Graph Transformations

An Introduction
to the
Categorical Approach

Hans J. Schneider
Chapter 2

Categorical Notions

As we have seen in the introduction, graph grammars and graph transformations can be applied in very different areas of computer science. We may use them to describe control-flow diagrams, entity relationship diagrams, term rewriting systems, Petri nets, state-charts, parallel logic programs, actor systems, and other graphical techniques. But the details we have to take into consideration are different: Control-flow diagrams may be considered as labeled graphs in the usual sense. The rules applicable to entity relationship diagrams contain variables, which have to be replaced before application. Petri nets require relabeling, whereas state-charts use hierarchical graphs. Finally, actor systems may be represented by graphs the labeling alphabet of which is a set of term graphs.

Of course, it is possible to formally define transformation rules for each application and to study the properties of derivations and derivation sequences in each application area separately. But this is not only a laborious task, but also it may hide analogies. Therefore, we use a language that brings to the surface the common techniques. The definitions should include as much concepts as possible, i.e., the general assumptions must be as weak as possible, and therefore, the point at which we need additional assumptions becomes explicit.

Such a language is category theory. Its development can be seen as an answer to specialization in mathematics: It puts the results of many existing mathematical theories together and provides a more general perspective. In a category, we have objects brought into relationships between one another by morphisms. For each of the examples we have mentioned above, we can define a category the objects and morphisms of which describe its characteristics. All these categories inherit the definitions, theorems, and constructions we can give in the general setting. In addition, there may be special results valid only in some application areas. To avoid misunderstandings: This is not a textbook on category theory; we use its notions as a language to express concepts of graph transformations. Readers interested in category theory should read a suitable textbook. The author has used the textbooks by Herrlich and Strecker [48], MacLane [57], Pierce [72], and Barr and Wells [6]. The last part of the introduction into mathematical foundations of computer science by Ehrig, Mahr, Cornelius, Grosse-

1Readers with emphasis on programming may note the similarity to object-oriented programming.
Rhode, and Zeitz can also be recommended [36]. Finally, we mention the approach by Burstall and Rydeheard, who combine category theory with functional programming [8].

In this chapter, we present the basic definitions of a category and of a morphism. We discuss the properties of special classes of morphisms and the relations between them. The duality principle allows us to gain new results by simply reversing arrows. The definition of derivability, which will be given in the next chapter, is based on the pushout construction that is a special type of a colimit. Therefore, a special section is devoted to its properties. Of course, the examples illustrating the categorical concepts are sets, graphs, and hypergraphs.

\section{Categories, Sets, and Graphs}

As we have just mentioned, we need a concept as general as possible, and from mathematics, we borrow the notion of the category, which indeed assumes only the existence of identities and the law of associativity.

\begin{definition} [Category]
A category\[K = (\text{Obj}_K, (\text{Mor}_K(A,B) | A, B \in \text{Obj}_K), \cdot)\]
is given by a class of objects $\text{Obj}_K$ together with a family of pairwise disjoint classes of morphisms $\text{Mor}_K(A,B)$ and a composition operation $\cdot$ such that the following properties hold:

\begin{itemize}
  \item[(a)] Existence of composition:
    \[\forall A, B, C \in \text{Obj}_K)(\forall f \in \text{Mor}_K(A,B))(\forall g \in \text{Mor}_K(B,C)) \exists g \cdot f \in \text{Mor}_K(A,C)\]
  \item[(b)] Associativity of composition:
    \[\forall A, B, C, D \in \text{Obj}_K)\)
    \[\forall f \in \text{Mor}_K(A,B))(\forall g \in \text{Mor}_K(B,C))(\forall h \in \text{Mor}_K(C,D)) (h \cdot (g \cdot f) = (h \cdot g) \cdot f)\]
  \item[(c)] Existence of identities with respect to composition:
    \[\forall A \in \text{Obj}_K)(\exists \text{id}_A \in \text{Mor}_K(A,A))\]
    \[((\forall f \in \text{Mor}_K(A,B))(f \cdot \text{id}_A = f) \land (\forall g \in \text{Mor}_K(B,A))(\text{id}_A \cdot g = g))\]
\end{itemize}

For convenience, we introduce two functions relating the morphisms to objects: If $f \in \text{Mor}_K(A,B)$, we write $\text{dom}_K(f) = A$ and $\text{codom}_K(f) = B$. The first condition in the definition ensures that we can compose two morphisms $f$ and $g$ whenever the target object $\text{codom}_K(f)$ of the first morphism is the source object $\text{dom}_K(g)$ of the second. Please note that we write the second morphism of a composition $g \cdot f$ on the left-hand side! Most of our examples use functions as morphisms, and in these cases, this order corresponds to the usual notation $g(f(x))$. The law of associativity allows us to avoid parentheses. Condition (a) ensures that both sides of this equation exist.
Finally, we should mention that we have not a universal identity morphism, but a specific identity morphism is associated with each object, since for two different objects \(A\) and \(B\), the classes \(\text{Mor}_K(A, A)\) and \(\text{Mor}_K(B, B)\) are disjoint by definition; therefore, each identity morphism uniquely determines an object. Conversely, if we consider an object \(A\), its identity morphism \(\text{id}_A\) is unambiguous. This follows from the definition immediately: If we assume the existence of another identity morphism \(\text{id}'_A\), we get \(\text{id}'_A = \text{id}_A \cdot \text{id}_A = \text{id}_A\). For this reason, it would be possible to remove the objects from the definition and to use the identities instead.

It is more convenient to denote the morphisms from \(A\) to \(B\), i.e., the elements of \(\text{Mor}_K(A, B)\), by \(f : A \rightarrow B\) or by \(A \xrightarrow{f} B\). Sometimes, we write \(gf\) instead of \(g \cdot f\).

Now, we introduce some example categories that we need later on in our graph-transformation applications.

**Example 2.1.2 (\textit{Set}):**

Our first example is the category \(\text{Set}\) of sets. We choose the class of all sets as \(\text{Obj}_{\text{Set}}\), i.e., each object is a set. In our examples, we need only the finite sets, but the definitions apply to infinite sets, too. The class of morphisms \(\text{Mor}_{\text{Set}}(A, B)\) is given by the class of all mappings from \(A\) to \(B\), and the composition is defined in the usual way: \((f \cdot g)(x) = f(g(x))\). Trivially, the law of associativity is satisfied, and the identities are given by \((\forall A \in \text{Obj}_{\text{Set}})(\forall x \in A)(\text{id}_A(x) = x)\).

It is not necessary to give all the details of this example. The reader, however, is recommended to check all the conditions we have mentioned in Definition 2.1.1. The next two examples use the same class of objects as the previous one; they demonstrate that a category is not fully defined by giving its objects, but the morphisms must also be chosen, explicitly.

**Example 2.1.3 (\textit{Rel}):**

Again, we choose the class of all sets as \(\text{Obj}_{\text{Rel}}\). But now, the class of morphisms \(\text{Mor}_{\text{Rel}}(A, B)\) is the class of all relations between \(A\) and \(B\), i.e., the class of all subsets of \(A \times B\) where \(A \times B = \{(a, b) \mid a \in A \land b \in B\}\) is the Cartesian product of \(A\) and \(B\). As usual, we define that \((a, c) \in g \cdot f\) if and only if there is a \(b\) such that \((a, b) \in f\) and \((b, c) \in g\). The identity relation on \(A\) is \(\text{id}_A = \{(a, a) \mid a \in A\}\).

The readers should go and see for themselves that the law of associativity holds.

These two examples have very similar categorical properties. Now, we introduce another example that can be used to make some properties clearer because it does not satisfy them.

**Example 2.1.4 (\textit{Setincl}):**

As above, the class of objects \(\text{Obj}_{\text{Setincl}}\) is the class of all sets. But now, we define the morphisms in an unusual way: There is exactly one morphism in \(\text{Mor}_{\text{Setincl}}(A, B)\) if \(A \subseteq B\) holds true. Otherwise, \(\text{Mor}_{\text{Setincl}}(A, B)\) is empty. Existence of composition follows from \(A \subseteq B \land B \subseteq C \Rightarrow A \subseteq C\) and existence of identities from \(A \subseteq A\).

If we consider two objects \(A\) and \(B\) in \(\text{Obj}_{\text{Setincl}}\), then there exists exactly one morphism from \(A\) to \(B\) or no morphism at all. Please keep in mind this special property; we need it later on.
In some applications, we need sets which are allowed to contain multiple elements. Such a set is called a multiset. E.g., in \{a, a, b, b, b, c\}, there are two elements which can be chosen as \(a\), three elements with property \(b\), whereas we have only one possibility to find an element \(c\).

**Example 2.1.5 (\(\mathcal{Mset}\))**: 
The class of objects \(\text{Obj}_{\mathcal{Mset}}\) is the class of all multisets. Between two multisets \(A\) and \(B\), there is a morphism if and only if \(A \subseteq B\) holds in the sense of multiset inclusion, i.e., each element must occur in \(B\) at least as many times as in \(A\). Anything else is as in the previous example.

We have no morphism from \{\(a, a, b, b, c, c\)\} to \{\(a, a, b, b, b, c\)\}, because there is only one \(c\) in the second set, but there is a morphism from \{\(a, a, b, b, c\)\} to \{\(a, a, b, b, b, c\)\}. Please note that our definition of morphisms in \(\mathcal{Mset}\) refers to the inclusion property and therefore, it results in a yes/no-decision. Another possibility is to distinguish the elements from one another and to define morphisms as mappings preserving the "names". This category is studied in Exercise 2.6.2.

The categories we have introduced so far are the basis of the category of graphs and labeled graphs. We now introduce graphs and different versions of labeled graphs step by step.\(^2\)

**Definition 2.1.6 (Graph)**: 
A graph is a quadruple \(G = (E, V, s, t)\) with \(E, V \in \text{Obj}_{\text{Set}}\) and \(s, t \in \text{Mor}_{\text{Set}}(E, V)\). \(E\) is called the set of edges, \(V\) the set of nodes or vertices. Function \(s\) assigns a source node to each edge and \(t\) a target node.

In graph theory, edges are often defined as pairs of nodes making it unnecessary to explicitly define \(s\) and \(t\). Contrary to this, our version allows parallel edges between two nodes as the following example shows:

**Example 2.1.7**: 
\[
\begin{array}{c|cccccc}
E & a & b & c & d & f & g \\
V & 1 & 2 & 3 & 4 & 5 & 6 \\
E & s & 1 & 1 & 1 & 3 & 2 \\
t & 2 & 2 & 3 & 2 & 3 & 5 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
1 & \rightarrow & 2 & \rightarrow \\
\rightarrow & \rightarrow & \rightarrow & \rightarrow \\
1 & 2 & 3 & 4 \\
\end{array}
\]

On the left-hand side, we have the formal definition of the graph. The nodes are numbered, the edges are identified by letters, and the functions \(s\) and \(t\) are given by a table defining the source node and the target node of each edge. On the right-hand side, the same graph is depicted in a more intuitive way. In most cases, we use this way to represent graphs.

Very often, we are not interested in the identifiers of the nodes and edges. Then, we use something like this:

\(^2\)In a later chapter, we shall present a unified definition.
Example 2.1.7 (Cont’d):

The graph we have just considered is not connected. Here is a connected graph:

Morphisms between graphs map nodes and edges in a structure preserving way:

Definition 2.1.8 (Graph morphism):

A graph morphism \( f : G \to H \) is a pair \( f = (f_E : E_G \to E_H, f_V : V_G \to V_H) \) of mappings such that \( f_V \cdot s_G = s_H \cdot f_E \land f_V \cdot t_G = t_H \cdot f_E \).

This condition can be illustrated graphically:

\[
\begin{align*}
E_G & \xrightarrow{s_G} V_G & E_G & \xrightarrow{t_G} V_G \\
E_H & \xrightarrow{s_H} V_H & E_H & \xrightarrow{t_H} V_H \\
f_E & = f_V & f_E & = f_V
\end{align*}
\]

It means that the image of the source node of an edge is the source node of the image of the edge, and the same holds true for the target node.

Example 2.1.7 (Cont’d):

We can map the first graph of our example to the second by the following graph morphism:

\[
\begin{array}{cccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
f_V(v) & 11 & 12 & 13 & 11 & 13 & 13 \\
\end{array}
\begin{array}{cccccccc}
 & a & b & c & d & f & g \\
f_E(e) & u & u & v & w & z & v \\
\end{array}
\]

If we map edge \( g \), e.g., to \( v \), we must have \( f_V(4) = 11 \land f_V(5) = 13 \) to satisfy the condition of Definition 2.1.8.

This assignment is not the only way of constructing a morphism from the first graph to the second. E.g., you may map node 6 to any other node because there are no edges connected to it.

Theorem 2.1.9 (Graph):

The class of graphs and the class of graph morphisms together with componentwise composition: \( g \cdot f = (g_E \cdot f_E, g_V \cdot f_V) \) constitute the category Graph.

---

3Throughout this book, \( E_G \) denotes the edges of graph \( G \), \( E_H \) the edges of \( H \), \( E' \) the edges of \( G' \), \( E_1 \) the edges of \( G_1 \), etc.
Proof:

We have to show that defining composition in this way yields a graph morphism again, that composition is associative, and that there exist identities. We consider two graph morphisms $f : G \to G'$ and $g : G' \to G''$. By definition, we have the left-hand diagram in $\text{Set}$:

$$
\begin{array}{cccc}
E_G & \xrightarrow{s_G} & V_G \\
\downarrow{f_E} & = & \downarrow{f_V} \\
E_{G'} & \xrightarrow{s_{G'}} & V_{G'} \\
\downarrow{g_E} & = & \downarrow{g_V} \\
E_{G''} & \xrightarrow{s_{G''}} & V_{G''}
\end{array}
\quad \implies
\begin{array}{cccc}
E_G & \xrightarrow{s_G} & V_G \\
\downarrow{g_E f_E} & = & \downarrow{g_V f_V} \\
E_{G''} & \xrightarrow{s_{G''}} & V_{G''}
\end{array}
$$

and we can take advantage of the properties of morphisms in $\text{Set}$:

$$
s_{G''} \cdot (g_E \cdot f_E) = (s_{G''} \cdot g_E) \cdot f_E = (g_V \cdot s_{G'}) \cdot f_E = g_V \cdot (s_{G'} \cdot f_E)
$$

$$
g_V \cdot (f_V \cdot s_G) = (g_V \cdot f_V) \cdot s_G
$$

Thus, we get commutativity of the right-hand diagram, i.e., the definition of composition satisfies definition of graph morphisms. This proof is a simple example of a technique we shall use again and again throughout this book. The trick is to replace a sequence of equations by constructing a commutative diagram from commutative subdiagrams by removing inner arrows. The reader should train this technique by proving that $\text{id}_G = (\text{id}_{E_G}, \text{id}_{V_G})$ is the identity of $G$.\(^4\)

Associativity follows from the associativity of components, immediately:

$$
(h \cdot g) \cdot f = ((h_E \cdot g_E) \cdot f_E, (h_V \cdot g_V) \cdot f_V) = (h_E \cdot (g_E \cdot f_E), h_V \cdot (g_V \cdot f_V)) = h \cdot (g \cdot f)
$$

Now, we switch over to labeled graphs, i.e., graphs the nodes and/or the edges of which are labeled with the elements of a certain alphabet $L$. More precisely, we assume $L = (L_E, L_V)$ to be a pair of sets the first component of which is used to label the edges and the second to label the nodes.

**Definition 2.1.10** (Labeled graph):

A labeled graph (or $L$-graph) is a sixtuple $G = (E, V, s, t, l_E, l_V)$ where $(E, V, s, t)$ is a graph and $l_E : E \to L_E$ and $l_V : V \to L_V$ are two set morphisms.

It is necessary to distinguish these labels from the identifiers we have used in Example 2.1.7.

\(^4\)For convenience, we write $\text{id}_{E_G}$ instead of $\text{id}_{E_G}$, etc.
Example 2.1.11:
In this example, due to Parisi-Presicce [67], we consider the following labeling alphabet:

\[ L_V = \{mary, joan, jane, dora, david\} \]
\[ L_E = \{mother of, sister of, female, male\} \]

In the following, the edge labels are abbreviated to \( mo, so, f, m \):

\[
\begin{array}{c}
mary \\
\downarrow \text{mo} \\
\downarrow \text{so} \\
\downarrow \text{f} \\
\text{jane} \\
\downarrow \text{mo} \\
\downarrow \text{so} \\
\downarrow \text{f} \\
\text{dora} \\
\downarrow \text{mo} \\
\downarrow \text{so} \\
\downarrow \text{f} \\
\text{david} \\
\end{array}
\]

The formal definition of this graph is:

\[
\begin{array}{c|c}
v & l_V \\
1 & mary \\
2 & joan \\
3 & jane \\
4 & dora \\
5 & david \\
\end{array}
\begin{array}{c|c}
\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c}
e & s & t & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\
1 & 1 & 2 & 3 & 3 & 3 & 4 & 1 & 2 & 3 & 4 & 5 & \\
2 & 1 & 2 & 3 & 3 & 3 & 4 & 1 & 2 & 3 & 4 & 5 & \\
3 & 2 & 3 & 3 & 2 & 4 & 5 & 5 & 1 & 2 & 3 & 4 & 5 & \\
4 & 2 & 3 & 3 & 2 & 4 & 5 & 5 & 1 & 2 & 3 & 4 & 5 & \\
5 & 2 & 3 & 3 & 2 & 4 & 5 & 5 & 1 & 2 & 3 & 4 & 5 & \\
\end{array}
\end{array}
\begin{array}{c|c}
l_E & mo so so mo so f f f f m \\
\end{array}
\]

In this example, we identify both nodes and edges by numbers. Labeling of the nodes is an injective function \( l_V \), i.e., there are no two nodes carrying the same label. This should not tempt the reader into confusing the sets \( V \) and \( L_V \). The labeling of edges shows that injective labeling functions are not the normal case. Nevertheless, we often omit the identification of nodes and edges to simplify the graphical representation, as we have done here. If it is necessary to mention the identifiers in the diagram explicitly, we use the notation \( \text{labelidentifier} \).

**Example 2.1.11 (Cont’d):**

Then, the above example looks like this:

\[
\begin{array}{c}
mary^1 \\
\downarrow \text{mo}^1 \\
\downarrow \text{so}^3 \\
\downarrow \text{f}^9 \\
\text{dora}^5 \\
\downarrow \text{mo}^5 \\
\downarrow \text{so}^7 \\
\downarrow \text{f}^{11} \\
\text{david}^5 \\
\end{array}
\]

\[
\begin{array}{c}
\text{f}^8 \\
\text{mo}^2 \\
\text{so}^3 \\
\text{mo}^6 \\
\text{m}^{12} \\
\end{array}
\]
In many applications, graph transformation may also include changing the labels. Nevertheless, we start with considering morphisms between labeled graphs that preserve the labels, i.e., the image of a node takes the same label as the node itself, and the same condition holds true for edges:

**Definition 2.1.12 (L-graph morphism):**

An L-graph morphism \( f : G \to H \) is a graph morphism \( f = (f_E : E_G \to E_H, f_V : V_G \to V_H) \) which additionally satisfies: \( l_{EH} \cdot f_E = l_{EG} \land l_{VH} \cdot f_V = l_{VG} \).

This condition can be illustrated graphically:

The difference between these two diagrams is the labeling of the arrows in the middle. The left-hand diagram describes the situation with respect to the source-node relation, whereas the right-hand side establishes the analogous property of the target-node relation. This duality is typical of all variations of graphs, and therefore, it is usual to draw only one diagram.

**Example 2.1.11 (Cont’d):**

There is an intuitive L-graph morphism from the following graph into the graph we have already considered:

Mapping the nodes and the edges is simply defined by the identifiers, i.e., node 1 is mapped onto node 1, etc.

**Theorem 2.1.13 (L-graph):**

The set of L-graphs and the set of L-graph morphisms together with componentwise composition constitute the category \( \mathcal{L}_{\text{graph}} \).

**Proof:**

We have to show only that componentwise composition satisfies the labeling condition of Definition 2.1.12. As we have just mentioned, it is sufficient to consider the source-node relation:
It is easy to see that this condition immediately follows from commutativity of the triangles on the left-hand side and on the right-hand side, e.g.,

\[ l_{EG''} \cdot (g_E \cdot f_E) = (l_{EG''} \cdot g_E) \cdot f_E = l_{EG'} \cdot f_E = l_{EG} \]

\[ \square \]

### 2.2 Hypergraphs

Let \( A \) be a set that we call an alphabet in this context. Then, \( A^* \) denotes the set of all words composed of the elements of \( A \) including the empty word \( \varepsilon_A \).

**Definition 2.2.1 (Word morphism):**

Each set morphism \( f : A \to B \) defines a word morphism \( f^* : A^* \to B^* \) by the following equations:

\[
\begin{align*}
  f^*(\varepsilon_A) &= \varepsilon_B \\
  f^*(a) &= f(a) \quad \forall a \in A \\
  f^*(xa) &= f^*(x)f(a) \quad \forall x \in A^*, a \in A
\end{align*}
\]

As a consequence, we get \( f^*(xy) = f^*(x)f^*(y) \) for all \( x, y \in A^* \).

**Theorem 2.2.2 (Word):**

The class of sets of words and the class of word morphisms together with composition \( g^* \cdot f^* = (g \cdot f)^* \) and identities \( (\text{id}_A)^* = \text{id}_{A^*} \) constitute the category \( \text{Word} \).

Proving the associativity of composition and the properties of identities is straightforward.

What is the relationship between \( \text{Set} \) and \( \text{Word} \)? Each set of words is a set and each word morphism is a set morphism, but not vice versa. This leads us to the notion of a subcategory:

**Definition 2.2.3 (Subcategory):**

A category \( \mathcal{K} \) is called a **subcategory** of \( \mathcal{K}' \) if the following conditions hold:
(a) \( \text{Obj}_K \subseteq \text{Obj}_{K'} \).

(b) \((\forall A, B \in \text{Obj}_K)(\text{Mor}_K(A, B) \subseteq \text{Mor}_{K'}(A, B)) \)

(c) \((\forall A, B, C \in \text{Obj}_K)((\forall f \in \text{Mor}_K(A, B)))(\forall g \in \text{Mor}_K(B, C))(g \cdot_K f = g \cdot_{K'} f) \)
where \( \cdot_K \) and \( \cdot_{K'} \) denote the compositions in \( K \) and \( K' \), respectively.

(d) \( \forall A \in \text{Obj}_K \): The identity on \( A \) in \( K \) is the identity on \( A \) in \( K' \), too.

It is called a full subcategory if in (b), the equality holds true instead of \( \subseteq \).

\( \text{Word} \) is a subcategory of \( \mathcal{S}et \), but it is not a full subcategory because there are set morphisms from a set of words \( A^* \) into another set of words \( B^* \) that do not fulfill the conditions of Definition 2.2.1.

In a graph, each edge visits two nodes: one node it starts from and another node which is its target. Now, we introduce hypergraphs the edges of which are allowed to visit an arbitrary number of nodes some of them are considered as source nodes and others as target nodes. Our examples show that the order of the nodes may be important. Therefore, we use the objects of \( \text{Word} \) to define the source and the target functions:

\textbf{Definition 2.2.4 (Hypergraph):} 
A hypergraph is a quadruple \( H = (E, V, s, t) \) with \( E, V \in \text{Obj}_{\mathcal{S}et} \) and \( s, t : E \to V^* \).

From a theoretical point of view, it is not necessary to distinguish \( s \) and \( t \); it is sufficient to associate one sequence of nodes with each edge. This consideration would result in the following alternative:

\textbf{Definition 2.2.5:} 
A hypergraph is a triple \( H = (E, V, s) \) with \( E, V \in \text{Obj}_{\mathcal{S}et} \) and \( s : E \to V^* \).

Especially, the examples of labeled hypergraphs will show us that from an application point of view, it makes sense to distinguish \( s \) and \( t \). Therefore, we prefer the first definition.

The main problem in drawing hypergraphs is how to depict the edges visiting a sequence of nodes. A possible solution is to represent a hyperedge by a rectangle connected to the nodes it visits. We use arrows labeled with \( s_1, s_2, \ldots \) and \( t_1, t_2, \ldots \) to characterize the order of source nodes and of target nodes, respectively. These arrows point from the source nodes to the hyperedge and from the hyperedge to the target nodes. In many practical applications such as Example 2.2.10, this is the most intuitive representation.\(^5\)

\textbf{Example 2.2.6:}
We consider two simple hypergraphs:

\(^5\)Other possibilities will be mentioned in the chapter on hypergraph replacement. In the case of Definition 2.2.5, lines without orientation are sufficient.
Hypergraph $H_1$ consists of four hyperedges $e_1$, $e_2$, $e_3$, and six nodes $v_1$, $v_2$, $v_3$, $v_4$, $v_5$, $v_6$. Edges $e_1$ and $e_2$ visit two source nodes and one target node each, $e_3$ has one source node and one target node, and $e_4$ has only a target node, but no source node: $s(e_4) = \varepsilon$, $t(e_4) = v_4$. In hypergraph $H_2$, edge $e_1'$ visits node $v_1'$ twice, namely as the first source node ($s_1$) and as the target node ($t_1$): $s'(e_1') = v_1'v_2'$ and $t'(e_1') = v_1'$.

**Definition 2.2.7** (Hypergraph morphism):
A hypergraph morphism $f : H \rightarrow H'$ is a pair $f = \langle f_E : E \rightarrow E', f_V : V \rightarrow V' \rangle$ such that $f_V^* \cdot s = s' \cdot f_E \wedge f_V^* \cdot t = t' \cdot f_E$.

Again, we illustrate the condition graphically:

$$
\begin{array}{c}
E \xrightarrow{s} V^* \\
\downarrow \quad f_E \\
E' \xrightarrow{s'} V'^*
\end{array}
\quad
\begin{array}{c}
E \xrightarrow{t} V^* \\
\downarrow \quad f_E \\
E' \xrightarrow{t'} V'^*
\end{array}
$$

Let us consider an edge of $H$. It visits a well-defined sequence of source (target) nodes. The condition means that the image of the edge visits the images of these nodes in the same order.

**Example 2.2.6** (Cont’d):
We define a hypergraph morphism $f$ from hypergraph $H_1$ to hypergraph $H_2$:

<table>
<thead>
<tr>
<th>$e$</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>$e_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_E(e)$</td>
<td>$e_1'$</td>
<td>$e_2'$</td>
<td>$e_3'$</td>
<td>$e_4'$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$v$</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
<th>$v_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_V(v)$</td>
<td>$v_1'$</td>
<td>$v_2'$</td>
<td>$v_3'$</td>
<td>$v_4'$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This mapping is neither injective nor surjective: Both $e_1$ and $e_2$ are mapped to $e_1'$, whereas $e_2'$ is not the image of any edge of hypergraph $H_1$. It is easy to check the condition; we consider edge $e_2$ as an example:

$$
\begin{align*}
&f_V^*(s_1(e_2)) = f_V^*(v_4v_5) = v_1'v_2' \\
&s_2(f_E(e_2)) = s_2(e_4') = v_1'v_2' \\
&f_V^*(t_1(e_2)) = f_V^*(v_2) = v_1' \\
&t_2(f_E(e_2)) = t_2(e_4') = v_1'
\end{align*}
$$
Since $f_V$ maps the nodes $v_4$ and $v_2$ onto the same node $v_1'$, the image edge $e_1'$ has to visit this node twice.

With the morphisms defined in this way, we get:

**Theorem 2.2.8 (Hgraph):**

The class of hypergraphs and the class of hypergraph morphisms together with componentwise composition $g \cdot f = (g_E \cdot f_E, g_V \cdot f_V)$ constitute the category $\mathcal{H}g\text{raph}$.

The proof is very similar to that of Theorem 2.1.9 and, therefore, left to the reader.

The analogy between the definition of hypergraphs and the definition of graphs (Def. 2.1.6) suggests to treat these notions as special cases of a general construction. We postpone this until we have more experience in the basic constructions.

Now, we consider labeled hypergraphs. Again, we assume a pair $L = (L_E, L_V)$ of labeling alphabets. In most application areas, the edge label determines the number, order, and labels of the nodes that an edge visits. Formally, we can reduce this to a relationship between edge labels and sequences of node labels, number and order are then given implicitly:

**Definition 2.2.9 (Hypergraph labeling):**

A hypergraph labeling consists of a pair $L = (L_E, L_V)$ of labeling alphabets together with two functions $\tau_s, \tau_t : L_E \to L_V$.

Some applications do not require explicit node labels. Then, it is sufficient to map the edge labels into the natural numbers. If we do not distinguish between $s$ and $t$, we also have only one $\tau$.

**Example 2.2.10:**

A signature $\Sigma = (S, OP, \tau_s, \tau_t)$ consists of a set of sorts, a set of operation symbols, and two functions $\tau_s : OP \to S^*$ and $\tau_t : OP \to S$. $\tau_s$ defines the number, order, and sorts of the parameters of an operation symbol, and $\tau_t$ the sort of its result. Let us consider the abstract data type queue as an example:

$$
\begin{align*}
S & = \{bool, elmt, queue\} \\
OP & = \{init, empty, next, appd, remove\}
\end{align*}
$$

<table>
<thead>
<tr>
<th>$\tau_s(op)$</th>
<th>init</th>
<th>appd</th>
<th>remove</th>
<th>empty</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_t(op)$</td>
<td>queue</td>
<td>queue elmt</td>
<td>queue</td>
<td>queue</td>
<td>queue</td>
</tr>
</tbody>
</table>

An operation represented, e.g., by the symbol $appd$, which abbreviates the operation to append an element to a given queue, needs two parameters of the sorts $queue$ and $elmt$, respectively, and yields a result of type $queue$.

We can easily translate a signature into a hypergraph, the so-called signature graph [7, p. 52]:

---

6It is not necessary to treat these cases formally, since in the first case, we may consider all nodes labeled identically, and in the second, we may use a $t$ defined by $t(e) = \varepsilon$ for all hyperedges $e$. 
We label the hyperedges with the operation symbols and the nodes with the sorts.

We draw labeled hypergraphs analogously to labeled graphs. A rectangle denotes a hyperedge, the label of which is put into the rectangle in roman characters, whereas the label of a node is not surrounded and is set in italics. In most cases, it is not necessary to identify nodes and edges explicitly; if it is, we use numbers written as exponents.

The example shows a strong relationship between the edge labels and the node labels: A hyperedge labeled with `appd` must visit a source node labeled with `queue` and another one labeled with `elmt`; the target node must be labeled with `queue`. Our definition of a labeled hypergraph reflects this relationship between edge labels and node labels:

**Definition 2.2.11** (Labeled hypergraph):

A labeled hypergraph (or L-hypergraph) is a sixtuple $H = (E, V, s, t, l_E, l_V)$ where $(E, V, s, t)$ is a hypergraph and $l_E : E \to L_E$ and $l_V : V \to L_V$ are set morphisms satisfying $\tau_s \cdot l_E = l_V \cdot s_E$ and $\tau_t \cdot l_E = l_V \cdot t_E$.

**Example 2.2.10** (Cont’d):

We resume the data type example and consider how to represent terms. Let $V = \bigcup_{s \in S} V_s$ be a class of disjoint sets of variables. Then, the set $T(\Sigma, V) = \bigcup_{s \in S} T_s(\Sigma, V)$ is defined inductively:

(a) If $x \in V_s$, then $x \in T_s(\Sigma, V)$.
(b) If $\tau_s(op) = s_1 s_2 \ldots s_n$ and $t_i \in T_s(\Sigma, V)$ for $i = 1, 2, \ldots, n$, then $op(t_1, t_2, \ldots, t_n) \in T_{\tau(op)}(\Sigma, V)$.

An example is the term $remove(appd(appd(init,x_1),x_2))$ with $x_1, x_2 \in V_{elmt}$. In the figure, $H_2$ is the hypergraph representation of this term. The variables do not occur in the graph explicitly. Hypergraphs describing how to compute the values of these variables may be connected to the nodes 4 and 6 the labels of which (elmt) define the sorts of the variables. $H_3$ represents a computation which takes advantage of common subexpressions:

\[
\begin{align*}
f &= appd(appd(init, x), next(appd(q, x))) \\
g &= appd(remove(appd(q, x)), next(appd(q, x)))
\end{align*}
\]
As we have already discussed, the arrows point from the source nodes to the hyperedge and from the hyperedge to the target nodes. If we interpret the hyperedges as operations, this is an intuitive representation.\footnote{In papers on term graph rewriting, the arrows are often depicted in the reverse direction.}

**Definition 2.2.12** (L-hypergraph morphism):  
An L-hypergraph morphism \( f : G \to H \) is given by a hypergraph morphism \( f = (f_E : E_G \to E_H, f_V : V_G \to V_H) \) additionally satisfying: \( l_{EH} \cdot f_E = l_{EG} \wedge l_{VH} \cdot f_V = l_{VG} \).

We can graphically interpret this condition in the following way:

\[
\begin{align*}
E_G & \xrightarrow{s_G} V_G^* \\
E_H & \xrightarrow{s_H} V_H^* \\
E_G & \xrightarrow{f_E} V_H^* \\
E_H & \xrightarrow{f_V} V_H^* \\
l_{EG} & = l_{EH} = l_{VH} \\
l_{EG} & = l_{EH} = l_{VH} \\
l_{EG} & = l_{EH} = l_{VH} \\
l_{EG} & = l_{EH} = l_{VH} \\
l_{EG} & = l_{EH} = l_{VH} \\
l_{EG} & = l_{EH} = l_{VH} \\
l_{EG} & = l_{EH} = l_{VH} \\
l_{EG} & = l_{EH} = l_{VH} \\
\end{align*}
\]

As usual, the analogous diagram holds true for the target nodes. Of course, \( l_{VH} \cdot f_V = l_{VG} \) implies \( l_{VH} \cdot f_V = l_{VG} \) by Definition 2.2.1.

**Example 2.2.10** (Cont’d):  
We can construct an L-hypergraph morphism from graph \( H_2 \) to graph \( H_1 \) by mapping all hyperedges onto the hyperedge of graph \( H_1 \) bearing the same label:

\[
\begin{array}{cccc}
e & 1 & 2 & 3 & 4 \\
\hline
f_E(e) & 1 & 3 & 3 & 2 \\
\end{array}
\]  
\[
\begin{array}{cccc}
v & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
f_V(v) & 1 & 1 & 1 & 2 & 1 & 2 \\
\end{array}
\]
In this special case, mapping the nodes is uniquely determined, because in graph $H_1$, each edge label occurs exactly once. We consider another example:

\[
\begin{array}{c}
\text{queue}^1 \\
\text{appd}^1 \\
\text{queue}^2 \\
\text{elmt}^3
\end{array}
\]

There are two possibilities to map hypergraph $H_4$ into hypergraph $H_2$, because there are two hyperedges in $H_2$ labeled with \textit{appd}; both hyperedges visit nodes compatible to the nodes of $H_4$:

\[
\begin{array}{c|c}
e & 1 \\f_E(e) & 2 \\
v & 1 & 2 & 3 \\
f_V(v) & 2 & 3 & 4
\end{array}
\]

or

\[
\begin{array}{c|c}
e & 1 \\
f'_E(e) & 3 \\
v & 1 & 2 & 3 \\
f'_V(v) & 3 & 5 & 6
\end{array}
\]

Although we use the same numbers to identify both hyperedges and nodes, there should be no misunderstanding.

\textbf{Theorem 2.2.13 ($\mathcal{LHG}graph$):}

The class of L-hypergraphs and the class of L-hypergraph morphisms together with componentwise composition constitute the category $\mathcal{LHG}graph$.

The proof exactly follows the proof of Theorem 2.1.13. Some very small changes in that diagram yield the diagram we need here. The reader should carry out these changes to keep in practice.

\section*{2.3 A Hierarchy of Morphisms}

We know from set theory that there are functions with distinguished properties, e.g., injections and surjections, and that we can prove relationships between them. A typical example is the fact that a function that is both injective and surjective is bijective, too. Such properties are usually defined or proved in terms of elements of the sets. The aim of this section is to generalize these concepts such that we can also apply them to other categories.

\textbf{Definition 2.3.1 (Monomorphism):}

If a category $\mathcal{C}$ is given, a $\mathcal{C}$-morphism $f : B \to A$ is called a \textit{monomorphism} if for all $\mathcal{C}$-objects $C$ and all $\mathcal{C}$-morphisms $g, h : C \to B$ with $f \cdot g = f \cdot h$, it follows that

\[g = h:\]

\[(\forall C)(\forall g, h : C \to B)(f \cdot g = f \cdot h \Rightarrow g = h)\]

This formulation is a little bit cumbersome because of the many repetitions of $C$. We therefore state that all our definitions and propositions assume the existence of a category $\mathcal{C}$ and that all morphisms and objects mentioned are arbitrary morphisms and objects of this category, if we do not state a special property explicitly. Another abbreviation is to simply write: $(\forall g, h)(f \cdot g = f \cdot h \Rightarrow g = h)$. This is sufficient since
codom\( (g) = \text{codom}(h) \) follows from the composition and \( \text{dom}(g) = \text{dom}(h) \) follows from the equality.

We may interpret the definition of monomorphism graphically:

\[
\begin{array}{c}
A \xrightarrow{f} B \\
\downarrow \quad \downarrow \\
C \xrightarrow{g} C
\end{array} \quad \Rightarrow \quad g = h
\]

**Lemma 2.3.2 (Monomorphisms in \( \text{Set} \))**:

The monomorphisms in the category \( \text{Set} \) coincide with the injective functions.

**Proof**:

First, we show that injective functions satisfy the defining property of monomorphisms. We assume that \( f \) is injective and that for all functions \( g \) and \( h \), we have \( f \cdot g = f \cdot h \), i.e., \( \forall c \in C \Rightarrow f(g(c)) = f(h(c)) \). Since \( f \) is injective, this means \( g(c) = h(c) \), and \( g \) and \( h \) denote the same function. Conversely, we assume that \( f \) satisfies the definition of monomorphism and that there exist elements \( a, a' \in A \) with \( f(a) = f(a') \). We consider two functions \( g, h : C \to A \), defined by \( g(c) := a \) and \( h(c) := a' \) for all \( c \in C \). Then, we get \( f(g(c)) = f(a) = f(a') = f(h(c)) \) for all \( c \), i.e., \( f \cdot g = f \cdot h \) and therefore, \( g = h \) because of the definition. This means that \( a \) is equal to \( a' \) and \( f \) is injective.

Now, it is not surprising that some characteristics of injections also apply to monomorphisms and can be proved without referring to sets:

**Lemma 2.3.3 (Composition of monomorphisms)**:

The composition of morphisms satisfies:

(a) If \( f, g \) are monomorphisms, then \( g \cdot f \) is a monomorphism.

(b) If \( g \cdot f \) is a monomorphism, then \( f \) is a monomorphism.

**Proof**:

To prove (a), we have to show that for all morphisms \( h \) and \( k \), \( h = k \) follows from \( (g \cdot f) \cdot h = (g \cdot f) \cdot k \):

\[
\begin{align*}
(g \cdot f) \cdot h &= (g \cdot f) \cdot k \\
g \cdot (f \cdot h) &= g \cdot (f \cdot k) \quad \text{(associativity)} \\
f \cdot h &= f \cdot k \quad \text{(\( g \) is a monomorphism)} \\
h &= k \quad \text{(\( f \) is a monomorphism)}
\end{align*}
\]

We prove part (b) of the lemma by a sequence of diagrams:
The definition of monomorphism requires that we can conclude \( h = k \) from diagram (i). Thus, we start from this diagram, and we add the morphism \( g : B \to C \). Then, we have \( g \cdot f \) (or \( gf \) for short) both in the upper half and in the lower half of the diagram (ii). We can remove the inner arrows because of commutativity, and we get diagram (iii), from which \( h = k \) follows because \( gf \) is a monomorphism by assumption.  

We have presented this proof in all the details to illustrate the proof techniques. The diagrams are more intuitive, and therefore, we prefer this technique. The reader, however, must be aware of the formalism behind the diagrams (Exercise 2.6.6).

Analogously, we introduce a concept generalizing surjective functions (Exercise 2.6.7):

**Definition 2.3.4** (Epimorphism):
A morphism \( f : A \to B \) is called an epimorphism if for all \( g, h : B \to C \) with \( g \cdot f = h \cdot f \), it follows that \( g = h \):

\[
(\forall g, h : B \to C)(g \cdot f = h \cdot f \Rightarrow g = h)
\]

Or graphically:

\[
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow & \quad = \quad & \downarrow \\
B & \xrightarrow{h} & C
\end{array}
\quad \Rightarrow \quad g = h
\]

Comparing Definitions 2.3.1 and 2.3.4, we see that we obtain the second definition from the first by simply reversing all arrows:

**Principle 2.3.5** (Duality principle):
If we have defined a categorical notion, then reversing all arrows in the definition yields the dual notion. If we have proved a categorical property, then reversing all arrows in specifying the property yields the dual property.

This principle is based on the notion of dual categories that is considered in Exercise 2.6.8. Especially, we have here: The notion of epimorphism is the dual notion of monomorphism, and vice versa. Applying this principle to Lemma 2.3.3, we immediately get the following lemma:

**Lemma 2.3.6** (Composition of epimorphisms):
The composition of morphisms satisfies:
(a) If \( f, g \) are epimorphisms, then \( f \cdot g \) is an epimorphism.

(b) If \( f \cdot g \) is an epimorphism, then \( f \) is an epimorphism.

Many categorical propositions hold true only “up to isomorphisms”. This means that there may be several solutions, but the differences are not of interest, e.g., if we construct a graph, we may number the nodes and the edges in some way or other without affecting the properties we are interested in.

**Definition 2.3.7 (Isomorphism):**

A morphism \( f : A \rightarrow B \) is called an **isomorphism** if a morphism \( g : B \rightarrow A \) exists such that \( f \cdot g = \text{id}_B \) \land \( g \cdot f = \text{id}_A \).

This notion is self-dual, i.e., if you reverse the arrows, you get the same definition again. Furthermore, morphism \( g \) is unambiguous. Assume another morphism \( g' : B \rightarrow A \) satisfying the same condition: \( f \cdot g' = \text{id}_B \) \land \( g' \cdot f = \text{id}_A \). Then, we get \( g = g' \) by the following simple transformation: \( g' = g' \cdot \text{id}_B = g' \cdot (f \cdot g) = (g' \cdot f) \cdot g = \text{id}_A \cdot g = g \). Thus, it makes sense to write \( f^{-1} \) instead of \( g \).

In \( \text{Set} \), the isomorphisms are the bijective functions. It is well-known that each bijection is injective as well as surjective:

**Lemma 2.3.8:**

Every isomorphism is both a monomorphism and an epimorphism.

Let \( f \) be an isomorphism and \( f^{-1} \) its inverse. If we have \( f \cdot g = f \cdot h \), we get \( f^{-1} \cdot f \cdot g = f^{-1} \cdot f \cdot h \) and \( g = h \) and therefore, \( f \) is a monomorphism. The second part of the lemma follows from the duality principle.

In the category \( \text{Set} \), we have the converse proposition, too: If a function is both injective and surjective, it is bijective. But, this does not hold true for arbitrary categories:

**Example 2.3.9:**

We consider the category \( \text{Set}_{\text{incl}} \) introduced in Example 2.1.4. If we have \( f \cdot g = f \cdot h \), then \( g \) and \( h \) must be parallel morphisms, i.e., \( \text{dom}(g) = \text{dom}(h) \) and \( \text{codom}(g) = \text{codom}(h) \). \( g \) and \( h \) must be the same morphism in each case, since in this category, there is at most one morphism between two objects, but this is not an isomorphism unless the domain and the codomain coincide.

Therefore, we need an additional notion in order to get an anlogous converse:

**Definition 2.3.10 (Retraction and coretraction):**

(a) A morphism \( f : A \rightarrow B \) is said to be a retraction if there exists a morphism \( g : B \rightarrow A \) such that \( f \cdot g = \text{id}_B \).

(b) A morphism \( f : A \rightarrow B \) is said to be a coretraction if there exists a morphism \( g : B \rightarrow A \) such that \( g \cdot f = \text{id}_A \).

Retractions and coretractions are dual notions. Trivially, if \( f \) is a retraction, there exists at least one corectraction \( g : B \rightarrow A \), and vice versa. To see this, you should interchange \( f \) with \( g \) and \( A \) with \( B \) in (b). Let us consider an example in \( \text{Set} \):
Example 2.3.11:

Let $f$ be a set morphism with $f : \{a, b, c\} \to \{d, e\}$, defined by $f(a) = d, f(b) = e, f(c) = e$. Then, we have two coretractions: $g(d) = a, g(e) = b$ and $g'(d) = a, g'(e) = c$.

Lemma 2.3.12:

(a) Every isomorphism is both a retraction and a coretraction.
(b) If a morphism is a retraction as well as a coretraction, then it is an isomorphism, too.

Part (a) is trivial since the definition of isomorphism combines the conditions of the definitions of retractions and coretractions. If $f : A \to B$ is a retraction, we have at least one $g : B \to A$ such that $f \cdot g = \text{id}_B$. If $f$ is a coretraction, too, we get another $g' : B \to A$ such that $g' \cdot f = \text{id}_A$. These morphisms, however, are equal: $g = (g' \cdot f) \cdot g = g' \cdot (f \cdot g) = g'$, and therefore, $f$ is an isomorphism. We can weaken the assumptions in (b) and characterize an isomorphism by being both a retraction and a monomorphism or a coretraction and an epimorphism (Exercise 2.6.9). This is indeed a weakening:

Lemma 2.3.13:

(a) Every retraction is an epimorphism.
(b) Every coretraction is a monomorphism.

We prove the second part. For each coretraction $f : A \to B$, we have a morphism $f' : B \to A$ such that $f' \cdot f = \text{id}_A$. We assume $f \cdot g = f \cdot h$. From this, we get $f' \cdot f \cdot g = f' \cdot f \cdot h$ and therefore, $g = h$.

We can join these lemmata together to give a first version of a hierarchy of morphisms:

Theorem 2.3.14 (Hierarchy of morphisms):

We can arrange the morphisms hierarchically:

(a) Isomorphisms $\subseteq$ retractions $\subseteq$ epimorphisms.
(b) Isomorphisms $\subseteq$ coretractions $\subseteq$ monomorphisms.

In the category $\text{Set}$, these inclusions are strict. Example 2.3.11 shows that retractions and coretractions need not be bijections, which are the isomorphisms of $\text{Set}$. Furthermore, we have already seen that the monomorphisms coincide with the injective functions (Lemma 2.3.2). If we have $f(a) = b$, we can define $g(b) = a$. This works if we have an $a$, but there are some pathological examples. Let $B$ be a nonempty set. Then, $f : \emptyset \to B$ is injective: it does not map different elements onto the same because there are no elements to map. On the other hand, it is not possible to find any function that maps a nonempty $B$ to the empty set. Hence, the set morphism $f$ is a monomorphism, but not a coretraction.

In Definition 2.1.8, we have introduced graph morphisms as pairs of set morphisms, and we have defined composition componentwise. We can, therefore, derive the properties of graph morphisms from that of set morphisms:

\footnote{We shall complete this hierarchy in Theorem 2.4.15.}
Theorem 2.3.15 (Properties of graph morphisms):

(a) If \( f = (f_E, f_V) \) is a coretraction in \( \mathcal{G}raph \), then \( f_E \) and \( f_V \) are coretractions in \( \mathcal{S}et \).

(b) If both components \( f_E \) and \( f_V \) of a graph morphism \( f = (f_E, f_V) \) are monomorphisms in \( \mathcal{S}et \), then \( f = (f_E, f_V) \) is a monomorphism in \( \mathcal{G}raph \).

Proof:

If \( f \) is a coretraction, there exist \( g_E \) and \( g_V \) such that we have \( (g_E, g_V) \cdot (f_E, f_V) = (id_E, id_V) \). By definition of composition in \( \mathcal{G}raph \), this is identical to \( g_E \cdot f_E = id_E \land g_V \cdot f_V = id_V \), and \( f_E \) and \( f_V \) are coretractions. To prove the second part, we have to consider two graph morphisms \( g = (g_E, g_V) \) and \( h = (h_E, h_V) \) such that \( (f_E, f_V) \cdot (g_E, g_V) = (f_E, f_V) \cdot (h_E, h_V) \). Decomposing this equation into the components, we get \( f_E \cdot g_E = f_E \cdot h_E \) and \( f_V \cdot g_V = f_V \cdot h_V \). Since the components of \( f \) are assumed to be monomorphisms, we have \( g_E = h_E \) and \( g_V = h_V \) and we complete the proof by putting these results together: \( (g_E, g_V) = (h_E, h_V) \). \( \square \)

In \( \mathcal{S}et \), coretractions and monomorphisms (injections) are nearly the same. The morphisms in the difference are not of practical interest. This has led to some carelessness: In many papers, you find morphisms explicitly assumed to be injective (= monomorphic), but afterwards the inverse morphism is constructed, which only exists if we assume a coretraction. In \( \mathcal{G}raph \), however, the difference is more serious: Many monomorphisms in \( \mathcal{G}raph \) are not coretractions. Here is an example:

Example 2.3.16:

\[
\begin{array}{ccc}
1 & \rightarrow & 2 \\
\downarrow & & \downarrow f \\
3 & \rightarrow & 2'
\end{array}
\]

\[
\begin{array}{ccc}
1' & \rightarrow & 2' \\
\downarrow & & \downarrow \\
3' & \rightarrow & 4'
\end{array}
\]

We have numbered the nodes, and \( f_V \) is given by \( 1 \mapsto 1', 2 \mapsto 2', 3 \mapsto 3' \). We have omitted the identifiers of the edges because \( f_E \) can unambiguously be constructed from \( f_V \). Both mappings are injective, and therefore, \( f \) is a monomorphism. But, it is not a coretraction since there is no way of finding an inverse morphism. Of course, it is possible to construct an \( f_V^{-1} \): You can map node 4' to any of the nodes \( \{1, 2, 3\} \); however, it is impossible to map the edges \( 3' \rightarrow 4' \) and \( 4' \rightarrow 2' \) to edges of the left-hand graph such that the definition of graph morphism is satisfied. (If we map 4' to 3, we need a cycle at node 3; if we map it to 2, we need a cycle there; and if we map it to 1, we need an edge 3 \( \rightarrow \) 1.)

2.4 Colimits

An advantage of category theory is that it provides us with general constructions applicable in different areas. Let us consider the smallest common multiple, the dual notion of the greatest common divisor: We start with two numbers \( n_1 \) and \( n_2 \) and define the set of all numbers that are multiples both of \( n_1 \) and of \( n_2 \); then, we consider
the smallest element of this set. A similar example is definition of the union of two sets $M_1$ and $M_2$. Again, all sets that include $M_1$ as well as $M_2$ are admitted to the competition, and we select the smallest. Readers familiar with the programming language Prolog or some modern functional language such as ML or HASKELL know another example: If there are two terms $t_1$ and $t_2$, we may substitute new terms for the variables in $t_1$ and $t_2$ (Exercise 2.6.4). A unifier is a substitution that makes $t_1$ and $t_2$ identical. The most general unifier is the “smallest”, i.e., the substitution from which all other unifiers can be reached by an additional substitution.

What is the common kernel of these examples? We start with two objects $A_1$ and $A_2$, and we define the set of all objects reachable from $A_1$ and $A_2$ by morphisms $f'_1$ and $f'_2$, respectively. Then, we ask for a distinguished object in this set from which we can reach all other elements:

**Definition 2.4.1** (Coproduct):

The coproduct of a pair of objects $(A_1, A_2)$ is a triple $(f_1, f_2, C)$ where $C$ is an object and $f_1 : A_1 \rightarrow C$ and $f_2 : A_2 \rightarrow C$ are morphisms such that

$$(\forall C')(\forall f'_1 : A_1 \rightarrow C')(\forall f'_2 : A_2 \rightarrow C')(\exists ! u : C \rightarrow C')(f'_1 = u \cdot f_1 \land f'_2 = u \cdot f_2)$$

We say that $K$ has coproducts provided that there exists a coproduct for every pair of objects in $\text{Obj}_K$. Sometimes the coproduct object is denoted by $C = A_1 \oplus A_2$.

The coproduct is a special case of a colimit; we shall come back to this point later on. The condition can be interpreted graphically:

![Diagram](a)

Of course, uniqueness of $u$, in the formula indicated by $\exists !$, is understood up to isomorphism and should not be confused with uniqueness of $C$ and the $f_i$:

**Lemma 2.4.2** (Uniqueness of coproduct):

The coproduct of $(A_1, A_2)$ is unambiguous up to isomorphism.

Proof:

We assume existence of two solutions $(f_1, f_2, C)$ and $(f'_1, f'_2, C')$, and we arrange them as shown in diagram (a):

![Diagram](b)
Since \((f_1, f_2, C)\) is a coproduct, there is a unique \(u : C \to C'\) such that \(u \cdot f_1 = f'_1 \wedge u \cdot f_2 = f'_2\). Conversely, from \((f'_1, f'_2, C')\) being a coproduct, we get a unique \(u' : C' \to C\) with \(u' \cdot f'_1 = f_1 \wedge u' \cdot f'_2 = f_2\). Thus, we have a morphism \(u' \cdot u : C \to C\) with \(u' \cdot u \cdot f_1 = f_1\) and \(u' \cdot u \cdot f_2 = f_2\). On the other hand, the coproduct diagram \((b)\) says that a morphism with this property must be unambiguous. But there is a well-known morphism with this property: the isomorphism of \(C\). Therefore, \(u' \cdot u\) must equal \(id_C\). By exchanging the roles of \(C\) and \(C'\) in the left-hand diagram, we also find \(u \cdot u' = id_C\).

Lemma 2.4.3 (Coproduct in \(\mathsf{Set}\)):
The coproduct in \(\mathsf{Set}\) is the disjoint union together with the natural injections.

First, we show that this construction has the coproduct property. Let \(A_1\) and \(A_2\) be two sets, and let \(C\) be their disjoint union. For reasons of simplicity, we denote \(C\) by \(\{f_1(x) \mid x \in A_1\} \cup \{f_2(x) \mid x \in A_2\}\). For \(i = 1, 2\), we consider two arbitrary mappings \(f'_i : A_i \to C\), and we define

\[
u(c) := \begin{cases} f'_1(x) & \text{if } c \in f_1[A_1] \land c = f_1(x) \\ f'_2(x) & \text{if } c \in f_2[A_2] \land c = f_2(x) \end{cases}
\]

By construction, we have \(f'_i = u \cdot f_i\). To show that \(u\) is unique, we assume existence of a \(u'\) with \(u' \cdot f_i = f'_i \wedge u' \neq u\). This means that there is a \(c \in C\) with \(u'(c) \neq u(c)\) and an \(a \in A_i\) with \(f_i(a) = c\) for either \(i = 1\) or \(i = 2\). Thus, we get a contradiction: \(u(c) = u(f_i(a)) = f'_i(a) = u'(f_i(a)) = u'(c)\). Therefore, the disjoint union is a coproduct of \(A_1\) and \(A_2\), and this is the only solution because of Lemma 2.4.2.

Example 2.4.4:
We consider two sets \(A_1 = \{1, 2, 3\}\) and \(A_2 = \{1, 2\}\). The coproduct is the set \(C = \{1', 2', 3', 1'', 2''\}\) together with the mappings \(f_1(i) = i'\) and \(f_2(i) = i''\).

The coproduct construction is the first example of a colimit construction; it starts from two objects. Now, we consider colimits starting from morphisms:

Definition 2.4.5 (Coequalizer):
A morphism \(q : B \to C\) is called the coequalizer of a pair of parallel morphisms \(f, g : A \to B\) if and only if the following conditions hold:

\[(a)\quad q \cdot f = q \cdot g\]
\[(b)\quad (\forall C')(\forall q' : B \to C')(q' \cdot f = q' \cdot g \Rightarrow (\exists u : C \to C')(q' = u \cdot q))\]

A category is said to have coequalizers if every pair of parallel morphisms has a coequalizer.

Again, we illustrate this factorization condition graphically:

\[
\begin{array}{cccc}
A & \overset{f}{\rightarrow} & B & \overset{g}{\rightarrow} & C \\
\downarrow & & \downarrow & & \downarrow \\
& & & & u \\
& & & & C' \\
& \downarrow & \downarrow & \downarrow & \\
& q' & = & u & \\
\end{array}
\]
The proof that the coequalizer is unique up to isomorphism is left to the reader (Exercise 2.6.10).

**Lemma 2.4.6 (Coequalizer in Set):**
In $\text{Set}$, we can construct the coequalizer of two morphisms $f, g : A \to B$ as a system of equivalence classes:

(a) Consider the relation $R := \{(f(x), g(x)) \mid x \in A\}$.

(b) Construct the finest equivalence relation $\tilde{R}$ containing $R$, i.e.,

\[
\begin{align*}
(y, z) \in R & \implies (y, z) \in \tilde{R} \\
(y, z) \in \tilde{R} \land (z, w) \in \tilde{R} & \implies (y, w) \in \tilde{R} \\
(y, z) \in \tilde{R} & \implies (z, y) \in \tilde{R}
\end{align*}
\]

(c) Define $C := \{[y] \mid y \in B\}$ and $q(y) := [y]$
where $[y]$ denotes the class of all elements equivalent to $y$.

By construction, we have $q(f(x)) = [f(x)] = [g(x)] = q(g(x))$ and therefore, $q \cdot f = q \cdot g$.
If we have a $q' : B \to C'$ with $q' \cdot f = q' \cdot g$, we can define $u$ by $u([y]) := q'(y)$. If we had another $u'$ with $u \cdot q = u' \cdot q$, we get $u = u'$ because by construction, $q$ is a surjection, i.e., an epimorphisms. Finally, $u$ is well-defined, i.e., the definition does not depend on the choice of $y$. The formal proof is a little bit cumbersome because we have to take into consideration sequences of relation steps in $R$. We present only the basic step: If $(y, y') \in R$, there exists an $x \in A$ such that $y = f(x) \land y' = g(x)$. From this, we get $u([y]) = q(y) = q'(f(x)) = q'(g(x)) = q'(y') = u([y'])$.

\[\square\]

**Example 2.4.7:**
We consider a simple example. Let $f$ and $g$ be given by the following table:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$f(i)$</th>
<th>$g(i)$</th>
<th>$i'$</th>
<th>$q(i')$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1'</td>
<td>1'</td>
<td>1'</td>
<td>$[1', 2', 3']$</td>
</tr>
<tr>
<td>2</td>
<td>2'</td>
<td>1'</td>
<td>2'</td>
<td>$[1', 2', 3']$</td>
</tr>
<tr>
<td>3</td>
<td>3'</td>
<td>2'</td>
<td>3'</td>
<td>$[1', 2', 3']$</td>
</tr>
<tr>
<td>4</td>
<td>4'</td>
<td>4'</td>
<td>4'</td>
<td>$[4']$</td>
</tr>
</tbody>
</table>

Definitions of $f$ and $g$ immediately yield $R = \{(1', 1'), (2', 1'), (3', 2'), (4', 4')\}$. From $2' \sim 1'$ and $3' \sim 2'$, we also get $3' \sim 1'$ by transitivity, and we have $C = \{[1', 2', 3'], [4']\}$. The right-hand diagram illustrates these mappings.

Whereas the coequalizer is the colimit of two parallel morphisms, our next colimit construction starts from two morphisms with different codomains:

**Definition 2.4.8 (Pushout):**
A commutative square $\tilde{g} \cdot f = \tilde{f} \cdot g$ of morphisms is called the pushout diagram of $(f, g)$ if and only if the following condition holds:

\[
(\forall D') (\forall \tilde{g}' : B \to D') (\forall \tilde{f}' : C \to D')
(\tilde{g}' \cdot f = \tilde{f}' \cdot g \implies (\exists u : D \to D') (\tilde{g}' = u \cdot \tilde{g} \land \tilde{f}' = u \cdot \tilde{f}))
\]
The pushout construction is the principal constituent of our approach to graph transformations. Therefore, we discuss the properties of pushout diagrams in a special section. Examples in various categories are also postponed. Here, we mention only the uniqueness:

**Lemma 2.4.9 (Uniqueness of pushout):**

The pushout of \((f : A \to B, g : A \to C)\) is unambiguous up to isomorphism.

It is not necessary to go into the details of the proof since it is analogous to the proof of Lemma 2.4.2. Consider the diagram of Definition 2.4.8 and assume the outer diagram to be another pushout diagram. Then, drawing the inner diagram outside of the \(D'\)-diagram, we get a \(u' : D' \to D\), and we find that \(u' \cdot u = \text{id}_D\) as in the case of coproducts. Exchanging the inner and the outer diagram completes the proof.

Comparing the definitions of coproduct, coequalizer, and pushout, we can give a generalized construction:

**Definition 2.4.10 (Colimits):**

We start with a diagram consisting of objects \(A_i\) and some morphisms \(f_k : A_i \to A_j\) between these objects.

(a) A *cococone* of this diagram is an object \(X'\) together with, for each \(A_i\), a morphism \(h_i : A_i \to X'\) such that for all morphisms \(f_k : A_i \to A_j\) in the given diagram, the triangle \(h_i = h_j \cdot f_k\) commutes.

(b) The *colimit* of the diagram is a distinguished cococone with object \(X\) such that for any cococone with object \(X'\), there is a unique morphism \(u\) from \(X\) to \(X'\) and the morphisms \(A_i \to X'\) can be factorized into \(A_i \to X\) and \(u\).

The table on top of the next page summarizes the colimit constructions we have considered up to now. The first column shows the diagrams we start with, the second contains an example of a cococone, and the last column illustrates the colimit property.

All colimits are unambiguous up to isomorphism. The proof is analogous to the proofs we have already seen.

**Definition 2.4.11 (Cocomplete categories):**

A category having colimits of all (finite) diagrams is said to be (finitely) cocomplete.
Arbitrary finite colimits can be constructed inductively. We do not prove this theorem, but we illustrate the idea behind the proof by considering the ternary coproduct of three objects (Exercise 2.6.11) and the canonical construction of pushouts (Theorem 2.5.6).

The coproduct is an example of a colimit the base diagram of which does not contain any morphism. In such cases, we may use arbitrary morphisms \( h_i : A_i \to X' \) in constructing a cocone, since the commutativity condition is satisfied trivially. Of course, we can also define a colimit with a base diagram that does not contain any objects at all:

**Definition 2.4.12** (Initial object):

An object \( A \) is called the **initial object** of category \( \mathcal{K} \) provided that for all objects \( A' \in \text{Obj}_{\mathcal{K}} \), \( \text{Mor}_{\mathcal{K}}(A, A') \) has exactly one member.

This definition immediately follows from the construction of colimits if we start with the empty diagram. A cocone of this diagram consists only of an object and every object \( A' \) of the category is admitted to the competition, since there is no need to find morphisms between given objects and \( A' \). Thus, unambiguity of \( u \) requires that there is exactly one morphism from \( A \) to every object in \( \mathcal{K} \).

**Example 2.4.13** (Initial object in \( \text{Set} \)):

The initial object in \( \text{Set} \) is the empty set. There is exactly one way of mapping the elements of the empty set into another set, namely to do nothing.
The initial object plays an important role in our context. We may consider the cocones over a given diagram as objects of a category and define suitable morphisms between the cocones. Then the colimit of the diagram can be defined as the initial object of this category (see, e.g., [8, Sect. 3.4]).

So far, we have considered only the colimit constructions. The duality principle allows us to reduce investigation of limit constructions to a short summary, since reversing all arrows leads to these notions.

**Definition 2.4.14 (Limits):**

The dual concepts of coproduct and coequalizer are the **product** and the **equalizer**, respectively. The dual concept of pushout is the **pullback** and the dual concept of initial object is the **terminal object**.

Applying the systematic way to denote the dual concept of $C$ by $\text{co-}C$, we had to use co-pullback instead of pushout and co-terminal object instead of initial object, but we prefer the usual terms.

In $\text{Set}$, the product is the Cartesian product (Exercise 2.6.14). This is the reason to call this construction the product and its dual construction the coproduct.

All the limits and all the colimits are unique up to isomorphism. The terminal object in $\text{Set}$ is well-suited to illustrate what “uniqueness up to isomorphism” means. The terminal object is the set consisting of exactly one element: You have only one way of mapping other sets onto the singleton set. We intuitively distinguish the set $\{a\}$ from the set $\{b\}$ if $a$ and $b$ are different, but this is not a structural difference since we can map $a$ to $b$ and vice versa. Thus, the name we use to denote an element does not play an important role. The (co-)limit constructions yield unambiguous solutions up to renaming the elements, but this is known to every programmer: The meaning of a program does not depend on the choice of the identifiers.

We conclude this section by completing the hierarchy of morphisms (Theorem 2.3.14).

**Theorem 2.4.15 (Hierarchy of morphisms):**

We can arrange the morphisms hierarchically:

(a) $\text{Isomorphisms} \subseteq \text{retractions} \subseteq \text{coequalizers} \subseteq \text{epimorphisms}.$

(b) $\text{Isomorphisms} \subseteq \text{coretractions} \subseteq \text{equalizers} \subseteq \text{monomorphisms}.$

**Proof:**

First, we assume $f : A \to B$ to be a retraction. Then, there is a coretraction $g : B \to A$ such that $f \cdot g = \text{id}_B$. We show that $f$ is the coequalizer of $(g \cdot f, \text{id}_A)$. We consider the following the diagram:

$$
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow{g} & & \downarrow{f} \\
\text{id}_A & \xrightarrow{g \cdot f} & A \\
\end{array}
$$

February 9, 2011
$q'$ must satisfy $q' \cdot \text{id}_A = q' = q' \cdot g \cdot f$. Please note that we can not deduce $g \cdot f = \text{id}_A$ from this because $q'$ need not be a monomorphism. We define $u$ by $u := q' \cdot g$, and we get $u \cdot f = q' \cdot g \cdot f = q'$. If we had another $u'$, $q' = u \cdot f = u' \cdot f$ results in $u = u'$, since $f$ is an epimorphism. (In other words, the definition of $u$ does not depend on which coretraction $g$ we have chosen.)

Now, we show that each coequalizer $q$ is an epimorphism, i.e., $(\forall f', g')(f' \cdot q = g' \cdot q \Rightarrow f' = g')$. Let $q$ be the coequalizer of some morphisms $f, g$. We consider a $q'$ defined by $q' := f' \cdot q = g' \cdot q$, and we arrange the morphisms in the following way:

This $q'$ satisfies the assumption of the coequalizer definition:

$$q' \cdot f = (f' \cdot q) \cdot f = f' \cdot (q \cdot f) = f' \cdot (q \cdot g) = (f' \cdot q) \cdot g = q' \cdot g$$

and therefore, there is exactly one morphism $u$ with $q' = u \cdot q$. From this, we get $u = f' = g'$.

2.5 Properties of Pushout Diagrams

The pushout construction is the basic feature of our approach to graph transformations. In the following chapters, we need some properties of this construction again and again, and therefore, it makes sense to study them in a special section. Most properties can be shown without referring to special categories. Nevertheless, we must also mention some properties that are typical of $\text{Set}$ and can not be applied to every category.

**Theorem 2.5.1** (Preservation theorem):

Pushout diagrams preserve epimorphisms and coretractions, i.e., if in a pushout diagram

$f$ is an epimorphism, then $\bar{f}$ is an epimorphism, and if $\bar{f}$ is a corectraction, then $\bar{f}$ is.

Proof:

First, we assume $f$ to be epimorphic, and we show that $\bar{f}$ is epimorphic, too, i.e., we have to prove that for each pair $(w_1, w_2)$ of morphisms, $w_1 = w_2$ follows from
$w_1 \cdot \bar{f} = w_2 \cdot \bar{f}$. We define $v := w_1 \cdot \bar{f} = w_2 \cdot \bar{f}$ and $u := w_1 \cdot \bar{g}$ as shown in Diagram (a). With these definitions, the outer diagram becomes commutative: $v \cdot g = w_1 \cdot \bar{f} \cdot g = w_1 \cdot \bar{g} \cdot f = u \cdot f$ and therefore, we can make use of the pushout property that gives us a unique $w$ such that $u = w \cdot \bar{g} \land v = w \cdot \bar{f}$. Since $w_1$ also fulfils this condition, we have $w = w_1$, but not yet $w = w_2$ because we could not deduce $u := w_2 \cdot \bar{g}$ from the construction. But we can get this from the fact that $f$ is an epimorphism:

\[ w_2 \cdot \bar{g} \cdot f = w_2 \cdot \bar{f} \cdot g = v \cdot g = u \cdot f = w_1 \cdot \bar{g} \cdot f \quad \Rightarrow \quad w_2 \cdot \bar{g} = w_1 \cdot \bar{g} = u. \]

Now, we consider a coretraction $f$, i.e., there exists a morphism $h$ such that $h \cdot f = \text{id}_A$ and we have to show the existence of a $u$ with $u \cdot \bar{f} = \text{id}_C$. We construct this $u$ as the universal morphism in Diagram (b). The outer diagram is commutative by construction: $\text{id}_C \cdot g = g = g \cdot \text{id}_A = g \cdot h \cdot f$ and therefore, we get a $u$ making commutative the triangles $g \cdot h = u \cdot \bar{g}$ and $\text{id}_C = u \cdot \bar{f}$.

Corollary 2.5.2:

Pullback diagrams preserve monomorphisms and rejections.

Of course, we gain this corollary by simply reversing the arrows in the diagram:

Since we prefer depicting arrows from left to right and from top to down, we get a backward inheritance of properties in pullback diagrams. The third diagram, which is identical to the second, shows this.

Please note that pushout diagrams preserve epimorphisms, but they do not preserve monomorphisms. The proof makes use of the stronger assumption that $f$ is a coretraction. We show by example that it is not sufficient to assume a monomorphic $f$:

Example 2.5.3:

We consider a category consisting of five objects $a, b, c, d, e$ and thirteen morphisms: Five identities (one for each object), two morphisms from $e$ to $c$: $h_1, h_2 : e \to c$, and exactly one morphism from $a \to b$, $a \to c$, $a \to d$, $b \to d$, $c \to d$, and $e \to d$:
Since there is only one morphisms $a \to d$, we have $\bar{f} \cdot g = \bar{g} \cdot f$. Similarly, we get $\bar{f} \cdot h_1 = \bar{f} \cdot h_2$, but $h_1 \neq h_2$, i.e., $\bar{f}$ is not a monomorphism. On the other hand, $f$ is monomorphic, since the only morphism that can be put in front of $f$ is the identity $\text{id}_a$.

We ask for the role of the preservation theorem in $\text{Set}$: Pushout diagrams lead from surjections to surjections, and from injections to injections if we neglect the special case of empty mappings that are injections, but are not coretractions. This case, however, also leads to injections, but we have to use another argument:

**Lemma 2.5.4:**

If in a pushout diagram

![Diagram](a)

object $I$ is the initial object, then $D$ together with $\bar{f}$ and $\bar{g}$ is the coproduct of $B$ and $C$.

The proof is left to the reader as an exercise (Exercise 2.6.18).

Since coproduct morphisms in $\text{Set}$ are injections, it is correct to say:

**Corollary 2.5.5** (Preservation theorem in $\text{Set}$):

In $\text{Set}$, both surjections and injections are preserved by the pushout construction.

Another important point is that we have a general procedure to construct pushout diagrams in various categories:

**Theorem 2.5.6** (Canonical construction of pushouts):

A category with coproducts and coequalizers has pushouts, too.

Proof:

Given two morphisms $f : A \to B$ and $g : A \to C$, we construct the pushout diagram as shown in Diagram (a):

(a) We construct the coproduct $(f' : B \to E, g' : C \to E, E)$ of $B$ and $C$.
(b) We construct the coequalizer $r : E \to D$ of $g'' := g' \cdot g$ and $f'' := f' \cdot f$.
(c) Finally, we define $\bar{f} := r \cdot g'$ and $\bar{g} := r \cdot f'$.

By construction, the outer diagram is commutative. We have to show only the universal property of pushouts. For this, we consider an arbitrary commutative diagram $\bar{f} \cdot g = \bar{g} \cdot f$ and we show that $\bar{f}'$ and $\bar{g}'$ can be factorized by a unique $u$ such that
In the next example, we consider two noninjective mappings $f$ and $g$. Again,
we consider the elements of $A$ one after another to decide which elements must be put together: 1 puts $a'$ and $\bar{a}$ into one class, 2 puts together $a'$ and $\bar{b}$, and 3 causes $c'$ and $\bar{a}$ to be in the same equivalence class.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$f(x)$</th>
<th>$g(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$a'$</td>
<td>$\bar{a}$</td>
</tr>
<tr>
<td>2</td>
<td>$a'$</td>
<td>$\bar{b}$</td>
</tr>
<tr>
<td>3</td>
<td>$c'$</td>
<td>$\bar{a}$</td>
</tr>
</tbody>
</table>

The examples in $\mathbf{Set}$ clarify an important difference between coproduct and pushout: Whereas the coproduct keeps separate the elements of the given sets, the pushout identifies distinguished elements.

Due to this construction, we can characterize pushouts in $\mathbf{Set}$ in the following way:

**Corollary 2.5.9 (Pushout in $\mathbf{Set}$):**

In $\mathbf{Set}$, a commutative diagram

$$
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow{g} & & \downarrow{\bar{g}} \\
C & \xrightarrow{f} & D
\end{array}
$$

is a pushout diagram if and only if

(a) $D = \bar{g}[B] \cup \bar{f}[C]$

(b) $d \in \bar{g}[B] \cap \bar{f}[C] \Rightarrow (\exists a \in A)(d = \bar{f}(g(a)) = \bar{g}(f(a)))$

(c) $\bar{g}(b) = \bar{g}(b') \Rightarrow b = b' \lor (\exists a_1, a_2, \ldots, a_n \in A)(b = f(a_1) \land g(a_1) = g(a_2) \land f(a_2) = f(a_3) \land \ldots \land g(a_{n-1}) = g(a_n) \land f(a_n) = b')$

(d) $\bar{f}(c) = \bar{f}(c') \Rightarrow c = c' \lor (\exists a_1, a_2, \ldots, a_n \in A)(c = g(a_1) \land f(a_1) = f(a_2) \land g(a_2) = g(a_3) \land \ldots \land f(a_{n-1}) = f(a_n) \land g(a_n) = c')$

Sometimes we need only the fact that $b, b'$ or $c, c'$ are in $f[A]$ or $g[A]$, respectively, as given by the first and the last term. Furthermore, conditions (c) and (d) become simpler if one of the morphisms is injective. Let, e.g., $f$ be an injection: In this case, (d) becomes trivial and in (c), we have $n = 2$. $f(a_{k-1}) = f(a_k)$ results in $a_{k-1} = a_k$ and therefore, $g(a_k) = g(a_{k+1}) = g(a_{k-1})$. This means that $g(a_1) = g(a_2) = \ldots = g(a_n)$ and it is sufficient to consider $a_1$ and $a_n$.

The corollary shows that the pushout object does not contain any elements that are neither images under $\bar{f}$ nor under $\bar{g}$. In general, neither $\bar{f}$ nor $\bar{g}$ is surjective, but

---

The $a_k$ need not be distinct.
together, they cover the pushout object. We can generalize this observation. In the canonical construction, \( D \) is the codomain object of the coequalizer \( r \), and each coequalizer is an epimorphism. We can, in some sense, transfer this property to the pair \((\bar{f}, \bar{g})\), saying that \( \bar{f} \) and \( \bar{g} \) together behave similar to an epimorphism:

**Definition 2.5.10** (Jointly epimorphic):

A pair \((f, g)\) of morphisms is called **jointly epimorphic** if and only if for all \(h_1, h_2\) with \(h_1 \cdot f = h_2 \cdot f \land h_1 \cdot g = h_2 \cdot g\), the equality \(h_1 = h_2\) follows.

As an immediate consequence of this definition, we get the following corollary:

**Corollary 2.5.11**:

The concept **jointly epimorphic** is closely related to **epimorphic**:

(a) \(f\) is epimorphic if and only if the pair \((f, f)\) is jointly epimorphic.

(b) If the pair \((f_2 \cdot f_1, g_2 \cdot g_1)\) is jointly epimorphic, then the pair \((f_2, g_2)\) is jointly epimorphic, too.

The second proposition of this corollary is analogous to Lemma 2.3.6. We shall take advantage of it in decomposing pushout diagrams, e.g., in proving the independence theorem (Theorem 5.3.2).

**Lemma 2.5.12**:

In a pushout diagram

\[
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow{g} & & \downarrow{\bar{g}} \\
C & \xrightarrow{f} & D \\
\end{array}
\]

the pair \((\bar{f}, \bar{g})\) is jointly epimorphic.

To show this, we start with \(h_1 \cdot \bar{f} = h_2 \cdot \bar{f} \land h_1 \cdot \bar{g} = h_2 \cdot \bar{g}\). We add the morphisms \(g\) and \(f\), respectively, and we get \(h_1 \cdot f \cdot g = h_2 \cdot \bar{f} \cdot \bar{g} \cdot f = h_1 \cdot \bar{g} \cdot f\). The pushout property yields \(h_1 = h_2\).

Let us now apply the canonical construction to the category \(\text{Setincl}\). As we have explained in Example 2.1.4, there is exactly one morphism from \(A\) to \(B\) if and only if \(A \subseteq B\).

**Example 2.5.13** (Colimits in \(\text{Setincl}\)):

The fact that morphisms coincide with inclusion allows a straightforward characterization of colimits in \(\text{Setincl}\):

(a) The initial object is the set that is contained in each set, i.e., the empty set.

(b) The coproduct object \(C\) is the object that is a superset of both \(A\) and \(B\) and is contained in all sets with this property, i.e., \(C\) is the union \(C = A \cup B\).

(c) The coequalizer is the identity since parallel morphisms are identical.

(d) From this, we get that the pushout coincides with the coproduct, even if object \(A\) is not the initial object.
The same result holds for the category of multisets with multiset inclusion as morphisms. Discussing multisets with morphisms to be mappings preserving the denotations of elements is left as an exercise (Exercise 2.6.19).

Now, we look at composed diagrams. Can we identify subdiagrams that satisfy the pushout property if we know something about the whole diagram and vice versa?

**Lemma 2.5.14 (Embedded pushouts):**

Suppose that the following diagram is commutative:

\[
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow{g} & & \downarrow{\bar{g}} \\
C & \xrightarrow{\bar{f}} & D
\end{array}
\]

(a) If \(\bar{g} \cdot f = \bar{f} \cdot g\) is a pushout diagram, then so is \(\bar{g} \cdot f' = \bar{f} \cdot g'\).

(b) If \(\bar{g} \cdot f' = \bar{f} \cdot g'\) is a pushout diagram and \(h\) is epimorphic, then \(\bar{g} \cdot f = \bar{f} \cdot g\) is a pushout diagram, too.

**Proof:**

To prove that the pushout property of the inner diagram follows from that of the outer diagram, we assume two morphisms \(\bar{g} \cdot f = \bar{f} \cdot g\) completing the pushout diagram. Since \(\bar{g} \cdot f = \bar{g} \cdot f' \cdot h = \bar{f} \cdot g' \cdot h = \bar{f} \cdot g\) also holds true, we can apply the pushout property of the outer diagram to get a unique \(u\) factorizing \(\bar{g} \cdot f\) and \(\bar{f} \cdot g\).

Proving the converse, we again start with morphisms \(\bar{g} \cdot f = \bar{f} \cdot g\), but now we have to suppose \(\bar{g} \cdot f' = \bar{f} \cdot g'\). We replace \(f\) and \(g\) in this equality: \(\bar{g} \cdot f' \cdot h = \bar{f} \cdot g' \cdot h\). At this point, we make use of the fact that \(h\) is epimorphic, and we get \(\bar{g} \cdot f' = \bar{f} \cdot g'\) that allows us to apply the pushout property of the inner diagram.

**Corollary 2.5.15 (Relationship between pushouts and pullbacks):**

Suppose that we have a pushout diagram \(\bar{g} \cdot f = \bar{f} \cdot g\):

\[
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow{g} & \xleftarrow{\text{PO}} & \downarrow{\bar{g}} \\
C & \xleftarrow{f} & D
\end{array}
\]

Then the pullback diagram of \(\bar{f}\) and \(\bar{g}\), i.e., \(\bar{g} \cdot f' = \bar{f} \cdot g'\), is also a pushout diagram.

This corollary immediately follows from the lemma because a suitable \(h\) can be constructed using the pullback property. It results in the observation that if we alternatingly construct embedded pushouts and pullbacks, the process converges very soon:

(a) Given \(f\) and \(g\), construct \(\bar{f}\) and \(\bar{g}\) completing the pushout diagram.

(b) Construct \(f'\) and \(g'\) as the pullback diagram of \(\bar{f}\) and \(\bar{g}\).

(c) Construct \(\bar{f}'\) and \(\bar{g}'\) as the pushout diagram of \(f',g'\). Then, we have \(\bar{f}' = \bar{f}\) and \(\bar{g}' = \bar{g}\).
Only the first two diagrams are different. Again, there is a special situation in the category $\text{Set}$: In this case, even the first two diagrams coincide, if an additional condition is satisfied:

**Lemma 2.5.16** (Relationship between pushouts and pullbacks in $\text{Set}$):

If the following diagram

\[
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow{g} & & \downarrow{\bar{g}} \\
C & \xrightarrow{\bar{f}} & D
\end{array}
\]

is a pullback diagram in $\text{Set}$ with monomorphic $\bar{f}$ and $\bar{g}$ and jointly epimorphic $(\bar{f}, \bar{g})$, then the diagram is a pushout diagram, too.

We insert the pushout diagram $\bar{g}' \cdot f = \bar{f}' \cdot g$ into the pullback diagram:

The pushout property yields a unique $u$ factorizing $\bar{f}$ and $\bar{g}$, and we prove that $u$ is an isomorphism in this special case. From the fact that $(\bar{f}, \bar{g})$ is a jointly epimorphic pair, we have that each element of $D$ is either in $\bar{g}[B]$ or in $\bar{f}[C]$ or in both. We define $u'(d) := \bar{g}'(d)$ if $d = \bar{g}(b)$ and $u'(d) := \bar{f}'(c)$ if $d = \bar{f}(c)$. Since $\bar{f}$ and $\bar{g}$ are injections, $b$ and $c$ are determined unambiguously. In the case $d \in \bar{g}[B] \cap \bar{f}[C]$, the definition is unambiguous, too, because there exists an element $a \in A$ with $f(a) = b$ and $g(a) = c$, and therefore, $\bar{g}'(b) = \bar{g}'(f(a)) = \bar{f}'(g(a)) = \bar{f}'(c)$. It is easy to see that $u \cdot u' = \text{id}_D$: Without loss of generality, we assume $d = \bar{g}(b)$, and we get $u(u'(d)) = u(\bar{g}'(b)) = \bar{g}(b) = d$. Conversely, we take advantage of the fact that the pushout object $D'$ is given by $\bar{g}'[B] \cup \bar{f}'[C]$. Consider, e.g., a $d' = \bar{g}'(b)$. Then, we have $u'(u(d')) = u'(\bar{g}'(b)) = \bar{g}'(b) = d'$.

In some papers, this lemma is mentioned without explicitly assuming the pair $(\bar{f}, \bar{g})$ to be jointly epimorphic. This is not sufficient. The solution to Exercise 2.6.17 is a counterexample. Moreover, it is also not sufficient that only one of the morphisms $\bar{f}$ and $\bar{g}$ is injective. Consider the following example:

\[
\begin{array}{ccc}
\emptyset & \xrightarrow{f} & \{b_1, b_2\} \\
\downarrow{g} & & \downarrow{\bar{g}} \\
\{c\} & \xrightarrow{\bar{f}} & \{[b_1, b_2], c\}
\end{array}
\]
\( \tilde{g} \) is not injective. Since there are no common images under \( \tilde{f} \) and \( \tilde{g} \), the pullback object is empty. Therefore, the pushout object is \( \{b_1, b_2, c\} \).

The lemma does not hold true in general. \( \mathsf{Setincl} \) is a counterexample.

**Example 2.5.17:**

In \( \mathsf{Setincl} \), each morphism is a monomorphism and an epimorphism. Therefore, each pair of morphisms with common codomain is jointly epimorphic. The pullback object of \( \tilde{g}: B \to D \) and \( \tilde{f}: C \to D \) is \( A := B \cap C \). The inserted pushout construction yields \( D' := B \cup C \). Of course, this is a subset of \( D \), but \( D \) may also include additional elements.

We now turn to construct pushout diagrams from given ones:

**Lemma 2.5.18** (Composing pushouts):

Consider the following commutative diagram:

\[
\begin{array}{c}
\begin{tikzpicture}
  \node (A) at (0,0) {A};
  \node (B) at (2,0) {B};
  \node (C) at (0,-2) {C};
  \node (D) at (2,-2) {D};
  \node (B') at (2,2) {B'};
  \node (D') at (2,-4) {D'};
  \node (H) at (4,-2) {H};
  \draw[->] (A) -- node[above] {$f_1$} (B);
  \draw[->] (B) -- node[above] {$f_2$} (B');
  \draw[->] (B') -- node[above] {$\tilde{f}_1$} (B);
  \draw[->] (B') -- node[above] {$\tilde{f}_2$} (D');
  \draw[->] (C) -- node[above] {$g$} (D);
  \draw[->] (D) -- node[above] {$f_1$} (D');
  \draw[->] (D') -- node[above] {$f_2$} (D);
  \draw[->] (A) -- node[above] {$g$} (C);
  \draw[->] (B') -- node[above] {$\tilde{g}_1$} (B);
  \draw[->] (D') -- node[above] {$\tilde{g}_2$} (D);
  \draw[->] (H) -- node[above] {$q$} (D');
  \draw[->] (H) -- node[above] {$p$} (B');
  \draw[->] (B) -- node[above] {$\tilde{g}_2$} (C);
\end{tikzpicture}
\end{array}
\]

(a) If both the left-hand subdiagram \( \tilde{g}_1 \cdot f_1 = \tilde{f}_1 \cdot g \) and the right-hand subdiagram \( \tilde{g}_2 \cdot f_2 = \tilde{f}_2 \cdot \tilde{g}_1 \) are pushout diagrams, then so is the outer diagram \( \tilde{g}_2 \cdot f_2 \cdot f_1 = \tilde{f}_2 \cdot \tilde{f}_1 \cdot g \).

(b) Conversely, if the left-hand subdiagram and the outer diagram are pushout diagrams, then the right-hand subdiagram is a pushout diagram, too.

(c) If the right-hand subdiagram and the outer diagram are pushout diagrams and if in addition, \( f_2 \) is a coretraction, then the left-hand subdiagram is a pushout diagram.

**Proof:**

First, we start with two morphisms \( p \) and \( q \) making the outer diagram commutative:

\[
q \cdot (f_2 \cdot f_1) = p \cdot g \quad (\text{Diagram (a)}).
\]

We have to show that there is a unique \( u \) with

\[
q = u \cdot \tilde{g}_2 \land p = u \cdot (\tilde{f}_2 \cdot \tilde{f}_1).
\]

Our assumption can be rewritten as \( (q \cdot f_2) \cdot f_1 = p \cdot g \), and the pushout property of the left-hand subdiagram yields the existence of a unique \( h \)
factorizing $p$ and $q \cdot f_2$: $p = h \cdot \bar{f}_1 \land q \cdot f_2 = h \cdot \bar{g}_1$. The second equality allows us to apply the pushout property of the right-hand subdiagram: There exists a unique $u$ such that $q = u \cdot \bar{g}_2 \land h = u \cdot \bar{f}_2$, and therefore, $p = h \cdot \bar{f}_1 = u \cdot \bar{f}_2 \cdot \bar{f}_1$. This $u$ does not depend on the special way we have used to construct it, because the pair $(\bar{f}_2, \bar{g}_2)$ is jointly epimorphic.

To show the second part of the lemma, we consider a similar diagram: In this case, we have to construct a unique $u$ starting with $p \cdot \bar{g}_1 = q \cdot f_2$. We can extend this equation by tracing back the morphisms to object $A$: $p \cdot \bar{f}_1 \cdot g = p \cdot \bar{g}_1 \cdot f_1 = q \cdot f_2 \cdot f_1$, and we then apply the pushout property of the “outer” diagram $\bar{g}_2 \cdot f_2 \cdot f_1 = \bar{f}_2 \cdot \bar{f}_1 \cdot g$. This leads to a morphism $u$ with $p \cdot \bar{f}_1 = u \cdot \bar{f}_2 \cdot \bar{f}_1 \land q = u \cdot \bar{g}_2$. The second equation is part of what we want to have, but the first is not yet the result. If $\bar{f}_1$ were epimorphic, we could get the result by Lemma 2.3.6, but this does not hold true in each case. Instead, we make use of the pushout property of the left-hand subdiagram: We can find an unambiguous $p'$ with $p \cdot \bar{f}_1 = p' \cdot \bar{f}_1$ and $u \cdot \bar{f}_2 \cdot \bar{g}_1 = p' \cdot \bar{g}_1$. Both $p$ and $u \cdot \bar{f}_2$ satisfy the properties of $p'$ because of $p \cdot \bar{g}_1 = q \cdot f_2 = u \cdot \bar{g}_2 \cdot f_2 = u \cdot \bar{f}_2 \cdot \bar{g}_1$. Therefore, we have $p' = p = u \cdot \bar{f}_2$. Since the pair $(\bar{f}_2 \cdot \bar{g}_1, \bar{g}_2)$ is jointly epimorphic, this does also hold for the pair $(\bar{f}_2, \bar{g}_2)$, and we get the uniqueness of $u$.

In the third case (Diagram (c)), we consider two morphisms $p : C \to H$ and $q : B \to H$ with $p \cdot g = q \cdot f_1$: As $f_2$ is a coretraction, we have a $f'_2$ with $f'_2 \cdot f_2 = \text{id}_B$ and we are allowed to rewrite $q \cdot f_1 = q \cdot (f'_2 \cdot f_2) \cdot f_1 = q \cdot f'_2 \cdot (f_2 \cdot f_1)$, i.e., we have a commutative diagram passing through the corners $B'$ and $C$ of the outer pushout diagram, giving
us a unique \( h \) with \( p = h \cdot (\bar{f}_2 \cdot \bar{f}_1) \) and \( q \cdot f'_2 = h \cdot \bar{g}_2 \). We have to find a \( u : D \to H \) with \( p = u \cdot \bar{f}_1 \) and \( q = u \cdot \bar{g}_1 \). The first condition is easily satisfied by defining \( u = h \cdot \bar{f}_2 \), and the second follows from this: \( u \cdot \bar{g}_1 = h \cdot \bar{f}_2 \cdot \bar{g}_1 = h \cdot \bar{g}_2 \cdot f_2 = q \cdot f'_2 \cdot f_2 = q \). Finally, we have to show that \( u \) is unique. We assume existence of another \( u' \) with \( p = u' \cdot \bar{f}_1 \) and \( q = u' \cdot \bar{g}_1 = (q \cdot f'_2) \cdot f_2 \). Since the right-hand subdiagram is a pushout diagram, there exists a unique \( h' \) with \( u' = h' \cdot \bar{f}_2 \) and \( q \cdot f'_2 = h' \cdot \bar{g}_2 \). This \( h' \) is (up to isomorphism) the same as the \( h \) we have constructed before. This follows from \( h \cdot \bar{f}_2 \cdot \bar{f}_1 = p = u' \cdot \bar{f}_1 = h' \cdot \bar{f}_2 \cdot \bar{f}_1 \) and \( h \cdot \bar{g}_2 = q \cdot f'_2 = h' \cdot \bar{g}_2 \) together with the fact that \((\bar{f}_2 \cdot \bar{f}_1, \bar{g})\) is a jointly epimorphic pair as part of the outer pushout diagram, and we also have \( u' = h' \cdot f_2 = h \cdot \bar{f}_2 = u \).

\( \square \)

**Corollary 2.5.19** (Composing pushouts in \( \text{Set} \)):

If in \( \text{Set} \), the right-hand subdiagram and the outer diagram are pushout diagrams and if in addition, \( f_2 \) is injective, then the left-hand subdiagram is a pushout diagram.

The only case of an injection that is not a coretraction implies that \( B \) is the empty set and therefore, \( A \) is empty, too. \( D' \) is the disjoint union of \( B' \) and \( C \) as well as of \( B' \) and \( D \) because of Lemma 2.5.4, thus \( C = D \), and the left-hand subdiagram is trivially a pushout diagram in this special case.

We now give an example showing that the additional assumption is necessary in the third part of the lemma:

**Example 2.5.20:**

We consider the following diagram with a noninjective \( f_2 \). The mappings are given by the numbering of the elements of the sets, e.g., \( f_2(3) = [3, 4] \).

\[
\begin{array}{ccc}
\{1, 2, 3, 4\} & \xrightarrow{f_1} & \{1, 2, 3, 4\} \\
\downarrow{g} & & \downarrow{\bar{g}_1} \\
\{1, 2, 3, 4\} & \xrightarrow{\bar{f}_1} & \{1, 4, 2, 3\} \\
& \downarrow{f_2} & \\
& \{1, 2, 3, 4\} & \xrightarrow{f_2} \{1, 2, 3, 4\}
\end{array}
\]

The outer diagram and the right-hand subdiagram are pushouts. (If you do not see this immediately, you should apply the canonical construction to the pair \((f_2 \cdot f_1, g)\) and to the pair \((f_2, \bar{g}_1)\).) The left-hand diagram, however, is not a pushout. Since \( f_1 \) is an isomorphism, \( \bar{f}_1 \) must also be an isomorphism if it were.

To conclude this chapter, we return to the category \( \text{Word} \) (Theorem 2.2.2), and we look at the pushout diagrams in this category.

**Lemma 2.5.21** (Pushout in \( \text{Word} \)):

The category \( \text{Word} \) has pushouts.

The proof is based on the observation that the pushout construction can be transferred from \( \text{Set} \) to \( \text{Word} \):

\[
\begin{array}{ccc}
A & \xrightarrow{f} & B \\
\downarrow{g} & & \downarrow{\bar{g}} \\
C & \xrightarrow{f} & D
\end{array} \quad \Rightarrow \quad \begin{array}{ccc}
A^* & \xrightarrow{f^*} & B^* \\
\downarrow{g^*} & & \downarrow{\bar{g}^*} \\
C^* & \xrightarrow{f^*} & D^*
\end{array}
\]
Commutativity of the right-hand diagram is an immediate consequence of commutativity of the left-hand diagram and the definition of the morphisms in \textit{Word}. Furthermore, if we have a commutative diagram $h^* \cdot f^* = k^* \cdot g^*$ in \textit{Word}, the corresponding diagram without stars is commutative in $\textit{Set}$, and we get a unique $u$ factorizing $k$ and $h$:

$$
\begin{array}{ccc}
A^* & \xrightarrow{f^*} & B^* \\
\downarrow{g^*} & & \downarrow{k^*} \\
C^* & \xrightarrow{\bar{f}^*} & D^* \\
\downarrow{k^*} & & \downarrow{\bar{g}^*} \\
E^* & \xrightarrow{\bar{h}^*} & \end{array}
$$

$u^*$ satisfies $h^* = u^* \cdot q^* \land k^* = u^* \cdot p^*$. Finally, if there were another $u'$ with this property, $u'$ would be an alternative to $u$ in $\textit{Set}$ contrary to the pushout property. $\square$

The $*$-operation constructing a new category from a given one, is an example of a functor between two categories. A functor that preserves all colimit constructions, such as the $*$-operation does, is said to be cocontinuous. We shall study some functors later on.

### 2.6 Exercises

**Exercise 2.6.1:**

In the appendix, we program categorical constructions, and in that context, an alternative definition of the notion of a category will be more convenient: A category is a quintuple $\mathcal{K} = (\text{Obj}_K, \text{Mor}_K, \text{dom}_K, \text{codom}_K, \cdot)$ where the following conditions hold:

\begin{enumerate}
  \item $\text{dom}_K, \text{codom}_K : \text{Mor}_K \rightarrow \text{Obj}_K$
  \item $\cdot : \text{Mor}_K \times \text{Mor}_K \rightarrow \text{Mor}_K$ is a partial function with the following properties:
    \begin{itemize}
      \item $g \cdot f$ exists if and only if $\text{dom}_K(f) = \text{dom}_K(g)$.
      \item $\text{dom}_K(g \cdot f) = \text{dom}_K(f) \land \text{codom}_K(g \cdot f) = \text{codom}_K(g)$.
      \item If in $(h \cdot g) \cdot f = h \cdot (g \cdot f)$, one side of the equation exists, the other also exists and both are equal.
      \item For each object $A \in \text{Obj}_K$, there exists a morphism $\text{id}_A \in \text{Mor}_K$ such that $(\forall A, B \in \text{Obj}_K)(\forall f : A \rightarrow B)(\text{id}_B \cdot f = f = f \cdot \text{id}_A)$.
    \end{itemize}
\end{enumerate}

Show that this version is equivalent to Definition 2.1.1.

**Exercise 2.6.2:**

We can represent multisets by usual sets if we number the elements, e.g., we represent \{a, a, b, b, b, c\} by \{(1, a), (2, a), (3, b), (5, b), (7, b), (11, c)\} where the numbers must be unique; besides, they can be chosen arbitrarily. A morphism is a function that preserves the second component, i.e., $f((i, l)) = (j, l)$. Show that this is a category and a subcategory of $\textit{Set}$.
Exercise 2.6.3:
Consider Example 2.2.10 and construct an L-hypergraph morphism from $H_3$ to $H_1$.

Exercise 2.6.4:
In Example 2.2.10, we have introduced the set of terms $\mathcal{T}(\Sigma, V)$. A substitution is a mapping $\sigma : \mathcal{T}(\Sigma, V) \to \mathcal{T}(\Sigma, V)$ satisfying:

(a) $\sigma(x) \in \mathcal{T}_s(\Sigma, V)$ for all $x \in V_s$.
(b) $\sigma(op(t_1, t_2, \ldots, t_n)) = op(\sigma(t_1), \sigma(t_2), \ldots, \sigma(t_n))$ for all operation symbols $op$.

Show that the class of sets of terms together with substitutions as morphisms and the usual composition $(\sigma' \cdot \sigma)(t) = \sigma'(\sigma(t))$ constitute a category.

Exercise 2.6.5:
Alternatively, we can define a category the objects of which are the terms (instead of sets of terms). A morphism from $t \to t'$ exists if there is a matching substitution $\sigma(t) = t'$. Prove that this is a category.

Exercise 2.6.6:
Consider Lemma 2.3.3. In the text, we have proved part (b) by a sequence of diagrams. Translate this proof into a sequence of formulae.

Exercise 2.6.7:
Prove that the epimorphisms in $\mathcal{Set}$ coincide with the surjective functions.

Exercise 2.6.8 (Dual category):
For any category $\mathcal{K} = (\text{Obj}_K, \text{Mor}_K, \text{dom}_K, \text{codom}_K, \ast)$ as defined in Exercise 2.6.1, the dual category is given by $\mathcal{K}^{\text{dual}} = (\text{Obj}_K, \text{Mor}_K, \text{codom}_K, \text{dom}_K, \ast)$ with $\ast$ being defined by $f \ast g = g \cdot f$. Show that this is indeed a category.

Exercise 2.6.9:
Prove: $f$ is an isomorphism if and only if it is both a retraction and a monomorphism.

Exercise 2.6.10:
Prove that the coequalizer of two parallel morphisms $f$ and $g$ is unique up to isomorphism.

Exercise 2.6.11 (Ternary coproduct):
The coproduct of $n$ objects $A_1, A_2, \ldots, A_n$ is an object $C$ together with $n$ morphisms $f_i : A_i \to C$ ($i = 1, 2, \ldots, n$) such that if an object $C'$ and $n$ morphisms $f'_i : A_i \to C'$ are given, there exists a unique $u : C \to C'$ with $f'_i = u \cdot f_i$ for all $i = 1, 2, \ldots, n$.

Consider the case $n = 3$. Construct the ternary coproduct, i.e., the coproduct of $A_1, A_2, A_3$ by first constructing the coproduct of $A_1$ and $A_2$ and then the coproduct of the result and $A_3$. Prove that you get the same result (up to isomorphism) when constructing first the coproduct of $A_2$ and $A_3$ and then combining $A_1$ with the result of the first step.

Exercise 2.6.12 (Uniqueness of initial object):
Prove that the initial object of a category is unique up to isomorphism.
Exercise 2.6.13 (Definition of limits):
Give the definitions of the product, the equalizer, and the pullback applying the duality principle. Describe the canonical construction of pullback diagrams!

Exercise 2.6.14:
Consider two objects $A$ and $B$ in the category of sets. Prove that the product of $A$ and $B$ in $\textbf{Set}$ is the Cartesian product $\{(a, b) \mid a \in A \land b \in B\}$ together with the projections $p_1((a, b)) := a$ and $p_2((a, b)) := b$.

Exercise 2.6.15:
Let $f : A \to B$ and $g : A \to B$ be two morphisms in $\textbf{Set}$. Show that
$$E := \{a \in A \mid f(a) = g(a)\}$$
together with the natural injection $e : a \mapsto a$ is the equalizer of $(f, g)$.

Exercise 2.6.16 (Pullback in $\textbf{Set}$):
Show that the pullback construction in $\textbf{Set}$ can be characterized as follows: Let $f : X \to B$ and $g : Y \to B$ two morphisms and let be
$$A := \{(x, y) \mid f(x) = g(y)\} \subseteq X \times Y.$$Then, $f \cdot p_1 = g \cdot p_2$ is the pullback diagram, where $p_1 : X \times Y \to X$ and $p_2 : X \times Y \to Y$ are the projections.

Exercise 2.6.17:
Consider the sets $X = \{1', 2', 3'\}$, $Y = \{1'', 2'', 3''\}$, and $B = \{a, b, c, d\}$ together with two morphisms given by:
$$f(1') = a \quad f(2') = b \quad f(3') = c$$
$$g(1'') = a \quad g(2'') = b \quad g(3'') = c$$
Construct the pullback diagram!

Exercise 2.6.18:
Prove Lemma 2.5.4.

Exercise 2.6.19:
Characterize the colimits of the category introduced in Exercise 2.6.2.

Exercise 2.6.20:
In $\textbf{Set}$, a pair $(f, g)$ of morphisms $f : A \to C$ and $g : B \to C$ is jointly epimorphic if and only if $C = f[A] \cup g[B]$.

Exercise 2.6.21:
A clever student may prove the third part of Lemma 2.5.18 by redrawing the Diagram (c) as given in the figure. Then, he or she can apply Lemma 2.5.14 and
get that the right-hand part of the composed diagram, i.e., the inner diagram of $(c')$, is a pushout diagram in each case. Find the mistake!
## Contents

2 Categorical Notions

2.1 Categories, Sets, and Graphs ............................................. 18
2.2 Hypergraphs ................................................................. 25
2.3 A Hierarchy of Morphisms ................................................. 31
2.4 Colimits ................................................................. 36
2.5 Properties of Pushout Diagrams ........................................... 43
2.6 Exercises ................................................................. 54