A Categorical Theory of Patches

Samuel Mimram Cinzia Di Giusto

CEA, LIST*

Abstract

When working with distant collaborators on the same documents, one often uses a version control system, which is a program tracking the history of files and helping importing modifications brought by others as patches. The implementation of such a system requires to handle lots of situations depending on the operations performed by users on files, and it is thus difficult to ensure that all the corner cases have been correctly addressed. Here, instead of verifying the implementation of such a system, we adopt a complementary approach: we introduce a theoretical model, which is defined abstractly by the universal property that it should satisfy, and work out a concrete description of it. We begin by defining a category of files and patches, where the operation of merging the effect of two coinitial patches is defined by pushout. Since two patches can be incompatible, such a pushout does not necessarily exist in the category, which raises the question of which is the correct category to represent and manipulate files in conflicting state. We provide an answer by investigating the free completion of the category of files under finite colimits, and give an explicit description of this category: its objects are finite sets labeled by lines equipped with a transitive relation and morphisms are partial functions respecting labeling and relations.

1 Introduction

It is common nowadays, when working with distant collaborators on the same files (multiple authors writing an article together for instance), to use a program which will track the history of files and handle the operation of importing modifications of other participants. These software called *version control systems* (VCS for short), like `git` or Darcs, implement two main operations. When a user is happy with the changes it has brought to the files it can record those changes in a *patch* (a file coding the differences between the current version and the last recorded version) and *commit* them to a server, called a *repository*. The user can also *update* its current version of the file by importing new patches added by other users to the repository and applying the corresponding modifications to the files. One of the main difficulties to address here is that there is no global notion of "time": patches are only partially ordered. For instance consider a repository with one file A and two users u_1 and u_2. Suppose that u_1 modifies file A into B by committing a patch f, which is then imported by u_2, and

*This work was partially supported by the French project ANR-11-INSE-0007 REVER.

then u_1 and u_2 concurrently modify the file B into C (resp. D) by committing a patch g (resp. h). The evolution of the file is depicted on the left and the partial ordering of patches in the middle:

$$\begin{tikzcd}
C & & D \\
& B \arrow[ul, "g"] \arrow[ur, "h"'] & \\
& A \arrow[u, "f"] &
\end{tikzcd}
\qquad
\begin{tikzcd}
g & & h \\
& f \arrow[ul] \arrow[ur] &
\end{tikzcd}
\qquad
\begin{tikzcd}
& E & \\
C \arrow[ur, "h/g"] & & D \arrow[ul, "g/h"'] \\
& B \arrow[ul, "g"] \arrow[ur, "h"'] & \\
& A \arrow[u, "f"] &
\end{tikzcd}$$

Now, suppose that u_2 imports the patch g or that u_1 imports the patch h. Clearly, this file resulting from the *merging* of the two patches should be the same in both cases, call it E. One way to compute this file, is to say that there should be a patch h/g, the *residual* of h after g, which transforms C into E and has the "same effect" as h once g has been applied, and similarly there should be a patch g/h transforming D into E. Thus, after each user has imported changes from the other, the evolution of the file is as pictured on the right above. In this article, we introduce a category $\mathcal{L}$ whose objects are files and morphisms are patches. Since residuals should be computed in the most general way, we formally define them as the arrows of pushout cocones, i.e. the square in the figure on the right should be a pushout.

However, as expected, not every pair of coinitial morphisms have a pushout in the category $\mathcal{L}$: this reflects the fact that two patches can be conflicting (for instance if two users modify the same line of a file). Representing and handling such conflicts in a coherent way is one of the most difficult part of implementing a VCS (as witnessed for instance by the various proposals for Darcs: mergers, conflictors, graphictors, etc. [10]). In order to be able to have a representation for all conflicting files, we investigate the free completion of the category $\mathcal{L}$ under all pushouts, this category being denoted $\mathcal{P}$, which corresponds to adding all conflicting files to the category, in the most general way as possible. This category can easily be shown to exist for general abstract reasons, and one of the main contributions of this work is to provide an explicit description by applying the theory of presheaves. This approach paves the way towards the implementation of a VCS whose correctness is deduced from universal categorical properties.

Related work. The Darcs community has investigated a formalization of patches based on commutation properties [10]. *Operational transformations* tackle essentially the same issues by axiomatizing the notion of residual patches [9]. In both cases, the fact that residual should form a pushout cocone is never explicitly stated, excepting in informal sentences saying that "g/f should have the same effect as g once f has been applied". We should also mention another interesting approach to the problem using inverse semigroups in [4]. Finally, Houston has proposed a category with pushouts, similar to ours, in order to model conflicting files [3], see Section 6.

Plan of the paper. We begin by defining a category $\mathcal{L}$ of files and patches in Section 2. Then, in Section 3, we abstractly define the category $\mathcal{P}$ of conflicting files obtained by free finite cocompletion. Section 4 provides a concrete description of the construction in the simpler case where patches can only insert lines.

We give some concrete examples in Section 5 and adapt the framework to the general case in Section 6. We conclude in Section 7.

2 Categories of files and patches

In this section, we investigate a model for a simplified VCS: it handles only one file and the only allowed operations are insertion and deletion of lines (modification of a line can be encoded by a deletion followed by an insertion). We suppose fixed a set $L = \{a, b, \ldots\}$ of *lines* (typically words over an alphabet of characters). A *file* A is a finite sequence of lines, which will be seen as a function $A : [n] \to L$ for some number of lines $n \in \mathbb{N}$, where the set $[n] = \{0, 1, \ldots, n-1\}$ indexes the lines of the files. For instance, a file A with three lines such that $A(0) = a$, $A(1) = b$ and $A(2) = c$ models the file *abc*. Given $a \in L$, we sometimes simply write a for the file $A : [1] \to L$ such that $A(0) = a$. A morphism between two files $A : [m] \to L$ and $B : [n] \to L$ is an injective increasing partial function $f : [m] \to [n]$ such that $\forall i \in [m], B \circ f(i) = A(i)$ whenever $f(i)$ is defined. Such a morphism is called a *patch*.

Definition 1. The category $\mathcal{L}$ has files as objects and patches as morphisms.

Notice that the category $\mathcal{L}$ is strictly monoidal with $[m] \otimes [n] = [m+n]$ and for every file $A : [m] \to L$ and $B : [n] \to L$, $(A \otimes B)(i) = A(i)$ if $i < m$ and $(A \otimes B)(i) = B(i-m)$ otherwise, the unit being the empty file $I : [0] \to L$, and tensor being defined on morphisms in the obvious way. The following proposition shows that patches are generated by the operations of inserting and deleting a line:

Proposition 2. *The category $\mathcal{L}$ is the free monoidal category containing L as objects and containing, for every line $a \in L$, morphisms $\eta_a : I \to a$ (insertion of a line a) and $\varepsilon_a : a \to I$ (deletion of a line a) such that $\varepsilon_a \circ \eta_a = \mathrm{id}_I$ (deleting an inserted line amounts to do nothing).*

Example 3. The patch corresponding to transforming the file *abc* into *dadeb*, by deleting the line c and inserting the lines labeled by d and e, is modeled by the partial function $f : [3] \to [5]$ such that $f(0) = 1$ and $f(1) = 4$ and $f(2)$ is undefined. Graphically,

$$\begin{tikzcd}[row sep=tiny]
a \arrow[dr, dash] & d \\
b \arrow[dddr, dash] & a \\
c & d \\
 & e \\
 & b
\end{tikzcd}$$

The deleted line is the one on which f is not defined and the inserted lines are those which are not in the image of f. In other words, f keeps track of the unchanged lines.

In order to increase readability, we shall consider the particular case where L is reduced to a single element. In this *unlabeled* case, the objects of $\mathcal{L}$ can be identified with integers (the labeling function is trivial), and Proposition 2 can be adapted to achieve the following description of the category, see also [6].

Proposition 4. *If L is reduced to a singleton, the category $\mathcal{L}$ is the free category whose objects are integers and morphisms are generated by $s_i^n : n \to n+1$ and*

$d_i^n : n+1 \to n$ for every $n \in \mathbb{N}$ and $i \in [n+1]$ (respectively corresponding to insertion and deletion of a line at i-th position), subject to the relations

$$s_i^{n+1} s_j^n = s_{j+1}^{n+1} s_i^n \qquad d_i^n s_i^n = \mathrm{id}_n \qquad d_i^n d_j^{n+1} = d_j^n d_{i+1}^{n+1} \tag{1}$$

whenever $0 \leq i \leq j < n$.

We will also consider the subcategory $\mathcal{L}^+$ of $\mathcal{L}$, with same objects, and *total* injective increasing functions as morphisms. This category models patches where the only possible operation is the insertion of lines: Proposition 2 can be adapted to show that $\mathcal{L}^+$ is the free monoidal category containing morphisms $\eta_a : I \to a$ and, in the unlabeled case, Proposition 4 can be similarly adapted to show that it is the free category generated by morphisms $s_i^n : n \to n+1$ satisfying $s_i^{n+1} s_j^n = s_{j+1}^{n+1} s_i^n$ with $0 \leq i \leq j < n$.

3 Towards a category of conflicting files

Suppose that A is a file which is edited by two users, respectively applying patches $f_1 : A \to A_1$ and $f_2 : A \to A_2$ to the file. For instance,

$$a \quad c \quad c \quad b \quad \xleftarrow{f_1} \quad a \quad b \quad \xrightarrow{f_2} \quad a \quad b \quad c \quad d \tag{2}$$

Now, each of the two users imports the modification from the other one. The resulting file, after the import, should be the smallest file containing both modifications on the original file: *accbcd*. It is thus natural to state that it should be a pushout of the diagram (2). Now, it can be noticed that not every diagram in $\mathcal{L}$ has a pushout. For instance, the diagram

$$a \quad c \quad b \quad \xleftarrow{f_1} \quad a \quad b \quad \xrightarrow{f_2} \quad a \quad d \quad b \tag{3}$$

does not admit a pushout in $\mathcal{L}$. In this case, the two patches f_1 and f_2 are said to be *conflicting*.

In order to represent the state of files after applying two conflicting patches, we investigate the definition of a category $\mathcal{P}$ which is obtained by completing the category $\mathcal{L}$ under all pushouts. Since, this completion should also contain an initial object (i.e. the empty file), we are actually defining the category $\mathcal{P}$ as the free completion of $\mathcal{L}$ under finite colimits: recall that a category is finitely cocomplete (has all finite colimits) if and only if it has an initial object and is closed under pushouts [6]. Intuitively, this category is obtained by adding files whose lines are not linearly ordered, but only partially ordered, such as on the left of

$$\begin{tikzcd} & a \arrow[dl] \arrow[dr] & \\ c \arrow[dr] & & d \arrow[dl] \\ & b & \end{tikzcd}$$

```
a
<<<<<<< HEAD
c
=======
d
>>>>>>> 5c55...
b
```

(4)

which would intuitively model the pushout of the diagram (3) if it existed, indicating that the user has to choose between c and d for the second line. Notice the similarities with the corresponding textual notation in `git` on the right. The name of the category $\mathcal{L}$ reflects the facts that its objects are files whose lines are linearly ordered, whereas the objects of $\mathcal{P}$ can be thought as files whose lines are only partially ordered. More formally, the category is defined as follows.

Definition 5. The category $\mathcal{P}$ is the *free finite conservative cocompletion* of $\mathcal{L}$: it is (up to equivalence of categories) the unique finitely cocomplete category together with an embedding functor $y : \mathcal{L} \to \mathcal{P}$ preserving finite colimits, such that for every finitely cocomplete category $\mathcal{C}$ and functor $F : \mathcal{L} \to \mathcal{C}$ preserving finite colimits, there exists, up to unique isomorphism, a unique functor $\tilde{F} : \mathcal{P} \to \mathcal{C}$ preserving finite colimits and satisfying $\tilde{F} \circ y = F$:

$$\begin{tikzcd}
\mathcal{L} \arrow[r, "F"] \arrow[d, "y"'] & \mathcal{C} \\
\mathcal{P} \arrow[ur, dotted, "\tilde{F}"'] &
\end{tikzcd}$$

Above, the term *conservative* refers to the fact that we preserve colimits which already exist in $\mathcal{L}$ (we will only consider such completions here). The "standard" way to characterize the category $\mathcal{P}$, which always exists, is to use the following folklore theorem, often attributed to Kelly [5, 1]:

Theorem 6. *The conservative cocompletion of the category $\mathcal{L}$ is equivalent to the full subcategory of $\hat{\mathcal{L}}$ whose objects are presheaves which preserve finite limits, i.e. the image of a limit in $\mathcal{L}^{\mathrm{op}}$ (or equivalently a colimit in $\mathcal{L}$) is a limit in* **Set** *(and limiting cones are transported to limiting cones). The finite conservative cocompletion $\mathcal{P}$ can be obtained by further restricting to presheaves which are finite colimits of representables.*

Example 7. The category **FinSet** of finite sets and functions is the conservative cocompletion of the terminal category **1**.

We recall that the category $\hat{\mathcal{L}}$ of *presheaves* over $\mathcal{L}$, is the category of functors $\mathcal{L}^{\mathrm{op}} \to \mathbf{Set}$ and natural transformations between them. The *Yoneda functor* $y : \mathcal{L} \to \hat{\mathcal{L}}$ defined on objects $n \in \mathcal{L}$ by $yn = \mathcal{L}(-, n)$, and on morphisms by postcomposition, provides a full and faithful embedding of $\mathcal{L}$ into the corresponding presheaf category, and can be shown to corestrict into a functor $y : \mathcal{L} \to \mathcal{P}$ [1]. A presheaf of the form yn for some $n \in \mathcal{L}$ is called *representable.*

Extracting a concrete description of the category $\mathcal{P}$ from the above proposition is a challenging task, because we a priori need to characterize firstly all diagrams admitting a colimit in $\mathcal{L}$, and secondly all presheaves in $\hat{\mathcal{L}}$ which preserve those diagrams. This paper introduces a general methodology to build such a category. In particular, perhaps a bit surprisingly, it turns out that we have to "allow cycles" in the objects of the category $\mathcal{P}$, which will be described as *the category whose objects are finite sets labeled by lines together with a transitive relation and morphisms are partial functions respecting labels and relations.*

4 A cocompletion of files and insertions of lines

In order to make our presentation clearer, we shall begin our investigation of the category $\mathcal{P}$ in a simpler case, which will be generalized in Section 6: we compute the free finite cocompletion of the category $\mathcal{L}^+$ (patches can only insert lines) in the case where the set of labels is a singleton. *To further lighten notations, in this section, we simply write $\mathcal{L}$ for this category.*

We sometimes characterize the objects in $\mathcal{L}$ as finite colimits of objects in a subcategory $\mathcal{G}$ of $\mathcal{L}$. This category $\mathcal{G}$ is the full subcategory of $\mathcal{L}$ whose objects are 1 and 2: it is the free category on the graph $\begin{tikzcd} 1 \arrow[r, shift left] \arrow[r, shift right] & 2 \end{tikzcd}$, the two arrows being s^1_0 and s^1_1. The category $\hat{\mathcal{G}}$ of presheaves over $\mathcal{G}$ is the category of *graphs*: a presheaf $P \in \hat{\mathcal{G}}$ is a graph with $P(1)$ as vertices, $P(2)$ as edges, the functions $P(s^1_1)$ and $P(s^1_0)$ associate to a vertex its source and target respectively, and morphisms correspond to usual morphisms of graphs. We denote by $x \twoheadrightarrow y$ a path going from a vertex x to a vertex y in such a graph. The inclusion functor $I : \mathcal{G} \to \mathcal{L}$ induces, by precomposition, a functor $I^* : \hat{\mathcal{L}} \to \hat{\mathcal{G}}$. The image of a presheaf in $\hat{\mathcal{L}}$ under this functor is called its *underlying graph*. By well known results about presheaves categories, this functor admits a right adjoint $I_* : \hat{\mathcal{G}} \to \hat{\mathcal{L}}$: given a graph $G \in \hat{\mathcal{G}}$, its image under the right adjoint is the presheaf $G_* \in \hat{\mathcal{L}}$ such that for every $n \in \mathbb{N}$, $G_*(n+1)$ is the set of paths of length n in the graph G, with the expected source maps, and $G_*(0)$ is reduced to one element.

Recall that every functor $F : \mathcal{C} \to \mathcal{D}$ induces a *nerve functor* $N_F : \mathcal{D} \to \hat{\mathcal{C}}$ defined on an object $A \in \mathcal{D}$ by $N_F(A) = \mathcal{D}(F-, A)$ [7]. Here, we will consider the nerve $N_I : \mathcal{L} \to \hat{\mathcal{G}}$ associated to the inclusion functor $I : \mathcal{G} \to \mathcal{L}$. An easy computation shows that the image $N_I(n)$ of $n \in \mathcal{L}$ is a graph with n vertices, so that its objects are isomorphic to $[n]$, and there is an arrow $i \to j$ for every $i, j \in [n]$ such that $i < j$. For instance,

$$N_I(3) = \begin{tikzcd} 0 \arrow[r] \arrow[rr, bend right] & 1 \arrow[r] & 2 \end{tikzcd} \qquad\qquad N_I(4) = \begin{tikzcd} 0 \arrow[r] \arrow[rr, bend left] \arrow[rrr, bend right] & 1 \arrow[r] \arrow[rr, bend left] & 2 \arrow[r] & 3 \end{tikzcd}$$

It is, therefore, easy to check that this embedding is full and faithful, i.e. morphisms in $\mathcal{L}$ correspond to natural transformations in $\hat{\mathcal{G}}$. Moreover, since $N_I(1)$ is the graph reduced to a vertex and $N_I(2)$ is the graph reduced to two vertices and one arrow between them, every graph can be obtained as a finite colimit of the graphs $N_I(1)$ and $N_I(2)$ by "gluing arrows along vertices". For instance, the initial graph $N_I(0)$ is the colimit of the empty diagram, and the graph $N_I(3)$ is the colimit of the diagram

$$\begin{tikzcd} & N_I(2) & & N_I(2) & \\ N_I(1) \arrow[ur, "N_I(s_1)"] \arrow[drr, "N_I(s_1)"'] & & N_I(1) \arrow[ul, "N_I(s_0)"'] \arrow[ur, "N_I(s_1)"] & & N_I(1) \arrow[ul, "N_I(s_0)"'] \arrow[dll, "N_I(s_0)"] \\ & & N_I(2) & & \end{tikzcd}$$

which may also be drawn as on the left of

$$\begin{tikzcd} & \bullet\to\bullet & & \bullet\to\bullet & \\ \bullet \arrow[ur] \arrow[drr] & & \bullet \arrow[ul] \arrow[ur] & & \bullet \arrow[ul] \arrow[dll] \\ & & \bullet\to\bullet & & \end{tikzcd} \qquad\qquad \begin{tikzcd} & 2 & & 2 & \\ 1 \arrow[ur] \arrow[drr] & & 1 \arrow[ul] \arrow[ur] & & 1 \arrow[ul] \arrow[dll] \\ & & 2 & & \end{tikzcd}$$

by drawing the graphs $N_I(0)$ and $N_I(1)$. Notice, that the object 3 is the colimit of the corresponding diagram in $\mathcal{L}$ (on the right), and this is generally true for all objects of $\mathcal{L}$, moreover this diagram is described by the functor $\mathrm{El}(N_I(3)) \xrightarrow{\pi} \mathcal{L}$. The notation $\mathrm{El}(P)$ refers to the *category of elements* of a presheaf $P \in \hat{\mathcal{C}}$, whose objects are pairs (A, p) with $A \in \mathcal{C}$ and $p \in P(A)$ and morphisms $f : (A, p) \to (B, q)$ are morphisms $f : A \to B$ in $\mathcal{C}$ such that $P(f)(q) = p$, and π is the first projection functor. The functor $I : \mathcal{G} \to \mathcal{L}$ is thus a dense functor in the sense of Definition 9 below, see [7] for details.

Proposition 8. *Given a functor $F : \mathcal{C} \to \mathcal{D}$, with $\mathcal{D}$ cocomplete, the associated nerve $N_F : \mathcal{D} \to \hat{\mathcal{C}}$ admits a left adjoint $R_F : \hat{\mathcal{C}} \to \mathcal{D}$ called the* realization *along F. This functor is defined on objects $P \in \hat{\mathcal{C}}$ by*

$$R_F(P) \quad = \quad \mathrm{colim}(\mathrm{El}(P) \xrightarrow{\pi} \mathcal{C} \xrightarrow{F} \mathcal{D})$$

Proof. Given a presheaf $P \in \hat{\mathcal{C}}$ and an object D, it can be checked directly that morphisms $P \to N_F D$ in $\hat{\mathcal{C}}$ with cocones from $\mathrm{El}(P) \xrightarrow{\mathcal{D}}$ to D, which in turn are in bijection with morphisms $R_F(P) \to D$ in $\mathcal{D}$, see [7]. □

Definition 9. A functor $F : \mathcal{C} \to \mathcal{D}$ is *dense* if it satisfies one of the two equivalent conditions:

(i) the associated nerve functor $N_F : \mathcal{D} \to \hat{\mathcal{C}}$ is full and faithful,

(ii) every object of $\mathcal{D}$ is canonically a colimit of objects in $\mathcal{C}$: for every $D \in \mathcal{D}$,

$$D \quad \cong \quad \mathrm{colim}(\mathrm{El}(N_F D) \xrightarrow{\pi} \mathcal{C} \xrightarrow{F} \mathcal{D}) \tag{5}$$

Since the functor I is dense, every object of $\mathcal{L}$ is a finite colimit of objects in $\mathcal{G}$, and $\mathcal{G}$ does not have any non-trivial colimit. One could expect the free conservative finite cocompletion of $\mathcal{L}$ to be the free finite cocompletion $\mathcal{P}$ of $\mathcal{G}$. We will see that this is not the case because the image in $\mathcal{L}$ of a non-trivial diagram in $\mathcal{G}$ might still have a colimit. By Theorem 6, the category $\mathcal{P}$ is the full subcategory of $\hat{\mathcal{L}}$ of presheaves preserving limits, which we now describe explicitly. This category will turn out to be equivalent to a full subcategory of $\hat{\mathcal{G}}$ (Theorem 15). We should first remark that those presheaves satisfy the following properties:

Proposition 10. *Given a presheaf $P \in \hat{\mathcal{L}}$ which is an object of $\mathcal{P}$,*

1. *the underlying graph of P is finite,*

2. *for each non-empty path $x \twoheadrightarrow y$ there exists exactly one edge $x \to y$ (in particular there is at most one edge between two vertices),*

3. *$P(n+1)$ is the set of paths of length n in the underlying graph of P, and $P(0)$ is reduced to one element.*

Proof. We suppose given a presheaf $P \in \mathcal{P}$, it preserves limits by Theorem 6. The diagram on the left

$$\begin{tikzcd} & 3 & \\ 2 \arrow[ur, "s_2^2"] & & 2 \arrow[ul, "s_0^2"'] \\ & 1 \arrow[ul, "s_0^1"] \arrow[ur, "s_1^1"'] & \end{tikzcd} \qquad\qquad \begin{tikzcd} & P(3) \arrow[dl, "P(s_2^2)"'] \arrow[dr, "P(s_0^2)"] & \\ P(2) \arrow[dr, "P(s_0^1)"'] & & P(2) \arrow[dl, "P(s_1^1)"] \\ & P(1) & \end{tikzcd}$$

is a pushout in $\mathcal{L}$, or equivalently the dual diagram is a pullback in $\mathcal{L}^{\mathrm{op}}$. Therefore, writing D for the diagram $2 \xleftarrow{s_0^1} 1 \xrightarrow{s_1^1} 2$ in $\mathcal{L}$, a presheaf $P \in \mathcal{P}$ should satisfy $P((\mathrm{colim}\, D)^{\mathrm{op}}) \cong \lim P(D^{\mathrm{op}})$, i.e. the above pushout diagram in $\mathcal{L}$ should be transported by P into the pullback diagram in **Set** depicted on the right of the above figure. This condition can be summarized by saying that P should satisfy the isomorphism $P(3) \cong P(2) \times_{P(1)} P(2)$ (and this isomorphism should respect obvious source and target maps given by the fact that the functor P should send a limiting cone to a limiting cone). From this fact, one can deduce that the elements α of $P(3)$ are in bijection with the paths $x \to y \to z$ of length 2 in the underlying graph of P going from $x = P(s_2^2 s_1^1)(\alpha)$ to $z = P(s_0^2 s_0^1)(\alpha)$. In particular, this implies that for any path $\alpha = x \to y \to z$ of length 2 in the underlying graph of P, there exists an edge $x \to z$, which is $P(s_1^2)(\alpha)$. More generally, given any integer $n > 1$, the object $n+1$ is the colimit in $\mathcal{L}$ of the diagram

$$\begin{tikzcd}
& 2 & & 2 & & & & 2 & & 2 & \\
1 \arrow[ur, "s_1^1"] & & 1 \arrow[ul, "s_0^1"'] \arrow[ur, "s_1^1"] & & {} \arrow[ul, "s_0^1"'] & \cdots & {} \arrow[ur, "s_1^1"] & & 1 \arrow[ul, "s_0^1"'] \arrow[ur, "s_1^1"] & & 1 \arrow[ul, "s_0^1"']
\end{tikzcd} \tag{6}$$

with $n+1$ occurrences of the object 1, and n occurrences of the object 2. Therefore, for every $n \in \mathbb{N}$, $P(n+1)$ is isomorphic to the set of paths of length n in the underlying graph. Moreover, since the diagram

$$\begin{tikzcd}
& 2 & & 2 & & & & 2 & & 2 & \\
1 \arrow[ur, "s_1^1"] \arrow[rrrrrd, "s_1^1"'] & & 1 \arrow[ul, "s_0^1"'] \arrow[ur, "s_1^1"] & & {} \arrow[ul, "s_0^1"'] & \cdots & {} \arrow[ur, "s_1^1"] & & 1 \arrow[ul, "s_0^1"'] \arrow[ur, "s_1^1"] & & 1 \arrow[ul, "s_0^1"'] \arrow[llllld, "s_0^1"] \\
& & & & & 2 & & & & &
\end{tikzcd} \tag{7}$$

with $n+1$ occurrences of the object 1 also admits the object $n+1$ as colimit, we should have $P(n+1) \cong P(n+1) \times P(2)$ between any two vertices x and y, i.e. for every non-empty path $x \twoheadrightarrow y$ there exists exactly one edge $x \to y$. Also, since the object 0 is initial in $\mathcal{L}$, it is the colimit of the empty diagram. The set $P(0)$ should thus be the terminal set, i.e. reduced to one element. Finally, since I is dense, P should be a finite colimit of the representables $N_I(1)$ and $N_I(2)$, the set $P(1)$ is necessarily finite, as well as the set $P(2)$ since there is at most one edge between two vertices. □

Conversely, we wish to show that the conditions mentioned in the above proposition exactly characterize the presheaves in $\mathcal{P}$ among those in $\hat{\mathcal{L}}$. In order to prove so, by Theorem 6, we have to show that presheaves P satisfying these conditions preserve finite limits in $\mathcal{L}$, i.e. that for every finite diagram $D : \mathcal{J} \to \mathcal{L}$ admitting a colimit we have $P(\operatorname{colim} D) \cong \lim(P \circ D^{\mathrm{op}})$. It seems quite difficult to characterize the diagrams admitting a colimit in $\mathcal{L}$, however the following lemma shows that it is enough to check diagrams "generated" by a graph which admits a colimit.

Lemma 11. *A presheaf $P \in \hat{\mathcal{L}}$ preserves finite limits if and only if it sends the colimits of diagrams of the form*

$$\mathrm{El}(G) \xrightarrow{\pi_G} \mathcal{G} \xrightarrow{I} \mathcal{L} \tag{8}$$

to limits in **Set***, where $G \in \hat{\mathcal{G}}$ is a finite graph such that the above diagram admits a colimit. Such a diagram in $\mathcal{L}$ is said to be* generated by the graph G.

Proof. In order to check that a presheaf $P \in \hat{\mathcal{L}}$ preserves finite limits, we have to check that it sends colimits of finite diagrams in $\mathcal{L}$ *which admit a colimit* to limits in **Set**, and therefore we have to characterize diagrams which admit colimits in $\mathcal{L}$. Suppose given a diagram $K : \mathcal{J} \to \mathcal{L}$. Since I is dense, every object of $\mathcal{L}$ is a colimit of a diagram involving only the objects 1 and 2 (see

Definition 9). We can therefore suppose that this is the case in the diagram K. Finally, it can be shown that the diagram K admits the same colimit as a diagram containing only s_0^1 and s_1^1 as arrows (these are the only non-trivial arrows in $\mathcal{L}$ whose source and target are 1 or 2), in which every object 2 is the target of exactly one arrow s_0^1 and one arrow s_1^1. For instance, the diagram in $\mathcal{L}$ below on the left admits the same colimits as the diagram in the middle.

$$\begin{tikzcd}
2 & & 3 & & \\
& 1 \arrow[ul, "s_0^1"] \arrow[ur, "s_2^2 s_1^1"] \arrow[dr, "s_1^1"'] & & 1 \arrow[ul, "s_0^2 s_0^1"'] \arrow[dl, "s_0^1"] & 1 \arrow[dll, bend left, "s_0^1"] \\
& & 2 & &
\end{tikzcd}
\qquad
\begin{tikzcd}
& 2 & & 2 & & 2 & \\
1 \arrow[ur, "s_1^1"] & & 1 \arrow[ul, "s_0^1"'] \arrow[ur, "s_1^1"] \arrow[dr, "s_1^1"'] & & 1 \arrow[ul, "s_0^1"'] \arrow[ur, "s_1^1"] & & 1 \arrow[ul, "s_0^1"'] \arrow[dlll, "s_0^1"] \\
& & & 2 & & &
\end{tikzcd}
\qquad
\begin{tikzcd}
0 \arrow[r] & 1 \arrow[r] \arrow[rr, bend right] & 2 \arrow[r] & 3
\end{tikzcd}$$

Any such diagram K is obtained by gluing a finite number of diagrams of the form $1 \xrightarrow{s_1^1} 2 \xleftarrow{s_0^1} 1$ along objects 1, and is therefore of the form $\mathrm{El}(G) \xrightarrow{\pi} \mathcal{G} \xrightarrow{I} \mathcal{L}$ for some finite graph $G \in \hat{\mathcal{G}}$: the objects of G are the objects 1 in K, the edges of G are the objects 2 in K and the source and target of an edge 2 are respectively given by the sources of the corresponding arrows s_1^1 and s_0^1 admitting it as target. For instance, the diagram in the middle above is generated by the graph on the right. The fact that every diagram is generated by a presheaf (is a discrete fibration) also follows more abstractly and generally from the construction of the comprehensive factorization system on **Cat** [8, 11]. □

Among diagrams generated by graphs, those admitting a colimit can be characterized using the following proposition:

Lemma 12. *Given a graph $G \in \hat{\mathcal{G}}$, the associated diagram* (8) *admits a colimit in $\mathcal{L}$ if and only if there exists $n \in \mathcal{L}$ and a morphism $f : G \to N_I n$ in $\hat{\mathcal{L}}$ such that every morphism $g : G \to N_I m$ in $\hat{\mathcal{L}}$, with $m \in \mathcal{L}$, factorizes uniquely through $N_I n$:*
$$\begin{tikzcd} G \arrow[r, "f"] \arrow[rr, bend right, "g"'] & N_I n \arrow[r, dotted] & N_I m \end{tikzcd}$$

Proof. Follows from the existence of a partially defined left adjoint to N_I, in the sense of [8], given by the fact that I is dense (see Definition 9). □

We finally arrive at the following concrete characterization of diagrams admitting colimits:

Lemma 13. *A finite graph $G \in \hat{\mathcal{G}}$ induces a diagram* (8) *in $\mathcal{L}$ which admits a colimit if and only if it is "tree-shaped", i.e. it is*

1. *acyclic: for any vertex x, the only path $x \twoheadrightarrow x$ is the empty path,*

2. *connected: for any pair of vertices x and y there exists a path $x \twoheadrightarrow y$ or a path $y \twoheadrightarrow x$.*

Proof. Given an object $n \in \mathcal{L}$, recall that $N_I n$ is the graph whose objects are elements of $[n]$ and there is an arrow $i \to j$ if and only if $i < j$. Given a finite graph G, morphisms $f : G \to N_I n$ are therefore in bijection with functions $f : V_G \to [n]$, where V_G denotes the set of vertices of G, such that $f(x) < f(y)$ whenever there exists an edge $x \to y$ (or equivalently, there exists a non-empty path $x \twoheadrightarrow y$).

Consider a finite graph $G \in \hat{\mathcal{G}}$, by Lemma 12, it induces a diagram (8) admitting a colimit if there is a universal arrow $f : G \to N_I n$ with $n \in \mathcal{L}$. From this it follows that the graph is acyclic: otherwise, we would have a non-empty path $x \twoheadrightarrow x$ for some vertex x, which would imply $f(x) < f(x)$. Similarly, suppose that G is a graph with vertices x and y such that there is no path $x \twoheadrightarrow y$ or $y \twoheadrightarrow x$, and there is an universal morphism $f : G \to N_I n$ for some $n \in \mathcal{L}$. Suppose that $f(x) \leq f(y)$ (the case where $f(y) \leq f(x)$ is similar). We can define a morphism $g : G \to N_I(n+1)$ by $g(z) = f(z) + 1$ if there is a path $x \twoheadrightarrow z$, $g(y) = f(x)$ and $g(z) = f(z)$ otherwise. This morphism is easily checked to be well-defined. Since we always have $f(x) \leq f(y)$ and $g(x) > g(y)$, there is no morphism $h : N_I n \to N_I(n+1)$ such that $h \circ f = g$.

Conversely, given a finite acyclic connected graph G, the relation $\leq$ defined on morphisms by $x \leq y$ whenever there exists a path $x \twoheadrightarrow y$ is a total order. Writing n for the number of vertices in G, the function $f : G \to N_I n$, which to a vertex associates the number of vertices strictly below it wrt $\leq$, is universal in the sense of Lemma 12. □

Proposition 14. *The free conservative finite cocompletion $\mathcal{P}$ of $\mathcal{L}$ is equivalent to the full subcategory of $\hat{\mathcal{L}}$ whose objects are presheaves P satisfying the conditions of Proposition 10.*

Proof. By Lemma 11, the category $\mathcal{P}$ is equivalent to the full subcategory of $\hat{\mathcal{L}}$ whose objects are presheaves preserving limits of diagrams of the form (8) generated by some graph $G \in \hat{\mathcal{G}}$ which admits a colimit, i.e. by Lemma 13 the finite graphs which are acyclic and connected. We write G_n for the graph with $[n]$ as vertices and edges $i \to (i+1)$ for $0 \leq i < n-1$. It can be shown that any acyclic and connected finite graph can be obtained from the graph G_n, for some $n \in \mathbb{N}$, by iteratively adding an edge $x \to y$ for some vertices x and y such that there exists a non-empty path $x \twoheadrightarrow y$. Namely, suppose given an acyclic and connected finite graph G. The relation $\leq$ on its vertices, defined by $x \leq y$ whenever there exists a path $x \twoheadrightarrow y$, is a total order, and therefore the graph G contains G_n, where n is the number of edges of G. An edge in G which is not in G_n is necessarily of the form $x \to y$ with $x \leq y$, otherwise it would not be acyclic. Since by Proposition 10, see (7), the diagram generated by a graph of the form

$$\begin{tikzcd}
& \bullet \arrow[r] & \bullet \arrow[r] & \cdots \arrow[r] & \bullet \arrow[r] & \bullet \arrow[dr] & \\
\bullet \arrow[ur] \arrow[rrrrrr] & & & & & & \bullet
\end{tikzcd}$$

is preserved by presheaves in $\mathcal{P}$ (which corresponds to adding an edge between vertices at the source and target of a non-empty path), it is enough to show that presheaves in $\mathcal{P}$ preserve diagrams generated by graphs G_n. This follows again by Proposition 10, see (6). □

One can notice that a presheaf $P \in \mathcal{P}$ is characterized by its underlying graph since $P(0)$ is reduced to one element and $P(n)$ with $n > 2$ is the set of paths of length n in this underlying graph: $P \cong I_*(I^*P)$. We can therefore simplify the description of the cocompletion of $\mathcal{L}$ as follows:

Theorem 15. *The free conservative finite cocompletion $\mathcal{P}$ of $\mathcal{L}$ is equivalent to the full subcategory of the category $\hat{\mathcal{G}}$ of graphs, whose objects are finite graphs such that for every non-empty path $x \twoheadrightarrow y$ there exists exactly one edge $x \to y$. Equivalently, it can be described as the category whose objects are finite sets equipped with a transitive relation $<$, and functions respecting relations.*

In this category, pushouts can be explicitly described as follows:

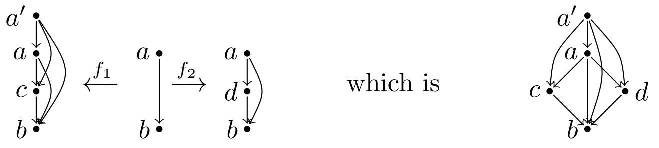
Proposition 16. *With the last above description, the pushout of a diagram in* $\mathcal{P}$ $(B, <_B) \xleftarrow{f} (A, <_A) \xrightarrow{g} (C, <_C)$ *is* $B \uplus C / \sim$ *with* $B \ni b \sim c \in C$ *whenever there exists* $a \in A$ *with* $f(a) = b$ *and* $f(a) = c$*, equipped with the transitive closure of the relation inherited by* $<_B$ *and* $<_C$*.*

Lines with labels. The construction can be extended to the labeled case (i.e. L is not necessarily a singleton). The forgetful functor $\hat{\mathcal{L}} \to \mathbf{Set}$ sending a presheaf P to the set $P(1)$ admits a right adjoint $! : \mathbf{Set} \to \hat{\mathcal{L}}$. Given $n \in \mathbb{N}^*$ the elements of $!L(n)$ are words u of length n over L, with $!L(s_i^{n-1})(u)$ being the word obtained from u by removing the i-th letter. The free conservative finite cocompletion $\mathcal{P}$ of $\mathcal{L}$ is the slice category $\mathcal{L}/!L$, whose objects are pairs (P, ℓ) consisting of a finite presheaf $P \in \hat{\mathcal{L}}$ together with a *labeling* morphism $\ell : P \to !L$ of presheaves. Alternatively, the description of Proposition 15 can be straightforwardly adapted by labeling the elements of the objects by elements of L (labels should be preserved by morphisms), thus justifying the use of labels for the vertices in following examples.

5 Examples

In this section, we give some examples of merging (i.e. pushout) of patches.

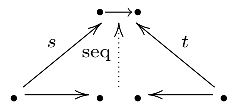
Example 17. Suppose that starting from a file ab, one user inserts a line a' at the beginning and c in the middle, while another one inserts a line d in the middle. After merging the two patches, the resulting file is the pushout of



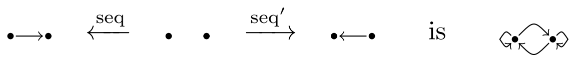
Example 18. Write G_1 for the graph with one vertex and no edges, and G_2 for the graph with two vertices and one edge between them. We write $s, t : G_1 \to G_2$ for the two morphisms in $\mathcal{P}$. Since $\mathcal{P}$ is finitely cocomplete, there is a coproduct $G_1 + G_1$ which gives, by universal property, an arrow $\mathrm{seq} : G_1 + G_1 \to G_2$:

$$\begin{tikzcd}
 & G_2 & \\
G_1 \arrow[ur, "s"] \arrow[r] & G_1 + G_1 \arrow[u, dotted, "\mathrm{seq}"] & G_1 \arrow[ul, "t"'] \arrow[l]
\end{tikzcd}$$

or graphically



that we call the *sequentialization morphism.* This morphism corresponds to the following patch: given two possibilities for a line, a user can decide to turn them into two consecutive lines. We also write $\mathrm{seq}' : G_1 + G_1 \to G_2$ for the morphism obtained similarly by exchanging s and t in the above cocone. Now, the pushout of



which illustrates how cyclic graphs appear in $\mathcal{P}$ during the cocompletion of $\mathcal{L}$.

Example 19. With the notations of the previous example, by taking the coproduct of two copies of $\mathrm{id}_{G_1} : G_1 \to G_1$, there is a universal morphism $G_1 + G_1 \to G_1$, which illustrates how two independent lines can be merged by a patch (in order to resolve conflicts).

$$\begin{tikzcd}
 & & \bullet & & \\
\bullet \arrow[r] \arrow[urr, "\mathrm{id}_\bullet"] & \bullet & {} \arrow[u, dotted, "\mathrm{merge}"] & \bullet & \bullet \arrow[l] \arrow[ull, "\mathrm{id}_\bullet"']
\end{tikzcd}$$

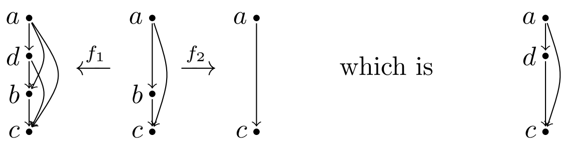
6 Handling deletions of lines

All the steps performed in previous sections in order to compute the free conservative finite cocompletion of the category $\mathcal{L}^+$ can be adapted in order to compute the cocompletion $\mathcal{P}$ of the category $\mathcal{L}$ as introduced in Definition 1, thus adding support for deletion of lines in patches. In particular, the generalization of the description given by Theorem 15 turns out to be as follows.

Theorem 20. *The free conservative finite cocompletion $\mathcal{P}$ of the category $\mathcal{L}$ is the category whose objects are triples $(A, <, \ell)$ where A is a finite set of lines, $<$ is a transitive relation on A and $\ell : A \to L$ associates a label to each line, and morphisms $f : (A, <_A, \ell_A) \to (B, <_B, \ell_B)$ are partial functions $f : A \to B$ such that for every $a, a' \in A$ both admitting an image under f, we have $\ell_B(f(a)) = \ell_A(a)$, and $a <_A a'$ implies $f(a) <_B f(a')$.*

Similarly, pushouts in this category can be computed as described in Proposition 16, generalized in the obvious way to partial functions.

Example 21. Suppose that starting from a file *abc*, one user inserts a line *d* after *a* and the other one deletes the line *b*. The merging of the two patches (in $\mathcal{P}'$) is the pushout of



i.e. the file *adc*. Notice that the morphism f_2 is partial: b has no image.

Interestingly, a category very similar to the one we have described in Theorem 20 was independently proposed by Houston [3] based on a construction performed in [2] for modeling asynchronous processes. This category is not equivalent to ours because morphisms are reversed partial functions: it is thus not the most general model (in the sense of being the free finite cocompletion). As a simplified explanation for this, consider the category **FinSet** which is the finite cocompletion of **1**. This category is finitely complete (in addition to cocomplete), thus $\mathbf{FinSet}^{\mathrm{op}}$ is finitely cocomplete and **1** embeds fully and faithfully in it. However, $\mathbf{FinSet}^{\mathrm{op}}$ is not the finite cocompletion of **1**. Another way to see this is that this category does not contain the “merging” morphism of Example 19, but it contains a dual morphism “duplicating” lines.

7 Concluding remarks and future works

In this paper, we have detailed how we could derive from universal constructions a category which suitably models files resulting from conflicting modifications. It is finitely cocomplete, thus the merging of any modifications of the file is well-defined.

We believe that the interest of our methodology lies in the fact that it adapts easily to other more complicated base categories $\mathcal{L}$ than the two investigated here: in future works, we should explain how to extend the model in order to cope with multiple files (which can be moved, deleted, etc.), different file types (containing text, or more structured data such as XML trees). Also, the structure of repositories (partially ordered sets of patches) is naturally modeled by event structures labeled by morphisms in $\mathcal{P}$, which will be detailed in future works, as well as how to model usual operations on repositories: cherry-picking (importing only one patch from another repository), using branches, removing a patch, etc. It would also be interesting to explore axiomatically the addition of inverses for patches, following other works hinted at in the introduction.

Once the theoretical setting is clearly established, we plan to investigate algorithmic issues (in particular, how to efficiently represent and manipulate the conflicting files, which are objects in $\mathcal{P}$). This should eventually serve as a basis for the implementation of a theoretically sound and complete distributed version control system (no unhandled corner-cases as in most current implementations of VCS).

Acknowledgments. The authors would like to thank P.-A. Melliès, E. Haucourt, T. Heindel, T. Hirschowitz and the anonymous reviewers for their enlightening comments and suggestions.

References

[1] J. Adámek and J. Rosicky. *Locally presentable and accessible categories*, volume 189. Cambridge Univ. Press, 1994.

[2] R. Cockett and D. Spooner. Categories for synchrony and asynchrony. *Electronic Notes in Theoretical Computer Science*, 1:66–90, 1995.

[3] R. Houston. *On editing text.* `http://bosker.wordpress.com/2012/05/10/on-editing-text`.

[4] J. Jacobson. A formalization of darcs patch theory using inverse semigroups. Technical report, CAM report 09-83, UCLA, 2009.

[5] M. Kelly. *Basic concepts of enriched category theory*, volume 64. Cambridge Univ. Press, 1982.

[6] S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics.* Springer Verlag, 1971.

[7] S. Mac Lane and I. Moerdijk. *Sheaves in geometry and logic: A first introduction to topos theory.* Springer, 1992.

[8] R. Paré. Connected components and colimits. *Journal of Pure and Applied Algebra*, 3(1):21–42, 1973.

[9] M. Ressel, D. Nitsche-Ruhland, and R. Gunzenhäuser. An integrating, transformation-oriented approach to concurrency control and undo in group editors. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 288–297. ACM, 1996.

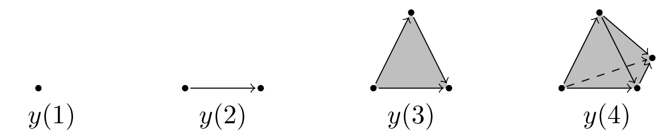
[10] D. Roundy and al. *The Darcs Theory.* `http://darcs.net/Theory`.

[11] R. Street and R. Walters. The comprehensive factorization of a functor. *Bull. Amer. Math. Soc*, 79(2):936–941, 1973.

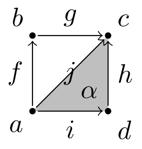
A A geometric interpretation of presheaves on $\mathcal{L}^+$

Since presheaf categories are sometimes a bit difficult to grasp, we recall here the geometric interpretation that can be done for presheaves in $\hat{\mathcal{L}}^+$. We forget about labels of lines and for simplicity suppose that the empty file is not allowed (the objects are strictly positive integers). In this section, we denote this category by $\mathcal{L}$. The same reasoning can be performed on the usual category $\mathcal{L}^+$, and even $\mathcal{L}$, but the geometrical explanation is a bit more involved to describe.

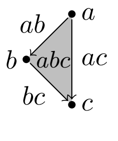
In this case, the presheaves in $\hat{\mathcal{L}}$ can easily be described in geometrical terms: the elements P of $\hat{\mathcal{L}}$ are *presimplicial sets.* Recall from Proposition 4 that the category $\mathcal{L}$ is the free category whose objects are strictly positive natural integers, containing for every integers $n \in \mathbb{N}^*$ and $i \in [n+1]$ morphisms $s_i^n : n \to n+1$, subject to the relations $s_i^{n+1} s_j^n = s_{j+1}^{n+1} s_i^n$ whenever $0 \leq i \leq j < n$. Writing $y : \mathcal{L} \to \hat{\mathcal{L}}$ for the Yoneda embedding, a representable presheaf $y(n+1) \in \hat{\mathcal{L}}_+^{\mathrm{fin}}$ can be pictured geometrically as an n-simplex: a 0-simplex is a point, a 1-simplex is a segment, a 2-simplex is a (filled) triangle, a 3-simplex is a (filled) tetrahedron, etc.:



Notice that the n-simplex has n faces which are $(n-1)$-dimensional simplices, and these are given by the image under $y(s_i^n)$, with $i \in [n]$, of the unique element of $y(n+1)(n+1)$: the i-th face of an n-simplex is the $(n-1)$-simplex obtained by removing the i-th vertex from the simplex. More generally, a presheaf $P \in \hat{\mathcal{L}}_+^{\mathrm{fin}}$ (a finite presimplicial set) is a finite colimit of representables: every such presheaf can be pictured as a gluing of simplices. For instance, the half-filled square on the right corresponds to the presimplicial set P with $P(1) = \{a, b, c, d\}$, $P(2) = \{f, g, h, i, j\}$, $P(3) = \{\alpha\}$ with faces $P(s_1^1)(f) = a$, $P(s_0^1)(f) = b$, etc.



Similarly, in the labeled case, a labeled presheaf $(P, \ell) \in \mathcal{L}/!$ can be pictured as a presimplicial set whose vertices (0-simplices) are labeled by elements of L. The word labeling of higher-dimensional simplices can then be deduced by concatenating the labels of the vertices it has as iterated faces. For instance, an edge (a 1-simplex) whose source is labeled by a and target is labeled by b is necessarily labeled by the word ab, etc.



More generally, presheaves in $\mathcal{L}^+$ can be pictured as *augmented presimplicial sets* and presheaves in $\mathcal{L}$ as *augmented simplicial sets*, a description of those can for instance be found in Hatcher's book *Algebraic Topology.*

B Proofs of classical propositions

In this section, we briefly recall proofs of well-known propositions as our proofs rely on a fine understanding of those. We refer the reader to [7] for further details.

Proposition 8 *Given a functor $F : \mathcal{C} \to \mathcal{D}$, with $\mathcal{D}$ cocomplete, the associated nerve $N_F : \mathcal{D} \to \hat{\mathcal{C}}$ admits a left adjoint $R_F : \hat{\mathcal{C}} \to \mathcal{D}$ called the* realization *along F. This functor is defined on objects $P \in \hat{\mathcal{C}}$ by*

$$R_F(P) \quad = \quad \operatorname{colim}(\operatorname{El}(P) \xrightarrow{\pi} \mathcal{C} \xrightarrow{F} \mathcal{D})$$

Proof. In order to show the adjunction, we have to construct a natural family of isormorphisms $\mathcal{D}(R_F(P), D) \cong \hat{\mathcal{C}}(P, N_F D)$ indexed by a presheaf $P \in \hat{\mathcal{C}}$ and an object $D \in \mathcal{D}$. A natural transformation $\theta \in \hat{\mathcal{C}}(P, N_F D)$ is a family of functions $(\theta_C : PC \to N_F DC)_{C \in \mathcal{C}}$ such that for every morphism $f : C' \to C$ in $\mathcal{C}$ the diagram

$$\begin{tikzcd}
P(C) \arrow[r, "\theta_C"] \arrow[d, "P(f)"'] & \mathcal{D}(FC, D) \arrow[d, "\mathcal{D}(Ff,D)"] \\
P(C') \arrow[r, "\theta_{C'}"'] & \mathcal{D}(FC', D)
\end{tikzcd}$$

commutes. It can also be seen as a family $(\theta_C(p) : FC \to D)_{(C,p) \in \operatorname{El}(P)}$ of morphisms in $\mathcal{D}$ such that the diagram

$$\begin{tikzcd}
FC \arrow[dr, "\theta_C(p)"] & \\
& D \\
FC' \arrow[uu, "Ff"] \arrow[ur, "\theta_{C'}(P(f)(p))"'] &
\end{tikzcd}
\qquad \text{or equivalently} \qquad
\begin{tikzcd}
F\pi_P(C,p) \arrow[dr, "\theta_C(p)"] & \\
& D \\
F\pi_P(C',p') \arrow[uu, "F\pi_P f"] \arrow[ur, "\theta_{C'}(p')"'] &
\end{tikzcd}$$

commutes for every morphism $f : C' \to C$ in $\mathcal{C}$. This thus defines a cocone from $F\pi_P : \operatorname{El}(P) \to \mathcal{D}$ to D, and those cocones are in bijection with morphisms $R_F(P) \to D$ by definition of $R_F(P)$ as a colimit: we have shown $\mathcal{D}(R_F(P), D) \cong \hat{\mathcal{C}}(P, N_F(D))$, from which we conclude. □

The equivalence between the two conditions of Definition 9 can be shown as follows.

Proposition 22. *Given a functor $F : \mathcal{C} \to \mathcal{D}$, the two following conditions are equivalent:*

(i) the associated nerve functor $N_F : \mathcal{D} \to \hat{\mathcal{C}}$ is full and faithful,

(ii) every object of $\mathcal{D}$ is canonically a colimit of objects in $\mathcal{C}$: for every $D \in \mathcal{D}$,

$$D \quad \cong \quad \operatorname{colim}(\operatorname{El}(N_F D) \xrightarrow{\pi} \mathcal{C} \xrightarrow{F} \mathcal{D})$$

Proof. In the case where $\mathcal{D}$ is cocomplete, the nerve functor $N_F : \mathcal{D} \to \hat{\mathcal{C}}$ admits $R_F : \hat{\mathcal{C}} \to \mathcal{D}$ as right adjoint, and the equivalence amounts to showing that the right adjoint is full and faithful if and only if the counit is an isomorphism, which is a classical theorem [6, Theorem IV.3.1]. The construction can be adapted to the general case where $\mathcal{D}$ is not necessarily cocomplete by considering $\operatorname{colim}(\operatorname{El}(-) \xrightarrow{\pi} \mathcal{C} \xrightarrow{F} \mathcal{D}) : \hat{\mathcal{C}} \to \mathcal{D}$ as a partially defined left adjoint (see [8]) and generalizing the theorem. □

C Proofs of the construction of the finite cocompletion

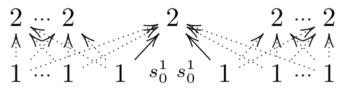
Lemma 11 *A presheaf $P \in \hat{\mathcal{L}}$ preserves finite limits, if and only if it sends the colimits of diagrams of the form*

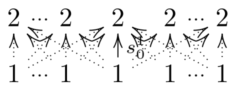
$$\mathrm{El}(G) \xrightarrow{\pi_G} \mathcal{G} \xrightarrow{I} \mathcal{L}$$

to limits in **Set**, *where $G \in \hat{\mathcal{G}}$ is a finite graph such that the above diagram admits a colimit. Such a diagram in $\mathcal{L}$ is said to be* generated by the graph G.

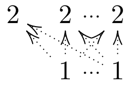
Proof. In order to check that a presheaf $P \in \hat{\mathcal{L}}$ preserves finite limits, we have to check that it sends colimits of finite diagrams in $\mathcal{L}$ *which admit a colimit* to limits in **Set**, and therefore we have to characterize diagrams which admit colimits in $\mathcal{L}$. The number of diagrams to check can be reduced by using the facts that limits commute with limits [6]. For instance, the inclusion functor $I : \mathcal{G} \to \mathcal{L}$ is dense, which implies that every object $n \in \mathcal{L}$ is canonically a colimit of the objects 1 and 2 by the formula $n \cong \mathrm{colim}(\mathrm{El}(N_I n) \xrightarrow{\pi} \mathcal{G} \xrightarrow{I} \mathcal{L})$, see Definition 9. Thus, given a finite diagram $K : \mathcal{J} \to \mathcal{L}$, we can replace any object n different from 1 and 2 occurring in the diagram by the corresponding diagram $\mathrm{El}(N_I n) \xrightarrow{\pi} \mathcal{G} \xrightarrow{I} \mathcal{L}$, thus obtaining a new diagram $K' : \mathcal{J} \to \mathcal{L}$ which admits the same colimit as K. This shows that P will preserve finite limits if and only if it preserves limits of finite diagrams in $\mathcal{L}$ in which the only occurring objects are 1 and 2. Since the only non-trivial arrows in $\mathcal{L}$ between the objects 1 and 2 are $s^1_0, s^1_1 : 1 \to 2$, and removing an identity arrow in a diagram does not change its colimit, the diagram K can thus be assimilated to a bipartite graph with vertices labeled by 1 or 2 and edges labeled by s^1_0 or s^1_1, all edges going from vertices 1 to vertices 2.

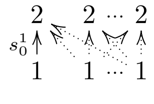
We can also reduce the number diagrams to check by remarking that some pairs of diagrams are "equivalent" in the sense that their image under P have the same limit, independently of P. For instance, consider a diagram in which an object 2 is the target of two arrows labeled by s^1_0 (on the left). The diagram obtained by identifying the two arrows along with the objects 1 in their source (on the right) can easily be checked to be equivalent by constructing a bijection between cocones of the first and cocones of the second.





More precisely, if we write $K : \mathcal{J}' \to \mathcal{L}$ and $K : \mathcal{J} \to \mathcal{L}$ for the two diagrams and $J : \mathcal{J}' \to \mathcal{J}$ for the obvious functor, the canonical arrow $\mathrm{colim}(K \circ J) \to \mathrm{colim}(K)$ is an isomorphism, i.e. the functor J is final. The same reasoning of course also holds with s^1_1 instead of s^1_0. We can therefore restrict ourselves to considering diagrams in which 2 is the target of at most one arrow s^1_0, and of at most one arrow s^1_1. Conversely, if an object 2 is the target of no arrow s^1_0 (on the left), then we can add a new object 1 and a new arrow from this object to the object 2 (on the right) and obtain an equivalent diagram:





The same reasoning holds with s_1^1 instead of s_0^1 and we can therefore restrict ourselves to diagrams in which every object 2 is the target of exactly one arrow s_0^1 and one arrow s_1^1.

Any such diagram K is obtained by gluing a finite number of diagrams of the form

$$\begin{tikzcd}
 & 2 & \\
1 \arrow[ur, "s_1^1"] & & 1 \arrow[ul, "s_0^1"']
\end{tikzcd}$$

along objects 1, and is therefore of the form $\mathrm{El}(G) \xrightarrow{\pi} \mathcal{G} \xrightarrow{I} \mathcal{L}$ for some finite graph $G \in \hat{\mathcal{G}}$: the objects of G are the objects 1 in K, the edges of G are the objects 2 in K and the source and target of an edge 2 are respectively given by the sources of the corresponding arrows s_1^1 and s_0^1 admitting it as target. For instance, the diagram on the left

$$\begin{tikzcd}
 & 2 & & 2 & & 2 & \\
1 \arrow[ur, "s_1^1"] & & 1 \arrow[ul, "s_0^1"'] \arrow[ur, "s_1^1"] \arrow[dr, "s_1^1"'] & & 1 \arrow[ul, "s_0^1"'] \arrow[ur, "s_1^1"] & & 1 \arrow[ul, "s_0^1"'] \arrow[dlll, "s_0^1"] \\
 & & & 2 & & &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
0 \arrow[r] & 1 \arrow[r] \arrow[rr, bend right] & 2 \arrow[r] & 3
\end{tikzcd}$$

is generated by the graph on the right. $\square$

Lemma 12 *Given a graph $G \in \hat{\mathcal{G}}$, the associated diagram* (8) *admits a colimit in $\mathcal{L}$ if and only if there exists $n \in \mathcal{L}$ and a morphism $f : G \to N_I n$ in $\hat{\mathcal{L}}$ such that every morphism $g : G \to N_I m$ in $\hat{\mathcal{L}}$, with $m \in \mathcal{L}$, factorizes uniquely through $N_I n$:*

$$\begin{tikzcd}
G \arrow[r, "f"] \arrow[rr, bend right, "g"'] & N_I n \arrow[r, dotted] & N_I m
\end{tikzcd}$$

Proof. We have seen in proof of Proposition 8 that morphisms in $\hat{\mathcal{L}}(G, N_I n)$ are in bijection with cocones in $\mathcal{L}$ from $\mathrm{El}(G) \xrightarrow{\pi_G} \mathcal{G} \xrightarrow{I} \mathcal{L}$ to n, and moreover given a morphism $h : n \to m$ in $\mathcal{G}$ the morphism $\hat{\mathcal{L}}(G, N_I n) \to \hat{\mathcal{L}}(G, N_I m)$ induced by post-composition with $N_I h$ is easily checked to correspond to the usual notion of morphism between n-cocones and m-cocones induced by $N_I h$ (every morphism $N_I n \to N_I m$ is of this form since N_I is full and faithful). We can finally conclude using the universal property defining colimiting cocones. $\square$

D Proofs for deletions of lines

In this section, we detail proofs of properties mentioned in Section 6.

D.1 Sets and partial functions

Before considering the conservative finite cocompletion of the category L, as introduced in Definition 1, it is enlightening to study the category **PSet** of sets and partial functions. A partial function $f : A \to B$ can always be seen

1. as a total function $f : A \to B \uplus \{\bot_A\}$ where $\bot_A$ is a fresh element wrt A, where given $a \in A$, $f(a) = \bot_A$ means that the partial function is undefined on A,

2. alternatively, as a total function $f : A \uplus \{\bot_A\} \to B \uplus \{\bot_B\}$ such that $f(\bot_A) = \bot_B$.

This thus suggests to consider the following category:

Definition 23. The category **pSet** of *pointed sets* has pairs (A, a) where A is a set and $a \in A$ as objects, and morphisms $f : (A, a) \to (B, b)$ are (total) functions $f : A \to B$ such that $f(a) = b$.

Point (ii) of the preceding discussion can be summarized by saying that a partial function can be seen as a pointed function and conversely:

Proposition 24. *The category* **PSet** *of sets and partial functions is equivalent to the category* **pSet** *of pointed sets.*

It is easily shown that the forgetful functor $U : \mathbf{pSet} \to \mathbf{Set}$, sending a pointed set (A, a) to the underlying set A, admits a left adjoint $F : \mathbf{Set} \to \mathbf{pSet}$, defined on objects by $FA = (A \uplus \{\bot_A\}, \bot_A)$. This adjunction induces a monad $T = UF$ on **Set**, from which point (i) can be formalized:

Proposition 25. *The category* **PSet** *is equivalent to the Kleisli category* $\mathbf{Set}_T$ *associated to the monad* $T : \mathbf{Set} \to \mathbf{Set}$.

Finally, it turns out that the category **pSet** of pointed sets might have been discovered from **PSet** using "presheaf thinking" as follows. We write $\mathcal{G}$ for the full subcategory of **PSet** containing two objects: the empty set $0 = \emptyset$ and a set $1 = \{*\}$ with only one element, and two non-trivial arrows $\star : 0 \to 1$ and $\bot : 1 \to 0$ (the undefined function) such that $\bot \circ \star = \mathrm{id}_0$. We write $I : \mathcal{G} \to \mathbf{PSet}$ for the inclusion functor. Consider the associated nerve functor $N_I : \mathbf{PSet} \to \hat{\mathcal{G}}$. Given a set A the presheaf $N_I A \in \hat{\mathcal{G}}$ is such that:

- $N_I A0 = \mathbf{PSet}(I0, A) \cong \{\star\}$: the only morphism $0 \to A$ in **PSet** is noted $\star$,
- $N_I A1 = \mathbf{PSet}(I1, A) \cong A \uplus \{\bot_A\}$: a morphism $1 \to A$ is characterized by the image of $* \in A$ which is either an element of A or undefined,
- $N_I A\star : N_I A1 \to N_I A0$ is the constant function whose image is $\star$,
- $N_I A\bot : N_I A0 \to N_I A1$ is the function such that the image of $\star$ is $\bot_A$.

Moreover, given $A, B \in \mathbf{PSet}$ a natural transformation from $N_I A$ to $N_I B$ is a pair of functions $f : A \uplus \{\bot_A\} \to B \uplus \{\bot_B\}$ and $g : \{\star\} \to \{\star\}$ such that the diagrams

$$\begin{tikzcd}
A \uplus \{\bot_A\} \arrow[r, "f"] \arrow[d, "N_I A\star"'] & B \uplus \{\bot_B\} \arrow[d, "N_I B\star"] \\
\{\star\} \arrow[r, "g"'] & \{\star\}
\end{tikzcd}
\quad \text{and} \quad
\begin{tikzcd}
A \uplus \{\bot_A\} \arrow[r, "f"] & \uplus\{\bot_B\} \\
\{\star\} \arrow[r, "g"'] \arrow[u, "N_I A\bot"] & \{\star\} \arrow[u, "N_I B\bot"']
\end{tikzcd}$$

commutes. Since $\{\star\}$ is the terminal set, such a natural transformation is characterized by a function $f : A \uplus \{\bot_A\} \to B \uplus \{\bot_B\}$ such that $f(\bot_A) = \bot_B$. The functor $N_I : \mathbf{PSet} \to \hat{\mathcal{G}}$ is thus dense and its image is equivalent to **pSet**.

D.2 A cocompletion of $\mathcal{L}$

The situation with regards to the category $\mathcal{L}$ is very similar. We follow the plan of Section 4 and first investigate the unlabeled case: $\mathcal{L}$ is the category with integers as objects and partial injective increasing functions $f : [m] \to [n]$ as morphisms $f : m \to n$.

We write $\mathcal{G}$ for the full subcategory of $\mathcal{L}$ whose objects are 0, 1 and 2. This is the free category on the graph

$$\begin{tikzcd}
0 \arrow[r, shift left, "s_0^0"] & 1 \arrow[l, shift left, "d_0^0"] \arrow[r, shift left=3, "s_0^1"] \arrow[r, shift left=1, "s_1^1"] & 2 \arrow[l, shift left=1, "d_0^1"] \arrow[l, shift left=3, "d_1^1"]
\end{tikzcd}$$

subject to the relations

$$s_0^1 s_0^0 = s_1^1 s_0^0 \quad d_0^0 s_0^0 = \mathrm{id}_1 \quad d_0^1 s_0^1 = \mathrm{id}_2 \quad d_1^1 s_1^1 = \mathrm{id}_2 \quad d_0^0 d_0^1 = d_0^0 d_1^1 \tag{9}$$

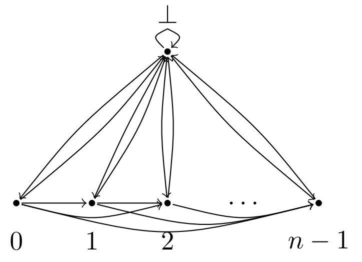
(see Proposition 4). We write $I : \mathcal{G} \to \mathcal{L}$ for the embedding and consider the associated nerve functor $N_I : \mathcal{L} \to \hat{\mathcal{G}}$. Suppose given an object $n \in \mathcal{L}$, the associated presheaf $N_I n$ can be described as follows. Its sets are

- $N_I n0 = \mathcal{L}(I0, n) \cong \{\star\}$,
- $N_I n1 = \mathcal{L}(I1, n) \cong [n] \uplus \{\bot\}$,
- $N_I n2 = \mathcal{L}(I2, n) \cong$
$\{(i, j) \in [n] \times [n] \mid i < j\} \uplus \{(\bot, i) \mid i \in [n]\} \uplus \{(i, \bot) \mid i \in [n]\} \uplus \{(\bot, \bot)\}$: a partial function $f : 2 \to n$ is characterized by the pair of images $(f(0), f(1))$ of $0, 1 \in [n]$, where $\bot$ means undefined.

and morphisms are

- $N_I n s_0^0 : N_I n1 \to N_I n0$ is the constant function whose image is $\star$,
- $N_I n d_0^0 : N_I n0 \to N_I n1$ is the function whose image is $\bot$,
- $N_I n s_0^1 : N_I n2 \to N_I n1$ is the second projection,
- $N_I n s_1^1 : N_I n2 \to N_I n1$ is the first projection,
- $N_I n d_0^1 : N_I n1 \to N_I n2$ sends $i \in [n] \uplus \{\bot\}$ to $(\bot, i)$
- $N_I n d_1^1 : N_I n1 \to N_I n2$ sends $i \in [n] \uplus \{\bot\}$ to $(i, \bot)$

Such a presheaf can be pictured as a graph with $N_I n1$ as set of vertices, $N_I n2$ as set of edges, source and target being respectively given by the functions $N_I n s_1^1$ and $N_I n s_0^1$:



Its vertices are elements of $[n] \uplus \{\bot\}$ and edges are of the form

- $i \to j$ with $i, j \in [n]$ such that $i < j$,
- $i \to \bot$ for $i \in [n]$
- $\bot \to i$ for $i \in [n]$
- $\bot \to \bot$

Morphisms are usual graphs morphisms which preserve the vertex $\bot$. We are thus naturally lead to define the following categories of pointed graphs and graphs with partial functions. We recall that a graph $G = (V, s, t, E)$ consists of a set V of vertices, a set E of edges and two functions $s, t : E \to V$ associating to each edge its source and its target respectively.

Definition 26. We define the category **pGraph** of *pointed graphs* as the category whose objects are pairs (G, x) with $G = (V, E)$ and $x \in V$ such that for every vertex there is exactly one edge from and to the distinguished vertex x, and morphisms $f : G \to G'$ are usual graph morphisms consisting of a pair (f_V, f_E) of functions $f_V : V_G \to V_{G'}$ and $f_E : E_G \to E_{G'}$ such that for every edge $e \in E_G$, $f_V(s(e)) = s(f_E(e))$ and $f_V(t(e)) = t(f_E(e))$, which are such that the distinguished vertex is preserved by f_V.

Definition 27. We define the category **PGraph** of *graphs and partial morphisms* as the category whose objects are graphs and morphisms $f : G \to G'$ are pairs (f_V, f_E) of partial functions $f_V : V_G \to V_{G'}$ and $f_E : E_G \to E_{G'}$ such that

- for every edge $e \in E_G$ such that $f_E(e)$ is defined, $f_V(s(e))$ and $f_V(t(e))$ are both defined and satisfy $f_V(s(e)) = s(f_E(e))$ and $f_V(t(e)) = t(f_E(e))$,
- for every edge $e \in E_G$ such that $f_V(s(e))$ and $f_V(t(e))$ are both defined, $f_E(e)$ is also defined.

More briefly: a morphism is defined on an edge if and only it is defined on its source and on its target.

Similarly to previous section, a partial morphism of graph can be seen as a pointed morphism of graph and conversely:

Proposition 28. *The categories* **pGraph** *and* **PGraph** *are equivalent.*

Now, notice that the category $\mathcal{L}$ is isomorphic to the full subcategory of **PGraph** whose objects are the graphs whose set of objects is $[n]$ for some $n \in \mathbb{N}$, and such that there is an edge $i \to j$ precisely when $i < j$. Also notice that the full subcategory of **pGraph** whose objects are the graphs $N_I n$ (with $\bot$ as distinguished vertex) with $n \in \mathbb{N}$ is isomorphic to the full subcategory of $\hat{\mathcal{G}}$ whose objects are the $N_I n$ with $n \in \mathbb{N}$. And finally, the two categories are equivalent via the isomorphism of Proposition 28. From this, we immediately deduce that the functor $N_I : \mathcal{L} \to \hat{\mathcal{G}}$ is full and faithful, i.e.

Proposition 29. *The functor $I : \mathcal{G} \to \mathcal{L}$ is dense.*

We can now follow Section 4 step by step, adapting each proposition as necessary. The conditions satisfied by presheaves in $\mathcal{P}$ introduced in Proposition 10 are still valid in our new case:

Proposition 30. *Given a presheaf $P \in \hat{\mathcal{L}}$ which is an object of $\mathcal{P}$,*

1. *the underlying graph of P is finite,*
2. *for each non-empty path $x \twoheadrightarrow y$ there exists exactly one edge $x \to y$,*
3. *$P(n+1)$ is the set of paths of length n in the underlying graph of P, and $P(0)$ is reduced to one element.*

Proof. The diagrams of the form (6) and (7) used in proof of Proposition 10 still admit the same colimit $n+1$ with the new definition of $\mathcal{L}$ and 0 is still initial. It can be checked that the limit of the image under a presheaf $P \in \hat{\mathcal{L}}$ of a diagram (6) is still the set of paths of length n in the underlying graph of P. □

Lemma 11 is also still valid:

Lemma 31. *A presheaf $P \in \hat{\mathcal{L}}$ preserves finite limits, if and only if it sends the colimits of diagrams of the form*

$$\mathrm{El}(G) \xrightarrow{\pi_G} \mathcal{G} \xrightarrow{I} \mathcal{L}$$

to limits in **Set***, where $G \in \hat{\mathcal{G}}$ is a finite pointed graph such that the above diagram admits a colimit. Such a diagram in $\mathcal{L}$ is said to be* generated by the pointed graph G.

Proof. The proof of Lemma 11 was done "by hand", but we mentioned a more abstract alternative proof. In the present case, a similar proof can be done but would be really tedious, so we provide the abstract one. In order to illustrate why we have to do so, one can consider the category of elements associated to the presheaves representable by 0 and 1, which are clearly much bigger than in the case of Section 4:

$$\begin{tikzcd}
\mathrm{El}(N_I 0) \arrow[r, phantom, "\cong"] & \star \arrow[r, shift left, "s_0^0"] & \bot \arrow[l, shift left, "d_0^0"] \arrow[r, shift left=3, "s_0^1"] \arrow[r, shift left=1, "s_1^1"] & (\bot,\bot) \arrow[l, shift left=1, "d_0^1"] \arrow[l, shift left=3, "d_1^1"]
\end{tikzcd}$$

and

$$\begin{tikzcd}
 & & & (\bot,\bot) \arrow[dl, shift left=1, "d_0^1"] \arrow[dl, shift left=3, "d_1^1"] \\
\mathrm{El}(N_I 1) \arrow[r, phantom, "\cong"] & & \bot \arrow[dl, shift left, "d_0^0"] \arrow[ur, shift left=3, "s_0^1"] \arrow[ur, shift left=1, "s_1^1"] \arrow[r, "s_1^1"] \arrow[dr, "s_0^1"] & (\bot,1) \arrow[dl, shift left, "d_0^1"] \\
 & \star \arrow[ur, shift left, "s_0^0"] \arrow[r, "s_0^0"'] & 1 \arrow[ur, shift left, "s_0^1"] \arrow[r, shift left, "s_1^1"] & (1,\bot) \arrow[l, shift left, "d_1^1"]
\end{tikzcd}$$

subject to relations which follow directly from (9).

Before going on with the proof, we need to introduce a few notions. A functor $F : \mathcal{C} \to \mathcal{D}$ is called *final* if for every category $\mathcal{E}$ and diagram $G : \mathcal{D} \to \mathcal{E}$ the canonical morphism $\mathrm{colim}(G \circ F) \to \mathrm{colim}(G)$ is an isomorphism [6]: restricting a diagram along F does not change its colimit. Alternatively, these functors can be characterized as functors such that for every object $D \in \mathcal{D}$ the category D/F is non-empty and connected (there is a zig-zag of morphisms between any two objects). A functor $F : \mathcal{C} \to \mathcal{D}$ is called a *discrete fibration* if for any object $C \in \mathcal{C}$ and morphism $g : D \to FC$ in $\mathcal{D}$ there exists a unique morphism $f : C' \to C$ in $\mathcal{C}$ such that $Ff = g$ called the lifting of g. To any such discrete fibration one can associate a presheaf $P \in \hat{\mathcal{D}}$ defined on any $D \in \mathcal{D}$ by $PD = F^{-1}(D) = \{C \in \mathcal{C} \mid FC = D\}$ and on morphisms $g : D' \to D$ as the function Pg which to $C \in PD$ associates the source of the lifting of g with codomain C. Conversely, any presheaf $P \in \hat{\mathcal{D}}$ induces a discrete fibration $\mathrm{El}(P) \xrightarrow{\pi} \mathcal{D}$, and these two operations induce an equivalence of categories between the category $\hat{\mathcal{D}}$ and the category of discrete fibrations over $\mathcal{D}$. It was shown by Paré, Street and Walters [8, 11] that any functor $F : \mathcal{C} \to \mathcal{D}$ factorizes as final functor $J : \mathcal{C} \to \mathcal{E}$ followed by a discrete fibration $K : \mathcal{E} \to \mathcal{D}$, and this factorization is essentially unique: this is called the *comprehensive factorization* of a functor. More explicitly, the functor K can be defined as follows. The inclusion functor $\mathbf{Set} \to \mathbf{Cat}$ which send a set to the corresponding discrete category admits a left adjoint $\Pi_0 : \mathbf{Cat} \to \mathbf{Set}$, sending a category to its connected components (its set of objects quotiented by the relation identifying two objects linked by a zig-zag of morphisms). The discrete fibration part K above can be defined as $\mathrm{El}(P) \xrightarrow{\pi} \mathcal{D}$ where $P \in \hat{\mathcal{D}}$ is the presheaf defined by $P = \Pi_0(-/F)$. In this precise sense, every diagram F in $\mathcal{D}$ is "equivalent" to one which is "generated" by a presheaf P on $\mathcal{D}$ (we adopted this informal terminology in the article in order to avoid having to introduce too many categorical notions).

In our case, we can thus restrict to diagrams in $\mathcal{L}$ generated by presheaves on $\mathcal{L}$. Finally, since $I : \mathcal{G} \to \mathcal{L}$ is dense, we can further restrict to diagrams generated by presheaves on $\mathcal{G}$ by interchange of colimits. □

Lemma 13 applies almost as in Section 4: since the morphism $f : G \to N_I n$ (seen as a partial functions between graphs) has to satisfy the universal property of Lemma 12, by choosing for every vertex x of G a partial function $g_x : G \to N_I m$ which is defined on x (such a partial function always exists), it can be shown that the function f has to be total. The rest of the proof can be kept unchanged. Similarly, Proposition 14 applies with proof unchanged.

Finally, we have that

Theorem 32. *The free conservative finite cocompletion $\mathcal{P}$ of $\mathcal{L}$ is equivalent to the full subcategory of $\hat{\mathcal{L}}$ whose objects are presheaves P satisfying the conditions of Proposition 30. Since its objects P satisfy $I_*I^*(P) \cong P$, it can equivalently be characterized as the full subcategory of $\hat{\mathcal{G}}$ whose objects P are*

1. finite*,*
2. transitive*: for each non-empty path $x \twoheadrightarrow y$ there exists exactly one edge $x \to y$,*
3. pointed*: $P(0)$ is reduced to one element.*

From this characterization (which can easily be extended to the labeled case), along with the correspondence between pointed graphs and graphs with partial functions (Proposition 28), the category is shown to be equivalent to the category described in Theorem 20: the relation is defined on vertices x, y of a graph G by $x < y$ whenever there exists a path $x \twoheadrightarrow y$.

As in case of previous section, the forgetful functor **pGraph** $\to$ **Graph** admits a left adjoint, thus inducing a monad on **Graph**. The category **pGraph** is equivalent to the Kleisli category associated to this monad, which is closely related to the *exception monad* as discussed in [3].

E Modeling repositories

We briefly detail here the modeling of repositories evoked in Section 7. As explained in the introduction, repositories can be modeled as partially ordered sets of patches, i.e. morphisms in $\mathcal{L}$. Since some of them can be incompatible, it is natural to model them as particular labeled event structures.

Definition 33. An *event structure* $(E, \leq, \#)$ consists of a set E of *events*, a partial order relation $\leq$ on E and *incompatibility* relation on events. We require that

1. for any event e, the downward closure of $\{e\}$ is finite and
2. given e_1, e'_1 and e_2 such that $e_1 \leq e'_1$ and $e_1 \# e_2$, we have $e'_1 \# e_2$.

Two events e_1 and e_2 are *compatible* when they are not incompatible, and *independent* when they are compatible and neither $e_1 \leq e_2$ nor $e_2 \leq e_1$. A *configuration* x is a finite downward-closed set of compatible events. An event e_2 is a *successor* of an event e_1 when $e_1 \leq e_2$ and there is no event in between. Given an event e we write $\downarrow e$ for the configuration, called the *cause* of e, obtained as the downward closure of $\{e\}$ from which e was removed. A *morphism* of event structures $f : (E, \leq, \#) \to (E', \leq', \#')$ is an injective function $f : E \to E'$ such that the image of a configuration is a configuration. We write **ES** for the category of event structures.

To every event structure E, we can associate a *trace graph* $T(E)$ whose vertices are configurations and edges are of the form $x \xrightarrow{e} x \uplus \{e\}$ where x is a configuration such that $e \notin x$ and $x \uplus \{e\}$ is a configuration. A *trace* is a path $x \twoheadrightarrow y$ in this graph. Notice that two paths $x \twoheadrightarrow y$ are of the same length. Moreover, given two configurations x and y such that $x \subseteq y$, there exists necessarily a path $x \twoheadrightarrow y$. It can be shown that this operation provides a faithful embedding $T : \mathbf{ES} \to \mathbf{Graph}$ from the category of event structures to the category of graphs, which admits a right adjoint.

Example 34. An event structure with five events is pictured on the left (arrows represent causal dependencies and $\sim$ incompatibilities). The associated trace graph is pictured on the right.

$$\begin{tikzcd}
 & d \arrow[rrd, dash, squiggly] & & \\
b \arrow[ru] & & c \arrow[lu] \arrow[r, dash, squiggly] & c' \\
 & a \arrow[lu] \arrow[ru] \arrow[rru] & &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & & \{a,b,c,d\} & \\
 & & \{a,b,c\} \arrow[u, "d"] & \\
\{a,b,c'\} & \{a,b\} \arrow[l, "c'"'] \arrow[ru, "c"] & & \{a,c\} \arrow[lu, "b"'] \\
 & \{a,c'\} \arrow[lu, "b"] & \{a\} \arrow[l, "c'"] \arrow[lu, "b"'] \arrow[ru, "c"'] & \\
 & & \emptyset \arrow[u, "a"] &
\end{tikzcd}$$

Definition 35. A *categorical event structure* (E, λ) in a category $\mathcal{C}$ with an initial object consists of an event structure equipped with a *labeling* functor $\lambda : (TE)^* \to \mathcal{C}$, where $(TE)^*$ is the free category generated by the graph TE, such that $\lambda\emptyset$ is the initial object of $\mathcal{C}$ and the image under λ of every square

$$\begin{tikzcd}
 & z & \\
y_1 \arrow[ur, "e_2"] & & y_2 \arrow[ul, "e_1"'] \\
 & x \arrow[ul, "e_1"] \arrow[ur, "e_2"'] &
\end{tikzcd}$$

in TE is a pushout in $\mathcal{C}$.

The following proposition shows that a categorical event structure is characterized by a suitable labeling of events of E by morphisms of $\mathcal{C}$.

Proposition 36. *The functor λ is characterized, up to isomorphism, by the image of the transitions $\downarrow e \xrightarrow{e} \downarrow e \uplus \{e\}$.*

We can now define a *repository* to be simply a finite categorical event structure $(E, \leq, \#, \lambda : T(E) \to \mathcal{L})$. Such a repository extends to a categorical event structure $(E, \leq, \#_0, I \circ \lambda : T(E) \to \mathcal{P})$, where $\#_0$ is the empty conflict relation. The *state* S of such an event structure is the file obtained as the image $S = I \circ \lambda(E)$ of the maximal configuration: this is the file that the users is currently editing given his repository. Usual operations on repositories can be modeled in this context, for instance importing the patches of another repository is obtained by a pushout construction (the category of repositories is finitely cocomplete).