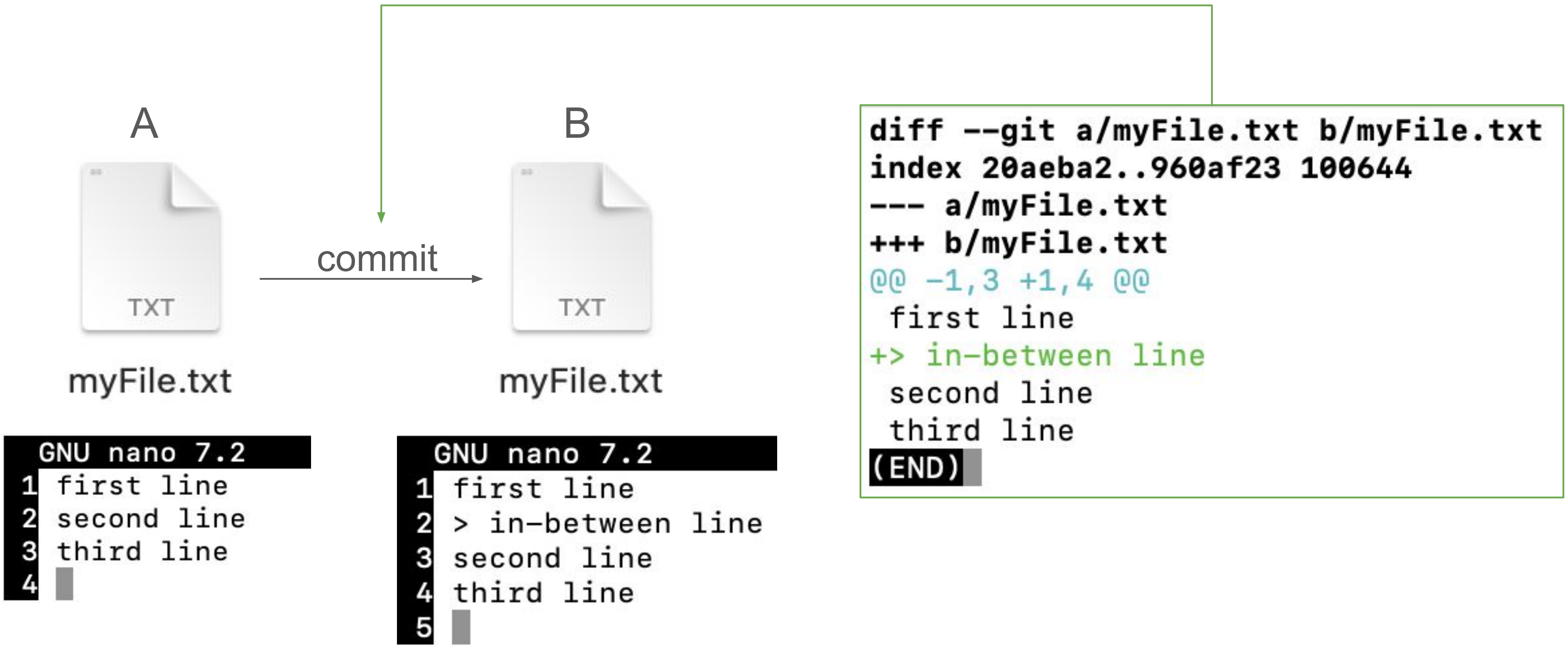
March 21, 2025

What is version control?

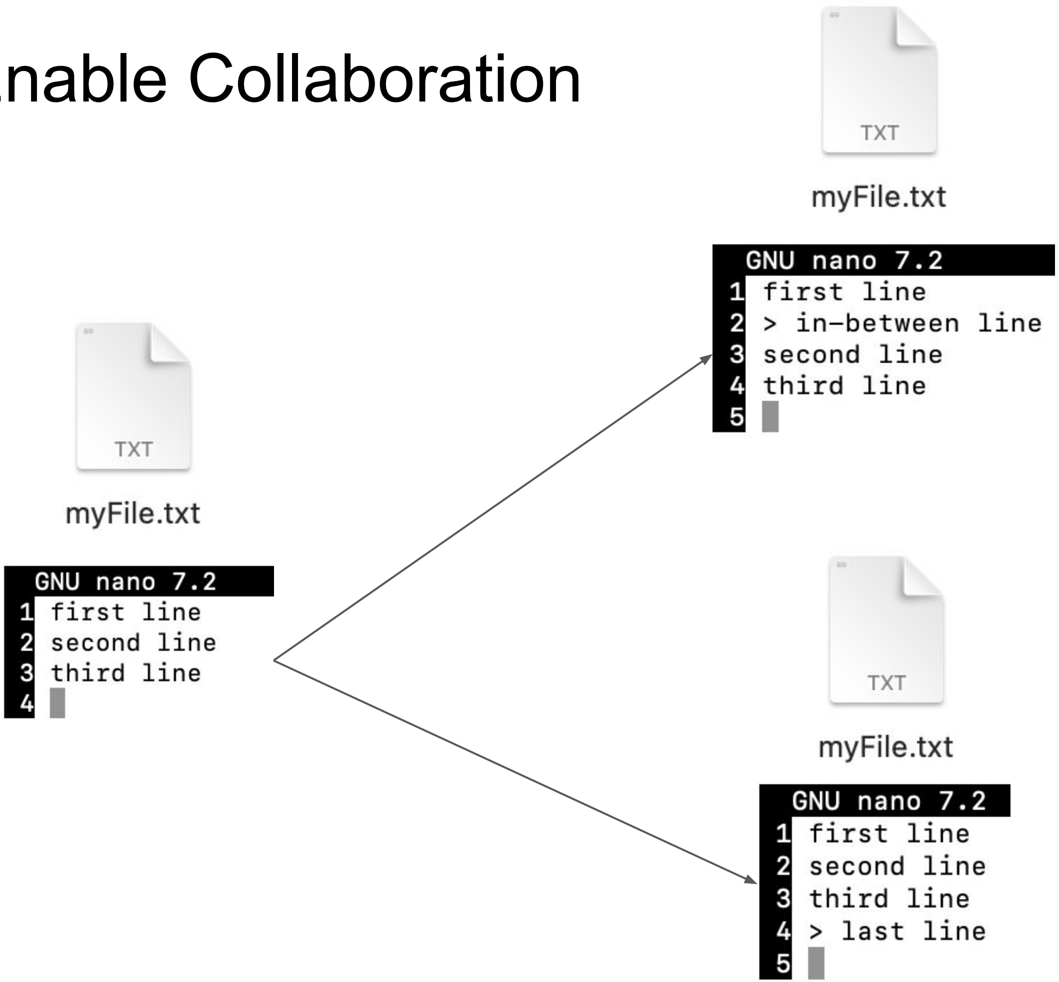
Keep track of changes



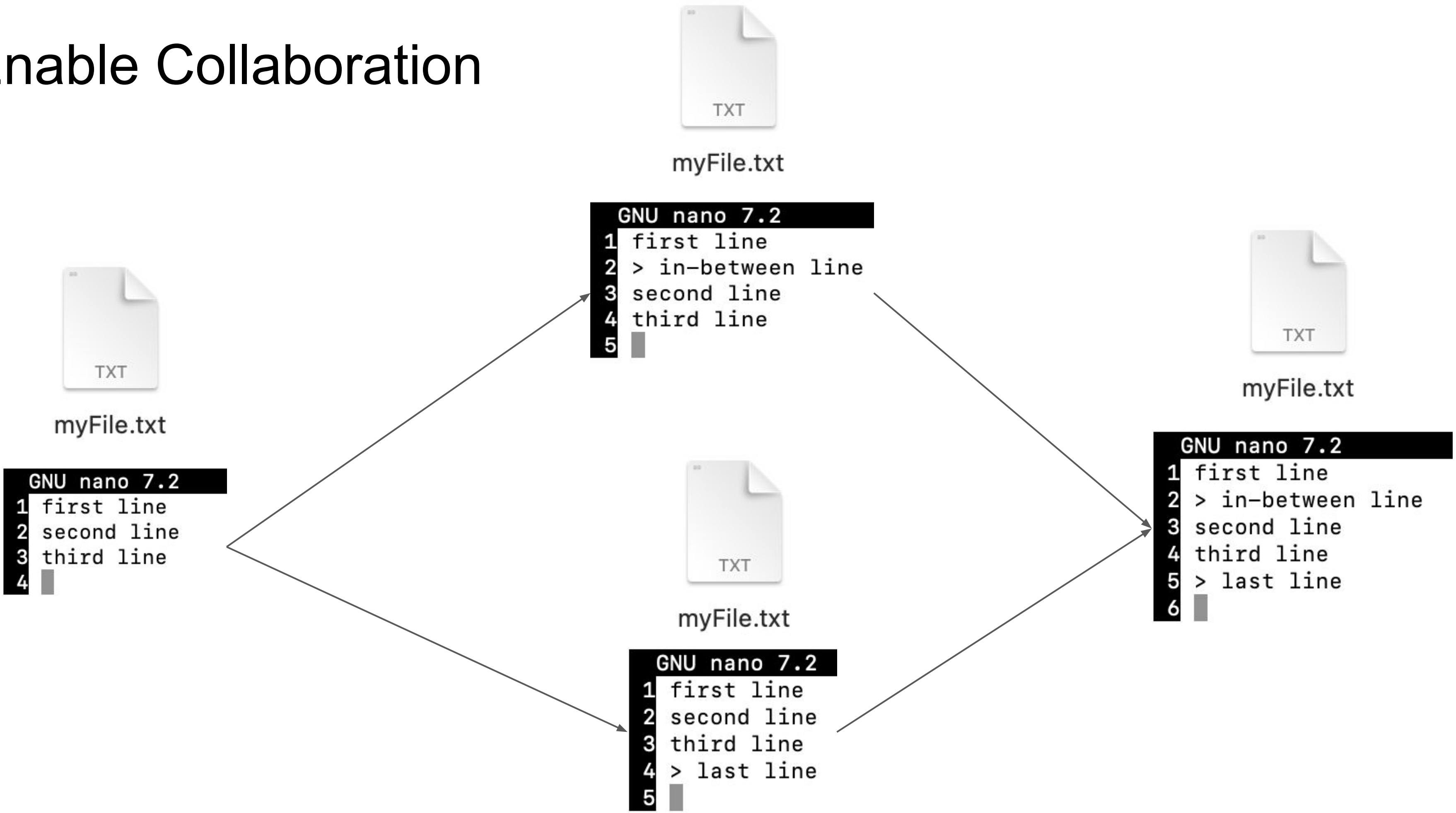
Keep track of changes



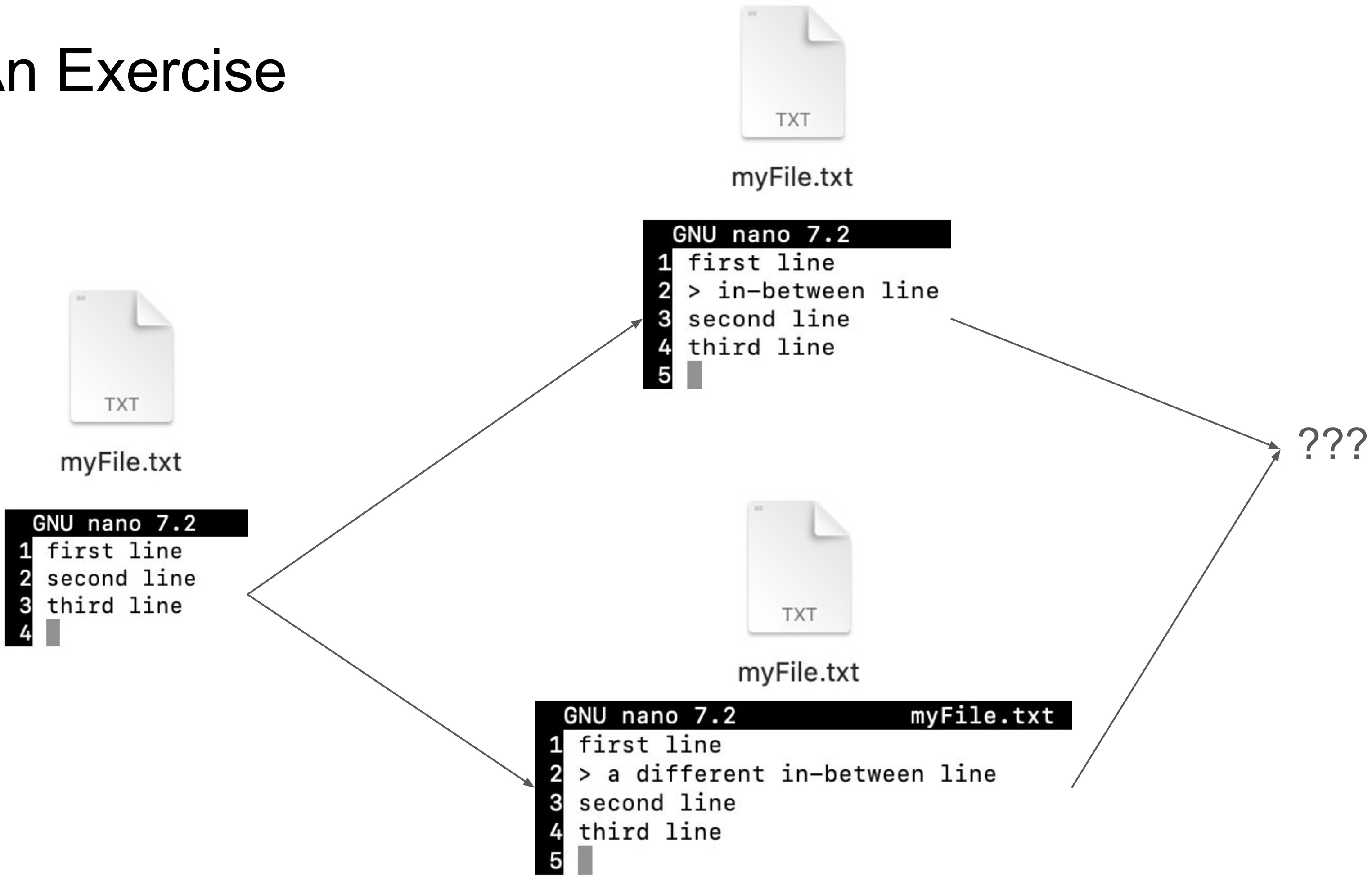
Enable Collaboration



Enable Collaboration



An Exercise


TXT
myFile.txt
GNU nano 7.2
1 first line
2 second line
3 third line
4
TXT
myFile.txt
GNU nano 7.2
1 first line
2 > in-between line
3 second line
4 third line
5
TXT
myFile.txt
GNU nano 7.2 myFile.txt
1 first line
2 > a different in-between line
3 second line
4 third line
5
???

An Exercise

```
→ (/dev/s001) <dwall@dylans-mbp-5.lan> presentation git:(master) git merge my-branch
Auto-merging myFile.txt
CONFLICT (content): Merge conflict in myFile.txt
Automatic merge failed; fix conflicts and then commit the result.
```

```
> in-between line
second line
third line
```

myFile.txt

```
GNU nano 7.2
first line
second line
third line

```

```
GNU nano 7.2                    myFile.txt
first line
<<<<<<< HEAD
> in-between line
=======
> a different in-between line
>>>>>>> my-branch
second line
third line

```

???

Search Results

Advanced Search Tips Ask Question

Results for merge conflict
Search options **not deleted**

500 results

Relevance Newest More

5424 votes
36 answers
3.7m views

How do I resolve merge conflicts in a Git repository?

How do I resolve **merge** conflicts in my Git repository? ...

git git-merge merge-conflict-resolution git-merge-conflict

Spoike **122k** asked Oct 2, 2008 at 11:31

3243 votes
14 answers
2.7m views

I ran into a merge conflict. How do I abort the merge?

I used git pull and had a **merge conflict**: unmerged: some_file.txt You are in the middle of a conflicted **merge**. How do I abandon my changes to the file and keep only the pulled changes? ...

git version-control git-merge git-merge-conflict

Gwyn Morfey **33.6k** asked Sep 19, 2008 at 13:21

13501 votes
Accepted

How do I force "git pull" to overwrite local files?

main): git branch backup-main Jump to the latest commit on origin/main and checkout those files: git reset --hard origin/main Explanation: git fetch downloads the latest from remote without tryin...

git version-control overwrite git-pull git-fetch

RNA **154k** answered Jan 17, 2012 at 0:02

18940 votes
Accepted

How to modify existing, unpushed commit messages?

Git will "collect" all the commits in the last n commits, and if there was a **merge** somewhere in between that range you will see all the commits as well, so the outcome will be n + ...

git git-commit git-rewrite-history git-amend

Community wiki EfForEffort

Can we design a system
where *every* merge succeeds?

Goals:

1. "Mathematicize" the notion of files and diffs/patches
2. Express our version control system in the language of Category Theory
3. Use results in Category Theory to make statements about our system (which will hopefully solve our issue!)

Basic Category Theory

Definition: Category

A **category** $\mathcal{C}$ is an algebraic structure consisting of a **collection of objects** $\mathrm{Ob}(\mathcal{C})$ and for every pair of objects $A, B \in \mathrm{Ob}(\mathcal{C})$, a **set of morphisms** $\mathrm{Hom}(A, B)$ such that the following conditions hold:

1. For any $f \in \mathrm{Hom}(A, B)$, $g \in \mathrm{Hom}(B, C)$, $\exists g \circ f \in \mathrm{Hom}(A, C)$ (*composition*)
2. $\mathrm{id}_X \in \mathrm{Hom}(X, X)$ that acts like the identity for all $X \in \mathrm{Ob}(\mathcal{C})$.

Examples:

1. The category of sets Set, where elements of $\mathrm{Ob}(\mathrm{Set})$ are sets and morphisms are functions between sets
2. The category of vector spaces $\mathrm{Vect}_{\mathbb{F}}$ over a field $\mathbb{F}$, with morphisms linear transformations between vector spaces

and there are many more

Now that we have categories, we want to define relations between them:

Definition: Functor

Given two categories $\mathcal{C}$ and $\mathcal{D}$, a **functor** $F : \mathcal{C} \to \mathcal{D}$ is a mapping consisting of

1. a function on objects $F : \mathrm{Ob}(\mathcal{C}) \to \mathrm{Ob}(\mathcal{D})$
2. for any two $X, Y \in \mathrm{Ob}(\mathcal{C})$, a function on morphisms $F : \mathrm{Hom}(X, Y) \to \mathrm{Hom}(F(X), F(Y))$

such that the following hold:

1. $F(\mathrm{id}_X) = \mathrm{id}_{F(X)} \in \mathrm{Hom}(F(X), F(X))$
2. For $f \in \mathrm{Hom}(X, Y)$ and $g \in \mathrm{Hom}(Y, Z)$ for $X, Y, Z \in \mathrm{Ob}(\mathcal{C})$, we have $F(g \circ f) = F(g) \circ F(f)$

From here on out we'll just say $X \in \mathcal{C}$ instead of $X \in \mathrm{Ob}(\mathcal{C})$ and $f : X \to Y$ instead of $f \in \mathrm{Hom}(X, Y)$ whenever it isn't ambiguous ☺

Limits: Examples

Products and Coproducts:

$$\begin{tikzcd}
& X \arrow[d, dashed] \arrow[ddl, bend right] \arrow[ddr, bend left] & \\
& \{1,2,3\}\times\{4,5,6\} \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & \\
\{1,2,3\} & & \{4,5,6\}
\end{tikzcd}
\qquad
\begin{tikzcd}
\{1,2,3\} \arrow[dr, "i_1"'] \arrow[ddr, bend right] & & \{4, \arrow[dl, "i_2"] \arrow[ddl, bend left] \\
& \{1,2,3,4,5,6\} \arrow[d, dashed, "f"] & \\
& X &
\end{tikzcd}$$

We want to generalize this notion to all "subsets" of a category, not just those with two elements and (and no morphisms)

Example category $\mathcal{C}$:

$$
\begin{tikzcd}
X \arrow[r, "f"] \arrow[dr, "h"] & Y \arrow[r, "g"] \arrow[d, "j"] \arrow[dr, "k"] & Z \\
& V \arrow[r, "l"] & W
\end{tikzcd}
$$

with $j \circ f = h$ and $l \circ j = k$.

Then we might want to consider a "subset" of the category, called a diagram:

$$
\begin{tikzcd}
X \arrow[dr, "h"] & & \\
& V \arrow[r, "l"] & W
\end{tikzcd}
$$

We will formalize this notion

Limits

Consider a category J with the following elements:

$$A \xrightarrow{\alpha} B \xrightarrow{\beta} C$$

then define a functor $D : J \to \mathcal{C}$ with $D(A) = X, D(B) = V, D(C) = W$, $D(\alpha) = h, D(\beta) = l$ so that the "image" of D is our diagram from before. Formally, we call the functor D itself the **diagram**.

$$\begin{tikzcd}
A \arrow[rr, "\alpha"] & {} \arrow[d, Rightarrow, "D"] & B \arrow[rr, "\beta"] & {} \arrow[d, Rightarrow, "D"] & C \\
X \arrow[rr, "h"'] & {} & V \arrow[rr, "l"'] & {} & W
\end{tikzcd}$$

Limits

For a given diagram, consider all the possible "cones":

$$\begin{tikzcd} X \arrow[r, "h"] \arrow[dr, "\psi_{XA}"'] & V \arrow[r, "l"] \arrow[d, "\psi_{VA}"] & W \arrow[dl, "\psi_{WA}"] \\ & A & \end{tikzcd} \qquad \begin{tikzcd} X \arrow[r, "h"] \arrow[dr, "\psi_{XB}"'] & V \arrow[r, "l"] \arrow[d, "\psi_{VB}"] & W \arrow[dl, "\psi_{WB}"] \\ & B & \end{tikzcd}$$

etc...

where the induced diagram commutes (e.g. $\psi_{VA} \circ h = \psi_{XA}$). Technically these are called cocones (like cones but in the reverse direction). For shorthand, we'll refer to our diagram as D and write our cocones as $\psi_A : D \to A$ or $\psi_B : D \to B$. (BTW THIS IS NOT RIGOROUS AT ALL) Likewise, we can imagine a cone $\phi_A : A \to D$ corresponding to

$$\begin{tikzcd} & A \arrow[dl, "\phi_{AX}"'] \arrow[d, "\phi_{AV}"] \arrow[dr, "\phi_{AW}"] & \\ X \arrow[r, "h"] & V \arrow[r, "l"] & W \end{tikzcd}$$

Limits

Definition: (Co)Limit

For a given diagram D, a cone $\phi_X : X \to D$ is called the **limit** of D if for any other cone ϕ_Y, there exists a morphism $f : Y \to X$ such that $\phi_{YI} = \phi_{XI} \circ f$.
Likewise, a cocone $\psi_X : D \to X$ is called the **colimit** of D if for any other cocone ψ_Y, there exists a *unique* morphism $g : X \to Y$ such that $\psi_{IY} = g \circ \psi_{IX}$ for all objects $I \in D$.
Usually we just refer to the object itself as the limit or colimit instead of the diagram as a whole.

$$
\begin{tikzcd}
 & B \arrow[d, dashed, "f"'] \arrow[ddl, bend right] \arrow[ddr, bend left] & \\
 & A \arrow[dl] \arrow[d] \arrow[dr] & \\
X \arrow[r, "h"] & V \arrow[r, "l"] & W
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
X \arrow[r, "h"] \arrow[dr] \arrow[ddr, bend right] & V \arrow[r, "l"] \arrow[d] & W \arrow[dl] \arrow[ddl, bend left] \\
 & A \arrow[d, dashed, "g"] & \\
 & B &
\end{tikzcd}
$$

The idea is that limits and colimits capture the nature of the diagram D with respect to morphisms into D, or morphisms out of D. If every diagram in a category has a (co)limit, it is called **(co)complete**.

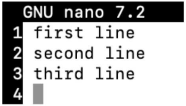
Version Control Category

Denote $[n] = \{1, \ldots, n\}$, and $\mathrm{Strings} = \{$All possible arrangements of characters in an alphabet$\}$. Then we can define a file:

Definition: File

A **file** is a function $F : [n] \to \mathrm{Strings}$.

e.g.



$\rightarrow$

$F : [3] \to \mathrm{Strings}$ defined by:

$$F(1) = \text{"first line"}$$
$$F(2) = \text{"second line"}$$
$$F(3) = \text{"third line"}$$

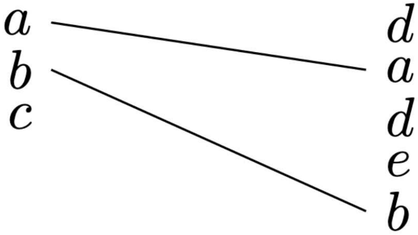
Version Control Category, contd.

Now that we have a mathematical notion of a file, we want to represent changes to them:

Definition: Patch

Given two files $F : [n] \to \text{Strings}$, $G : [m] \to \text{Strings}$, a **patch** from F to G is an injective increasing partial function $p : [n] \to [m]$, such that $G \circ p(i) = F(i)$ whenever $p(i)$ is defined.

e.g. the following diagram is a morphism:



Version Control Category, contd.

Now we can formally define a category:

Definition: Category of Files

Define $\mathcal{L}$ to be the category of all files $F : [n] \to \mathrm{String}$ for $n \in \mathbb{N}$, where morphisms are patches between objects and the identities are the identity patch $\mathrm{id}_n : [n] \to [n]$.

With this, we can finally express what a merge conflict looks like in categorical terms.

Version Control Category, contd.

What is a "merge" in our category?

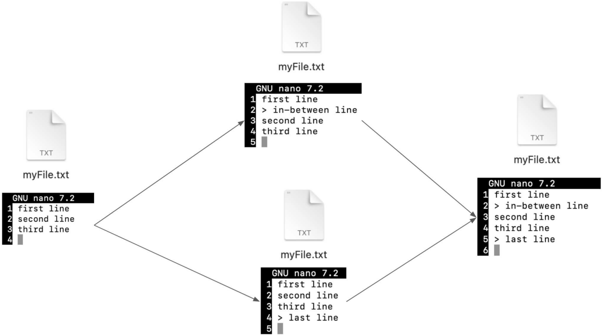
$$
\begin{tikzcd}
F \arrow[r, "g"] \arrow[d, "h"'] & G \\
H &
\end{tikzcd}
\qquad
\begin{tikzcd}
F \arrow[r, "g"] \arrow[d, "h"'] & G \arrow[d, "h'"] \\
H \arrow[r, "g'"'] & X
\end{tikzcd}
\qquad
\begin{tikzcd}
F \arrow[r, "g"] \arrow[d, "h"'] & G \arrow[d, "h'"] \arrow[ddr, bend left] & \\
H \arrow[r, "g'"'] \arrow[drr, bend right] & X \arrow[dr, dashed, "\exists!"] & \\
& & X'
\end{tikzcd}
$$

Version Control Category, contd.

What is a "merge" in our category?

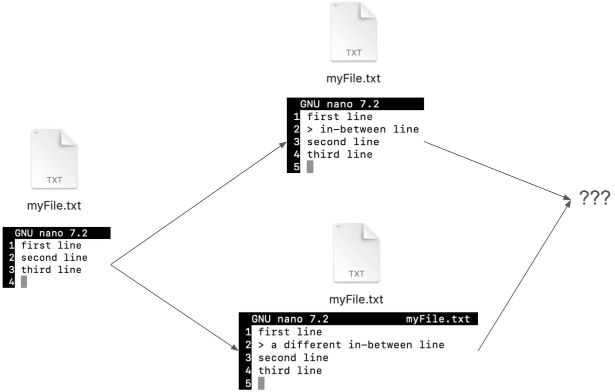
$$
\begin{tikzcd}
F \arrow[r, "g"] \arrow[d, "h"'] & G \\
H &
\end{tikzcd}
\qquad
\begin{tikzcd}
F \arrow[r, "g"] \arrow[d, "h"'] \arrow[dr, phantom, "\lrcorner", very near start] & G \arrow[d, "h'"] \\
H \arrow[r, "g'"'] & X
\end{tikzcd}
\qquad
\begin{tikzcd}
F \arrow[r, "g"] \arrow[d, "h"'] \arrow[dr, phantom, "\lrcorner", very near start] & G \arrow[d, "h'"] \arrow[ddr, bend left] & \\
H \arrow[r, "g'"'] \arrow[drr, bend right] & X \arrow[dr, dashed, "\exists!"] & \\
 & & X'
\end{tikzcd}
$$

i.e. we want a colimit of the first diagram ("**pushout**")



$$
\begin{tikzcd}
ab \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & acb \arrow[d] \\
abd \arrow[r] & acbd
\end{tikzcd}
$$

Version Control Category



$$
\begin{tikzcd}
ab \arrow[r] \arrow[d] & acb \arrow[d] \\
adb \arrow[r] & ??
\end{tikzcd}
$$

Unfortunately not all diagrams in $\mathcal{L}$ have pushouts

Version Control Category

We care about colimits in general:

$$\begin{tikzcd}
 & & & & H_5 \\
F_1 \arrow[r] & F_2 \arrow[r] \arrow[dr] & F_3 \arrow[r] & F_4 \arrow[r] \arrow[ur] & F_5 \\
 & & G_3 \arrow[r] & G_4 \arrow[r] & G_5
\end{tikzcd}$$

Theorem

A category is cocomplete if it has an initial object (colimit of the empty diagram) and all small pushouts.

So $\mathcal{L}$ isn't cocomplete... what if we try to make it one?

Cocompletion of $\mathcal{L}$

```
GNU nano 7.2                myFile.txt
1 first line
2 <<<<<<< HEAD
3 > in-between line
4 =======
5 > a different in-between line
6 >>>>>>> my-branch
7 second line
8 third line
```

$\Longrightarrow$ instead of indexing files by $[n]$, what if we use a more general object?

Cocompletion of $\mathcal{L}$

Definition: Poset

A **poset** is a pair $(X, \leq)$ satisfying

1. $x \leq x$ for all $x \in X$;
2. $a \leq b$ and $b \leq c \implies a \leq c$;
3. $a \leq b$ and $b \leq a \implies a = b$.

Then we can modify our definitions of files and patches:

Definitions: Files and Patches

A **poset file** is a function $F : X \to \mathrm{Strings}$, where X is a finite poset.
Given two poset files $F : X \to \mathrm{String}$ and $G : Y \to \mathrm{String}$, a **poset patch** is an ascending partial function $p : X \to Y$ with $G \circ p(i) = F(i)$ for all $i \in X$ where $p(i)$ is defined.

Because $[n]$ is a poset, every file is a poset file and every patch is a poset patch.

Cocomplection of $\mathcal{L}$

Let $\mathcal{P}$ be the category of poset files, with poset patches as morphisms. Then $\mathcal{L} \subset \mathcal{P}$, and $\mathcal{P}$ has limits of all diagrams in $\mathcal{L}$. But is $\mathcal{P}$ a canonical cocompletion of $\mathcal{L}$? i.e.

$$\begin{tikzcd} \mathcal{L} \arrow[r, hook, "y"] \arrow[d, "F"'] & \mathcal{P} \arrow[dl, dashed, "\exists! \widetilde{F}"] \\ \mathcal{C} & \end{tikzcd}$$

?

Better question: Is $\mathcal{P}$ even cocomplete?

$\mathcal{P}$ isn't cocomplete

It turns out it isn't!

$\mathcal{P}$ isn't cocomplete

It turns out it isn't!

Theorem

The (free conservative finite) cocompletion of $\mathcal{L}$ is the category of files over transitive sets (i.e. $F : X \to \mathrm{String}$ over $(X, <)$ where $<$ is a transitive relation) where morphisms are partial functions that obey transitivity.

$\mathcal{P}$ isn't cocomplete

It turns out it isn't!

Theorem (Mimram-Giusto)

The (free conservative finite) cocompletion of $\mathcal{L}$ is the category of files over transitive sets (i.e. $F : X \to \mathrm{String}$ over $(X, <)$ where $<$ is a transitive relation) where morphisms are partial functions that obey transitivity.

Another formulation:

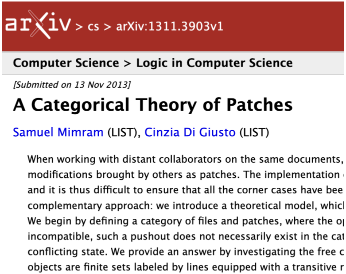
Theorem (Mimram-Giusto)

The (free conservative finite) cocompletion of $\mathcal{L}$ is equivalent to the full subcategory of the category of graphs $\hat{\mathcal{G}}$ whose objects are finite graphs with the following property: For every path $x \twoheadrightarrow y$, there exists exactly one edge $x \to y$.

1. The issue of conflict-free merging can be solved only by indexing our files by a special kind of graph
2. Systems that involve composition can be easily "categorified"
3. Categorifyng systems can reveal special insights that we otherwise could have missed

(Re)Sources

If you're interested...


arXiv > cs > arXiv:1311.3903v1

Computer Science > Logic in Computer Science

[Submitted on 13 Nov 2013]

A Categorical Theory of Patches

Samuel Mimram (LIST), Cinzia Di Giusto (LIST)

When working with distant collaborators on the same documents, modifications brought by others as patches. The implementation and it is thus difficult to ensure that all the corner cases have bee complementary approach: we introduce a theoretical model, whicl We begin by defining a category of files and patches, where the o incompatible, such a pushout does not necessarily exist in the cat conflicting state. We provide an answer by investigating the free c objects are finite sets labeled by lines equipped with a transitive r

pijul.com

TOPOS INSTITUTE