Alfio Martini

Graph-Grammars- An Introduction to the Double-Pushout Approach

In cooperation with

Hartmut Ehrig & Daltro Nunes

March, 1996

Graph Grammars – An Introduction to the Double-Pushout Approach

Alfio Martini Hartmut Ehrig Daltro Nunes

e-mail:{alfio,ehrig}@cs.tu-berlin.de,daltro@inf.ufrgs.br

Abstract

Graph Grammars provide an intuitive description for the manipulation of complex structures based on graphs, for instance, data base systems, operating systems, and complex applicative software in general. The algebraic theory of graph grammars is a tentative to describe graph grammars and the gluing construction for graphs as the basic concepts for the construction of derivations. This report consists of four sections. In the first one we present a series of definitions and elementary results concerning graphs, graph morphisms and congruences on graphs, aiming at both establishing terminology for further sections, as well as making it as self-contained as possible. The second section introduces carefully and with extreme detail the formal description of a graph derivation and its abstract (categorical) properties. The third is based heavily upon the second in order to introduce the notion of graph derivation within the double-pushout approach (short DPA), while the last one presents some nice theoretical results concerning Church-Rosser properties and parallelism of graph derivations.

Contents

1 Basic definitions **2**

1.1 Labeled graphs and labeled graph morphisms 2

1.2 Gluing constructions 5

2 Direct derivations **8**

2.1 Rules and matches 9

2.2 Explicit constructions, pushouts and pullbacks 10

3 The double-pushout approach **20**

3.1 Rules and matches 20

3.2 Derivations 22

4 Results **26**

4.1 Independence of derivations 26

4.2 The Church-Rosser theorems 28

4.3 The parallelism theorem 33

5 Bibliographic Notes **35**

A Categorical Background **36**

A.1 Categories and diagrams 36

A.2 Special kinds of morphisms 38

A.3 Universal constructions 38

A.4 General limit and colimit constructions 42

1 Basic definitions

The purpose of this section is to present a small set of basic definitions concerning graphs and graph homomorphisms in order to establish notation and terminology used throughout the text. Definitions and basic results concerning congruences on graphs should be considered the key point.

1.1 Labeled graphs and labeled graph morphisms

In the sequel, we present a couple of definitions related to (labeled) graphs and graph homomorphisms (also called graph morphisms). In order to be closer to practical applications we allow vertices and edges to be labeled over a labeling alphabet. Therefore, let $L = (L_E, L_V)$ be a pair of sets, called **labels** (also called **color alphabets**) for the sets of edges and vertices, respectively, which are mean to be fixed in the following:

Definition 1.1.1 *A* **(labeled) graph** $G = (G_E, G_V, s^G, t^G, a^G, n^G)$ *consists of sets* G_E *and* G_V*, called, respectively, sets of edges and vertices, mappings* $s^G : G_E \to G_V$*,* $t^G : G_E \to G_V$*, called, respectively,* **source** *and* **target** *maps as well as a pair of additional mappings* $a^G : G_E \to L_E$*,* $n^G : G_V \to L_V$*, called, respectively,* **edge** *and* **vertex** *labeling map. This information can be summarized in the following diagram:*

$$L_E \xleftarrow{a^G} G_E \underset{t^G}{\overset{s^G}{\rightrightarrows}} G_V \xrightarrow{n^G} L_V$$

Remark 1.1.2 We may sometimes use just G to denote the whole tuple as presented above. Besides, if we don't want to distinguish between edges and vertices we use the term **item** and $x \in G$ means $x \in G_V$ or $x \in G_E$. Furthermore, we use interchangeably the terms *map, mapping and function*, all of them meaning a *total function.*

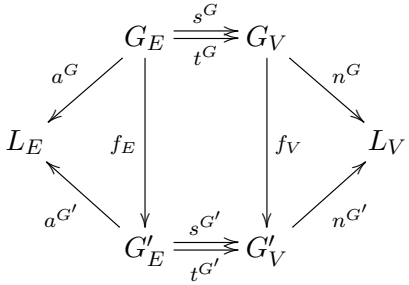
Example 1.1.3 Figure 1 shows a graphical representation of a graph $G = (G_E, G_V, s^G, t^G, a^G, n^G)$ with $G_E = \{a, b, c, d\}, G_V = \{1, 2, 3, 4, 5, 6\}, s^G = \{a \mapsto 1, b \mapsto 2, c \mapsto 5, d \mapsto 6\}, t^G = \{a \mapsto 3, b \mapsto 3, c \mapsto 4, d \mapsto 4\}, n^G = \{1 \mapsto u, 2 \mapsto v, 3 \mapsto w, 4 \mapsto w, 5 \mapsto x, 6 \mapsto v\}, a^G = \{a \mapsto t, b \mapsto t, c \mapsto t, d \mapsto t\}$. Edges are drawn in the usual way as arrows from the source to the target, and the label of each node or edge is written after its identity, separated by a colon.

Definition 1.1.4 *A graph G is called* **discrete** *if $G_E = \emptyset$.*

Definition 1.1.5 *A graph G' is called a* **subgraph** *of G, if $G'_E \subseteq G_E$, $G'_V \subseteq G_V$, and all the mappings $s^{G'}$, $t^{G'}$, $a^{G'}$ and $n^{G'}$ are restrictions of the corresponding mappings of G, i.e.,*

1. $\forall e \in G_E : s^{G'}(e) = s^G(e)$
2. $\forall e \in G_E : t^{G'}(e) = t^G(e)$
3. $\forall e \in G_E : a^{G'}(e) = a^G(e)$
4. $\forall v \in G_V : n^{G'}(v) = n^G(v)$

Definition 1.1.6 *Given two graphs G and G', a graph morphism $f : G \to G'$ is a pair of mappings $f = (f_E : G_E \to G'_E, f_V : G_V \to G'_V)$, such that the following diagram commutes (for source and target mappings separately):*



More explicitly, the following equations should hold:

1. $\forall e \in G_E : f_V(s^G(e)) = s^{G'}(f_E(e))$
2. $\forall e \in G_E : f_V(t^G(e)) = t^{G'}(f_E(e))$
3. $\forall e \in G_E : a^G(e) = a^{G'}(f_E(e))$
4. $\forall v \in G_V : n^G(v) = n^{G'}(f_V(v))$

Remark 1.1.7 The last two equations assure preservation of labels for edges and vertices, respectively. We also say that $f : G \to G'$ is a graph morphism if f is at the same time a function and compatible with source, target and labeling maps.

Definition 1.1.8 *A graph morphism $f = (f_E, f_V) : G \to G'$ is called* **injective**, *respectively* **surjective**, *if both f_E and f_G are injective, respectively surjective mappings.*

Proposition 1.1.9 *If $f : G \to G'$ is injective and surjective then f is called an* **isomorphism** *and thus, there exists an* **inverse** *isomorphism $f^{-1} = (f_E^{-1}, f_V^{-1}) : G' \to G$.*

Figure 1: Graphical representation of a graph

Proof: Note that bijective maps in **Set** are isomorphic, i.e., $f^{-1} \circ f = (f_E^{-1} \circ f_E, f_V^{-1} \circ f_V) = (id_{G_E}, id_{G_V}) = id_G$ and the same for $f \circ f^{-1} = id_{G'}$. The compatibility properties of f^{-1} are easy to see. For instance,

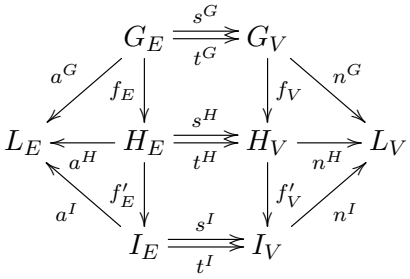
$$\begin{array}{rcll} f_V^{-1}(s^{G'}(e')) & = & f_V^{-1}(s^{G'}(f_E(f_E^{-1}(e')))) & (f \circ f^{-1} = id_{G'}) \\ & = & f_V^{-1}(f_V(s^G(f_E^{-1}(e')))) & (f \text{ is a graph morphism}) \\ & = & s^G(f_E^{-1}(e')) & (f^{-1} \circ f = id_G) \end{array}$$

The other compatibility conditions are proved analogously. □

Remark 1.1.10 The notation $G \cong H$ denotes that there is an isomorphism from G to H.

Definition and proposition 1.1.11 *The composition $f' \circ f : G \to I$ of two graph morphisms $f = (f_E, f_V) : G \to H$ and $f' = (f'_E, f'_V) : H \to I$ is defined by $f' \circ f = (f'_E \circ f_E, f'_V \circ f_V)$.*

Proof: We must show that the following diagram commute:



We show that $f' \circ f$ is compatible with the source mapping. The other cases are similar or even easier.

$$\begin{array}{rcll} (f'_V \circ f_V) \circ s^G & = & f'_V \circ (f_V \circ s^G) & \text{(associativity)} \\ & = & f'_V \circ (s^H \circ f_E) & (f \text{ is a graph morphism}) \\ & = & (f'_V \circ s^H) \circ f_E & \text{(associativity)} \\ & = & (s^I \circ f'_E) \circ f_E & (f' \text{ is a graph morphism}) \\ & = & s^I \circ (f'_E \circ f_E) & \text{(associativity)} \end{array}$$

Proposition 1.1.12 *The class of labeled graphs, together with the morphisms defined in 1.1.6 and the composition of morphisms in 1.1.6, constitutes a category, the category* $\mathbf{Graph_L}$*, of (labeled) graphs.*

Proof: Since graph morphisms can be composed (see above), the associativity law holds trivially, since the definition of graph morphisms is based on maps in the category **Set**. Moreover for every graph G we have the identity graph morphism $id_G = (id_{G_E}, id_{G_V}) : G \to G$ (which is clearly compatible with sources, targets and labels in the sense of 1.1.6) such that

for every graph morphism $f : G \to G'$ we have that $id_{G'} \circ f = f = f \circ id_G$ (once again because the morphisms are based on maps in **Set**).

□

1.2 Gluing constructions

Definition 1.2.1 (Gluing relation) *A* **gluing relation** $\equiv$ *in a graph G is a pair of equivalence relations $\equiv= (\equiv_E \subseteq G_E \times G_E, \equiv_V \subseteq G_V \times G_V)$ such that the following compatibility conditions are satisfied:*

1. *They identify only objects of the same label:*

$$\begin{array}{lcl} v_1 \equiv_V v_2 & \Rightarrow & n^G(v_1) = n^G(v_2) \\ e_1 \equiv_E e_2 & \Rightarrow & a^G(e_1) = a^G(e_2) \end{array}$$

2. *Two edges are only identified if their sources and targets are also identified:*

$$e_1 \equiv_E e_2 \Rightarrow s^G(e_1) \equiv s^G(e_2) \wedge t^G(e_1) \equiv t^G(e_2)$$

Remark 1.2.2 In simpler terms, a gluing relation $\equiv$ on G can be viewed as a congruence relation on G if we see graphs as special kinds of "algebras".

Definition and proposition 1.2.3 (Gluing of a graph) *Let G be a graph and $\equiv$ a gluing relation on G. Then the gluing of G along $\equiv$, denoted $G_{/\equiv}$, is defined in the following way:*

$$\begin{array}{lll} G_{/\equiv_V} = \{[v]_\equiv | v \in G_V\} & \textbf{\textit{where}} & [v]_\equiv = \{v' \in G_V | v \equiv_V v'\}, \\ G_{/\equiv_E} = \{[e]_\equiv | e \in G_E\} & \textbf{\textit{where}} & [e]_\equiv = \{e' \in G_V | e \equiv_E e'\}, \end{array}$$

$$\begin{array}{llcll} s^{G_{/\equiv}} : G_{/\equiv_E} \to G_{/\equiv_V} & , & s^{G_{/\equiv}}([e]_\equiv) & =_{def} & [s^G(e)]_\equiv \\ t^{G_{/\equiv}} : G_{/\equiv_E} \to G_{/\equiv_V} & , & t^{G_{/\equiv}}([e]_\equiv) & =_{def} & [t^G(e)]_\equiv \\ a^{G_{/\equiv}} : G_{/\equiv_E} \to L_E & , & a^{G_{/\equiv}}([e]_\equiv) & =_{def} & a^G(e) \\ n^{G_{/\equiv}} : G_{/\equiv_V} \to L_V & , & n^{G_{/\equiv}}([v]_\equiv) & =_{def} & n^G(v) \end{array}$$

Proof: We have to verify well-definedness of $s^{G_{/\equiv}}, t^{G_{/\equiv}}, a^{G_{/\equiv}}$, and $n^{G_{/\equiv}}$. We show here explicitly for $s^{G_{/\equiv}}$ and $n^{G_{/\equiv}}$ (the proofs for $t^{G_{/\equiv}}$ and $a^{G_{/\equiv}}$ are analogous).

Now, in case $e_1 \equiv_E e_2$, it follows that $s^G(e_1) \equiv s^G(e_2)$ by definition of a gluing relation and hence that $[s^G(e_1)]_\equiv = [s^G(e_2)]_\equiv$, since $\equiv$ is an equivalence relation. Now we have that

$$\begin{array}{lcll} s^{G_{/\equiv}}([e_1]_\equiv) & = & [s^G(e_1)]_\equiv & \text{(by definition of } s^{G_{/\equiv}}) \\ & = & [s^G(e_2)]_\equiv & \text{(assumption)} \\ & = & s^{G_{/\equiv}}([e_2]_\equiv) & \text{(by definition of } s^{G_{/\equiv}}) \end{array}$$

This shows that $s^{G/\equiv}$ is well-defined. Moreover, since $\equiv$ is a gluing relation, we have that $v_1 \equiv_V v_2 \Rightarrow n^G(v_1) = n^G(v_2)$ and hence that

$$\begin{array}{rcll} n^{G/\equiv}([v_1]_\equiv) & = & n^G(v_1) & \text{(by definition of } n^{G/\equiv}) \\ & = & n^G(v_2) & \text{(by assumption)} \\ & = & n^{G/\equiv}([v_2]_\equiv) & \text{(by definition of } n^{G/\equiv}) \end{array}$$

□

Example 1.2.4 Consider the graph G from Figure 1 and the least gluing relation $\equiv$ on G that contains $R = (\{(3,4),(2,6)\},\{(b,d)\})$ (i.e., the least equivalence relation[1] that contains R and it is compatible in the sense of 1.2.1). Now the gluing of G along $\equiv$ is exactly H (see Figure 2).

Proposition 1.2.5 *Let G and H be graphs and $h = (h_V, h_E) : G \to H$ a graph morphism. Then the kernel[2] of h, $kern(h) = (kern(h_V), kern(h_E))$ is a gluing relation $\equiv^h$ on G.*

Proof: We have to show that $\equiv^h$ is a gluing relation. Since $\equiv^h$ is an equivalence relation on G, it remains to show that the requirements of 1.2.1 are satisfied. Now, note that $v_1 \equiv^h_V v_2 \Rightarrow h_V(v_1) = h_V(v_2)$ and therefore

$$\begin{array}{rcll} n^G(v_1) & = & n^H(h_V(v_1)) & (h \text{ is a graph morphism}) \\ & = & n^H(h_V(v_2)) & \text{(by assumption)} \\ & = & n^G(v_2) & (h \text{ is a graph morphism}) \end{array}$$

The proof for t^G can be done analogously. Now, $e_1 \equiv^h e_2 \Rightarrow h_E(e_1) = h_E(e_2)$ and hence

$$\begin{array}{rcll} h_V(s^G(e_1)) & = & s^H(h_E(e_1)) & (h \text{ is a homomorphism}) \\ & = & s^H(h_E(e_2)) & \text{(by assumption)} \\ & = & h_V(s^G(e_2)) & (h \text{ is a homomorphism}) \end{array}$$

Now, $h_V(s^G(e_1)) = h_V(s^G(e_2))$ implies $s^G(e_1) \equiv^h_V s^G(e_2)$ and we are done. The proofs for t^G, n^G can be carried out in a similar way and we have shown that the kernel of a natural graph morphism h, $kern(h) = \equiv^h$, is a gluing relation. □

Remark 1.2.6 Given a function $f : A \to B$, a subset A' of A, then $f|_{A'}$ means the restriction of f to A'.

[1] The least equivalence relation that contains a given relation S is the reflexive, symmetric and transitive closure of S.

[2] For any function $f : A \to B$, the kernel of f, $kern(f)$ is defined by the set $\{(x,y)|f(x) = f(y)\}$, which is clearly an equivalence relation on A

Figure 2: H as a gluing of G from Figure 1

Definition 1.2.7 (Image of a graph) *The* **image** *of a graph G under a graph morphism $f : G \to H$ is defined as $f(G) = (f_V(G_V), f_E(G_E), s^{f(G)} = s^H|_{f_E(G_E)}, t^{f(G)} = t^H|_{f_E(G_E)}, n^{f(G)} = n^H|_{f_V(G_V)}, a^{f(G)} = a^H|_{f_E(G_E)})$.*

Proposition 1.2.8 *Let $f : G \to H$ be a graph morphism and $f(G)$ the image of G under f. Then $f(G)$ is a subgraph of H, i.e., $f(G) \subseteq H$.*

Proof: Note that, by construction, we have that $f_E(G_E) \subseteq H_E \wedge f_V(G_V) \subseteq H_V$. It remains to show that $s^{f(G)}$, $t^{f(G)}$, $a^{f(G)}$, and $n^{f(G)}$ are well-defined mappings. We show here explicitly for $s^{f(G)}$.

First, note that $s^{f(G)}$ is well-defined since s^H is well-defined. To see it is total, note that $e \in f_E(G_E) \Rightarrow \exists e' \in G_E$ such that $f_E(e') = e$. Now we have

$$\begin{array}{rcll} s^{f(G)}(e) & = & s^{f(G)}(f_E(e')) & \text{(assumption)} \\ & = & s^H(f_E(e')) & \text{(by definition of } s^{f(G)}) \\ & = & f_V(s^G(e')) & (f \text{ is a graph morphism}) \end{array}$$

Now, $s^{f(G)}(e) = f_V(s^G(e')) \in f_V(G_V)$ implies that $s^{f(G)}$ is total. □

Definition 1.2.9 *Given two graphs G and H, a* **surjective graph morphism** *$h : G \to H$ is a pair $(h_V : G_V \to H_V, h_E : G_E \to H_E)$ of surjective mappings h_V and h_E which preserves sources, targets and labels in the sense of 1.1.6.*

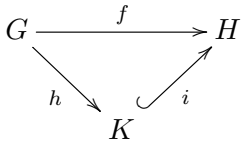
Proposition 1.2.10 *Let $f = (f_V, f_E) : G \to H$ be a graph morphism and $f(G)$ the image of G under f. Then there exists a surjective graph morphism $f' = (f'_V, f'_E) : G \to f(G)$ with $f'_V(v) = f_V(v)$ and $f'_E(e) = f_E(e)$.*

Proof: By 1.2.8, $f(G)$ is a graph and moreover, f' is trivially surjective. The conditions for f' to be a graph morphism are easily seen to be satisfied. For instance, we have that

$$\begin{array}{rcll} f'_V(s^G(e)) & = & f_V(s^G(e)) & \text{(definition of } f') \\ & = & s^H(f_E(e)) & (f \text{ is a graph morphism}) \\ & = & s^{f(G)}(f_E(e)) & \text{(definition of } s^{f(G)}) \end{array}$$

□

Proposition 1.2.11 (Factorization of graph morphisms). *For every graph morphism $f : G \to H$ there is a surjective graph morphism $h : G \to K$ and an injective graph morphism $i : K \to H$, so that the next diagram commutes, i.e., $f = i \circ h$.*



Proof: Choose $K = f(G)$. Now we can define the injection $i : f(G) \to H$ with $i = (i_V, i_E)$ as $i_V(v) = v$ and $i_E(e) = e$. This trivially satisfies the conditions for being a graph morphism. According to 1.2.10 there is a natural graph morphism $h : G \to f(G)$ with $h = (h_V, h_E)$ and $h_V(v) = f_V(v)$ and $h_E(e) = f_E(e)$. Therefore, it holds that $f = i \circ h$, since $(i \circ h)_V = i_V \circ h_V(v) = i_V(h_V)(v) = i_V(f_V)(v) = f_V(v)$ and the same for $(i \circ h)_E$.

Remark 1.2.12 In the above proof, $K = G/kern(f)$ also works, and thus $h : G \to K$ is the natural mapping $x \mapsto [x]$.

2 Direct derivations

In this section we begin to formalize the notion of a graph derivation by means of rules. By "begin", we mean that, at this moment, we are going to deal with the simplest case, namely, with rules that do not delete neither vertices nor edges, which means that in a derivation process a graph becomes always bigger. The formalization begins with the presentation of the concepts of rules and matches. After we construct in detail the (concrete) derived graph using elementary set-theoretic tools. Afterwards, we formalize this construction only by means of graph morphisms. The construction shows itself very long and tedious mainly due to the fact that a lot of steps must be carried out in order to verify that all the maps involved in this formalization are well-defined and indeed graph morphisms. However, the main fact of this section, theorem 2.2.6, shows that this construction can be characterized abstractly by means of a categorical colimit construction called pushout. That we are lucky to have at our hands such abstract characterization will only become apparent in the last section when we present main theoretical results concerning parallelism and concurrency in the double-pushout approach. This section ends with definitions concerning direct derivation, derivation sequence and generated graph grammar.

2.1 Rules and matches

Definition 2.1.1 (Non-deleting rule) *A* **graph rewriting rule** *that doesn't delete neither edges nor vertices is a (total) graph morphism $p : L \to R$. We call L the left side and R the right side.*

Example 2.1.2 Figure 3 shows a non-deleting rule which describes the gluing of two vertices, where the dashed lines indicate the desired morphism. Besides, from now on, whenever we have a graph (or two graphs related my a morphism) where all the vertices (resp. all the edges) are labeled over the same element of L_V (resp. of L_E), we omit label information in its graphical representation. We also omit the identities of edges and vertices in case we don't need to reference them explicitly.

A rule p can be applied to a graph G not only if L is a subgraph of G, but also allowing that items (vertices and edges) in L may be mapped onto the same item in G. The corresponding match, the **occurrence map**, is then said to be non-injective. More precisely, a rule L is applicable to a graph G if G contains a homomorphic image of L. This is represented in the following

Figure 3: Rule q for gluing of two vertices

Definition 2.1.3 (Match) *A* **match** *m for a rule $q : L \to R$ in a graph G is a (total) graph morphism $m : L \to G$.*

In order to describe formally the result of an application of a rule $p : L \to R$ in a graph G according to a match $m : L \to G$, we must first to inspect the effect of translating a gluing relation along a graph morphism. Two difficulties arise: when the graph morphism is not surjective, then the items (nodes/vertices) not in the homomorphic image will not be equivalent to themselves (the reflexivity axiom does not hold); if the graph morphism is not injective, then the transitivity relation may be destroyed, since new identifications are made. In other words, the translation of a gluing relation along a graph morphism may not yeld a gluing relation. However, the situation is not so bad, since the "missing" reflexive and transitive pairs can be added in order to "fix the bug". This is the work of the next

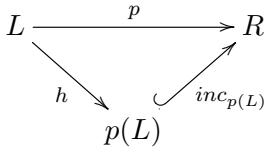
Proposition 2.1.4 *Let $\equiv^G = (\equiv^G_V, \equiv^G_E)$ be a gluing relation on a graph G and $f : G \to H$ a graph morphism. Then $f(\equiv^G) = (f(\equiv^G)_V, f(\equiv^G)_E)$ defined by $f(\equiv^G)_V = (f_V(v_1), f_V(v_2))|(v_1, v_2) \in \equiv^G_V)$ and $f(\equiv^G)_E = (f_E(e_1), f_E(e_2))|(e_1, e_2) \in \equiv^G_E)$ induces a gluing relation $\equiv^H$ on H, i.e., $\equiv^H = (\equiv^H_V, \equiv^H_E)$ with $\equiv^H_V = trans(refl(f(\equiv^G)_V))$ and $\equiv^H_E = trans(refl(f(\equiv^G)_E)$*

Proof: $\equiv^H$ is by construction reflexive and transitive. We show here that $\equiv^H_V$ is (already) symmetric (the proof for $\equiv^H_E$ is analogous). It follows from an elementary result, that the reflexive and transitive closure of a symmetric relation (resp. congruent relation) is again symmetric (resp. congruent). Thus, it remains only to show that $f(\equiv^G_V)_V$ is symmetric. Now, in case $v_1\, f(\equiv^G)_V\, v_2$, there exists $v'_1, v'_2 \in G_V$ with $f_V(v'_1) = v_1$ and $f_V(v'_2) = v_2$, and hence, by definition of $f(\equiv^G)_V$, that $v'_1 \equiv^G_V v'_2$. Now, since $\equiv^G$ is symmetric, it holds that $v'_2 \equiv^G_V v'_1$ and therefore that $f_V(v'_2)\, f(\equiv^G)_V\, f_V(v'_1)$. This shows that $\equiv^H$ is an equivalence relation and still remains to show that it satisfies the requirements of 1.2.1. Since the reflexive, transitive closure of a congruent relation is again a congruent relation, it is only necessary to show the compatibility of $f(\equiv^G)$ w.r.t s^H, t^H, a^H and n^H. We show here the compatibility for s^H. The other ones can be conducted similarly or even more easily. Thus, if $e_1\, f(\equiv^G)_E\, e_2$ then there must exist $e'_1, e'_2 \in G_E$ with $f_E(e'_1) = e_1$ and $f_E(e'_2) = e_2$ and hence that $e'_1 \equiv^G_E e'_2$. Now, since $\equiv^G$ is a gluing relation, it follows that $s^G(e'_1) \equiv^G_V s^G(e'_2)$. By definition of $f(\equiv^G)_V$ we have that $f_V(s^G(e'_1))\, f(\equiv^G)_V\, f_V(s^G(e'_2))$. Since f is a graph morphism, it holds that $s^H(f_E(e'_1))\, f(\equiv^G)\, s^H(f_E(e'_2))$ and hence that $s^H(e_1)\, f(\equiv^G)_V\, s^H(e_2)$, by substitution. □

2.2 Explicit constructions, pushouts and pullbacks

Definition 2.2.1 (Construction of the derived graph) *When $p : L \to R$ is a non-deleting rule and $m : L \to G$ is a match for p in G, then H is derivable from G in the following way:*

First we factorize p according to 1.2.11 such that the following diagram commutes:



Now, according to 1.2.5, h induces on L a gluing relation $\equiv^h$, whose image under m, according to 2.1.4, induces again a gluing relation $\equiv$ on G. Let $G_{/\equiv}$ be the gluing of G along $\equiv$ according to 1.2.3. Then H can be constructed as follows:

$H_V = (G_{/\equiv_V} \times \{1\}) \cup ((R_V - p_V(L_V)) \times \{2\})$
$H_E = (G_{/\equiv_E} \times \{1\}) \cup ((R_E - p_E(L_E)) \times \{2\})$

$$s^H ::= s^H(x) = \begin{cases} (s^{G_{/\equiv}}(e), 1) & , \textit{ if } x = (e,1) \\ ([m_V(v)]_\equiv, 1) & , \textit{ if } x = (e,2) \textit{ and } s^R(e) = p_V(v) \\ (s^R(e), 2) & , \textit{ if } x = (e,2) \textit{ and } s^R(e) \in R_V - p_V(L_V) \end{cases}$$

$$t^H ::= t^H(x) = \begin{cases} (t^{G_{/\equiv}}(e), 1) & , \textit{ if } x = (e,1) \\ ([m_V(v)]_\equiv, 1) & , \textit{ if } x = (e,2) \textit{ and } t^R(e) = p_V(v) \\ (t^R(e), 2) & , \textit{ if } x = (e,2) \textit{ and } t^R(e) \in R_V - p_V(L_V) \end{cases}$$

$$n^H ::= n^H(x) = \begin{cases} n^{G_{/\equiv}}(v) & , \textit{ if } x = (v,1) \\ n^R(v) & , \textit{ if } x = (v,2) \end{cases}$$

$$a^H ::= a^H(x) = \begin{cases} a^{G_{/\equiv}}(e) & , \textit{ if } x = (e,1) \\ a^R(e) & , \textit{ if } x = (e,2) \end{cases}$$

Remark 2.2.2 The main idea of the construction is that we "steal" some sources and targets from some edges of R. These edges are exactly those edges whose sources and/or targets are already in H, which in principle, come from $G_{/\equiv}$. Therefore, these edges of R must be associated with the corresponding vertices already existent in $G_{/\equiv}$.

Example 2.2.3 Consider the rule q and the match m in Figure 4. The graphs G, L, R are given the following formalization:

$L = (\{1,2\}, \emptyset, s^L = \emptyset, t^L = \emptyset)$
$R = (\{1 = 2\}, \emptyset, s^R = \emptyset, t^R = \emptyset)$
$G = (\{4,5\}, \{b\}, s^G = \{b \mapsto 4\}, t^G = \{b \mapsto 5\})$

where the rule q as well as the match m can be formalized as follows:

$q = (q_V, q_E)$ with $q_V = \{1 \mapsto 1 = 2, 2 \mapsto 1 = 2\}$ and $q_E = \emptyset$
$m = (m_V, m_E)$ with $m_V = \{1 \mapsto 4, 2 \mapsto 5\}$ and $m_E = \emptyset$

Now we can construct the derived graph H. Since q is already a surjective graph morphism, it is not necessary to factorize it. Thus, the induced equivalence relation on L, $\equiv^q = (\equiv^q_V, \equiv^q_E)$, is represented by $\equiv^q_V = \{(1,1),(2,2),(1,2),(2,1)\}$ and $\equiv^q_E = \emptyset$. Now, we can obtain $m(\equiv^q) = (m(\equiv^q_V), m(\equiv^q_V))$ with $m(\equiv^q_V) = \{(4,4),(5,5),(4,5),(5,4)\}$ and $m(\equiv^q_E) = \emptyset$.

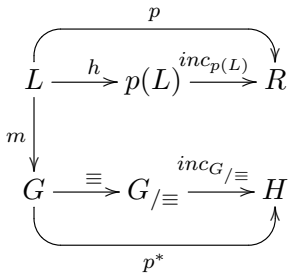
Figure 4: Derivation with rule q

Note that $m(\equiv^q)$ is not a gluing relation on G, but according to 2.1.4, $trans(refl(m(\equiv^q))) = \equiv$ is. Therefore, we obtain $\equiv_V = m(\equiv^q_V)$ and $\equiv_E = \{(b,b)\}$. Now we can glue G along $\equiv$ according to 1.2.3:

$$G_{/\equiv} = \{\{[4]_\equiv\}, \{[b]_\equiv\}, s^{G_{/\equiv}} = \{[b]_\equiv \mapsto [4]_\equiv\}, t^{G_{/\equiv}} = \{[b]_\equiv \mapsto [5]_\equiv\}\}$$

The rest of the construction simply introduces $R - q(L)$. But since q is a surjective graph morphism, $R - q(L) = \emptyset$ and the remaining construction is simply a renaming of $G_{/\equiv}$.

Now, we have to find a way to relate the constructed graph H with the other graphs presented in the construction. Since H was derived from G we may try to begin with this one. Let us remember how this derivation succeeded. First, G was glued along $\equiv$ to form $G_{/\equiv}$. This relation can be easily described with a natural graph morphism $\equiv\colon G \to G_{/\equiv}$. After, additional vertices and edges were introduced into $G_{/\equiv}$. At the same time, the items from $G_{/\equiv}$ were equiped with a unique identification (index 1) and included in H. Thus, $G_{/\equiv}$ is included (up to renaming) in H, and this relation we can represent by an inclusion $j = inc_{G_{/\equiv}} : G_{/\equiv} \to H$. Summarizing everything we have the situation represented by the following diagram:



This diagram suggest that it is reasonable to look for a relation between R and H. Note that, from the construction of H, we have included (up to renaming) part of R in H, since its remaining part is somehow already present in $G_{/\equiv}$. It remains only to express this relation as a graph morphism from R to G.

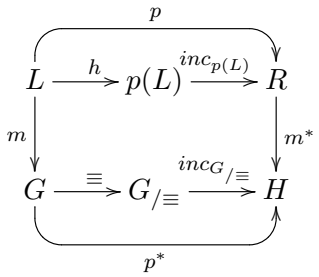
Proposition 2.2.4 *If $p : L \to R$ is a non-deleting rule, $m : L \to G$ is a match for p in G and H is the derived graph, then there exist graph morphisms $p^* : G \to H$ and $m^* : R \to H$, so that we have $m^* \circ p = p^* \circ m$.*

Proof: We define $p^* = j \circ \equiv$, i.e., $p^*_V(v) = j_V(\equiv_V (v)) = j_V([v]_\equiv) = ([v]_\equiv, 1)$ and similarly $p^*_E(e) = ([e]_\equiv, 1)$. Now let m^* be defined by

$$m^*_V ::= m^*_V(x) = \begin{cases} p^*_V(m_V(v)) & \text{, if } x = p_V(v) \\ (x,2) & \text{, otherwise} \end{cases}$$

$$m^*_E ::= m^*_E(x) = \begin{cases} p^*_E(m_E(e)) & \text{, if } x = p_E(e) \\ (x,2) & \text{, otherwise} \end{cases}$$

Now, p^* is itself a graph morphism, since is defined as a composition of two graph morphisms. By definition of m^* we have for all $v \in L_V$ that $(m^*_V \circ p_V)(v) = m^*_V(p_V(v)) = p^*_V(m_V(v)) = (p^*_V \circ m_V)(v)$ (the same holds for $(m^*_E \circ p_E)(e) = (p^*_E \circ m_E)(e)$), so that the following diagram commutes:



We still have to show that m^* is a well-defined graph morphism, that is to say, that m^* is a total well-defined function, and besides that this function is indeed a graph morphism. First note that m^* is trivially total by the case distinction in its definition. We show here well-definedness for m^*_V since for m^*_E the proof runs analogous. Now note that if $x = (x, 2)$, then m^*_V is trivially well-defined, since in this case it is just an injection. Now, this is not so clear if $x = p_V(v)$, since p^*_V is not necessarily injective. What we have to show is that $p_V(v_1) = p_V(v_2) \Rightarrow m^*_V(p_V(v_1)) = m^*_V(p_V(v_2))$. Note that $p_V(v_1) = p_V(v_2) \Rightarrow (v_1, v_2) \in kern(p)$. This means that $v_1 \equiv^L_V v_2$ (see 1.2.5). Now, by 2.1.4, $m_V(v_1) \equiv^G_V m_V(v_2)$ which implies that $\equiv_V (m_V(v_1)) = \equiv_V (m_V(v_2))$ (where here $\equiv: G \to G_{/\equiv}$). Now we have

$$\begin{array}{rcll} m^*_V(p_V(v_1)) & = & p^*_V(m_V(v_1)) & \text{(definition of } m^*_V) \\ & = & j_V(\equiv_V (m_V(v_1))) & \text{(definition of } p^*_V) \\ & = & j_V(\equiv_V (m_V(v_2))) & \text{(assumption)} \\ & = & p^*_V(m_V(v_2)) & \text{(defintion of } p^*_V) \\ & = & m^*_V(p_V(v_2)) & \text{(definition of } m^*_V) \end{array}$$

It remains to show that m^*_V is compatible w.r.t. source, target and labeling maps. Compatibility with the source map means that $m^*_V(s^R(e)) = s^H(m^*_E(e))$. We have to distinguish three cases:

1. $e = p_E(e')$.

$$\begin{array}{rcll} m^*_V(s^R(e)) & = & m^*_V(s^R(p_E(e'))) & \text{(substitution)} \\ & = & m^*_V(p_V(s^L(e'))) & (p \text{ is a graph morphism}) \\ & = & p^*_V(m_V(s^L(e'))) & \text{(definition of } m^*_V) \\ & = & p^*_V(s^G(m_E(e'))) & (m \text{ is a graph morphism}) \\ & = & ([s^G(m_E(e'))]_\equiv, 1) & \text{(definition of } p^*_V) \\ & = & (s^{G_{/\equiv}}([m_E(e')]_\equiv), 1) & \text{(definition of } G_{/\equiv}) \\ & = & s^H(([m_E(e')]_\equiv, 1)) & \text{(definition of } H) \\ & = & s^H(p^*_E(m_E(e'))) & \text{(definition of } p^*) \\ & = & s^H(m^*_E(p_E(e'))) & \text{(definition of } m^*) \\ & = & s^H(m^*_E(e)) & \text{(substitution)} \end{array}$$

2. $e \in R_E - p_E(L_E)$ but $s^R(e) = p_V(v)$.

$$\begin{array}{rcll} m_V^*(s^R(e)) & = & m_V^*(p_V(v)) & \text{(substitution)} \\ & = & p_V^*(m_V(v)) & \text{(definition of } m^*\text{)} \\ & = & ([m_V(v)]_{\equiv}, 1) & \text{(definition of } p^*\text{)} \\ & = & s^H((e,2)) & \text{(definition of } H\text{)} \\ & = & s^H(m_E^*(e)) & \text{(definition of } m_E^*\text{)} \end{array}$$

3. $e \in R_E - p_E(L_E)$ and $s^R(e) \in R_V - p_V(L_V)$.

$$\begin{array}{rcll} m_V^*(s^R(e)) & = & (s^R(e), 2) & \text{(definition of } m_V^*\text{)} \\ & = & s^H((e,2)) & \text{(definition of } H\text{)} \\ & = & s^H(m_E^*(e)) & \text{(definition of } m_E^*\text{)} \end{array}$$

The proof for the target map is analogous. It remains to show label compatibility. We show here explicitly for the edge labeling map, which means that we have to show that $a^R(e) = a^H(m_E^*(e))$. We distinguish two cases:

1. $e = p_E(e')$

$$\begin{array}{rcll} a^R(e) & = & a^R(p_E(e')) & \text{(substitution)} \\ & = & a^L(e') & (p \text{ is a graph morphism}) \\ & = & a^G(m_E(e')) & (m \text{ is a graph morphism}) \\ & = & a^H(p_E^*(m_E(e'))) & (p^* \text{ is a graph morphism}) \\ & = & a^H(m_E^*(p_E(e'))) & \text{(definition of } m_E^*\text{)} \\ & = & a^H(m_E^*(e)) & \text{(substitution)} \end{array}$$

2. $e \in R_E - p_E(L_E)$.

$$\begin{array}{rcll} a^R(e) & = & a^H((e,2)) & \text{(definition of } H\text{)} \\ & = & a^H(m_E^*(e)) & \text{(definition of } m_E^*\text{)} \end{array}$$

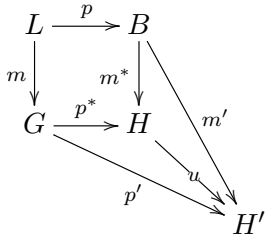
□

Remark 2.2.5

The commutativity $p^* \circ m = m^* \circ p$ means that items (vertices and edges) from G and R coming from L are identified in H.

Theorem 2.2.6 *if $p : L \to R$ is a non-deleting rule, $m : L \to G$ is match for p in G, and H is derived from G with $p^* : G \to H$ and $m^* : R \to H$, then the following holds:*

1. *p^* and m^* are jointly surjective, i.e., $\forall v \in H_V\ \exists v' \in G_V$ or $v' \in R_V$ such that $p^*_V(v') = v$, respectively, $m^*_V(v') = v$ (similarly for H_E).*
2. *For all graphs H' and graph morphisms $p' : G \to H'$, $m' : R \to H'$ with $p' \circ m = m' \circ p$, there exists a unique graph morphism $u : H \to H'$ such that $u \circ m^* = m'$ and $u \circ p^* = p'$ (see the following diagram).*



Proof of 1: We show here explicitly for H_V. Since H_V is constructed as a disjunction of two subsets, we distinguish here two cases:

Case $v = ([x]_{\equiv}, 1)$ with $x \in G_V$.

Choose $v' = x$. Then it holds that $p^*_V(v') = p^*_V(x) = j_V(\equiv_V (x)) = ([x]_{\equiv}, 1)$.

Case $v = (x, 2)$ with $x \in R_V - p_V(L_V)$.

Choose $v' = x$. Then it holds that $m^*_V(v') = m^*_V(x) = (x, 2)$.

Proof of 2:. We define $u = (u_V, u_E)$ as follows:

$$u_V ::= u_V(x) = \begin{cases} p'_V(v) & \text{, if } x = ([v]_{\equiv}, 1) \\ m'_V(v) & \text{, if } x = (v, 2) \end{cases}$$

$$u_E ::= u_E(x) = \begin{cases} p'_E(e) & \text{, if } x = ([e]_{\equiv}, 1) \\ m'_E(v) & \text{, if } x = (e, 2) \end{cases}$$

Note that u is a total well-defined function, i.e., it is total by the case distinction in its definition and it is also well-defined because p' and m' are. We must now show that that $u \circ p^* = p'$ and $u \circ m^* = m'$. Again, we show here explicitly for u_V, since for u_E the proof is similar. First we show that $u_V \circ m^*_V = m'$ and distinguish two cases:

Case $v = p_V(v')$. Then it holds that

$$\begin{array}{rcll} u_V(m^*_V(v)) & = & u_V(m^*_V(p_V(v'))) & \text{(substitution)} \\ & = & u_V(p^*_V(m_V(v'))) & \text{(definition of } m^*) \\ & = & u_V(([m_V(v')]_{\equiv}, 1)) & \text{(definition of } p^*_V) \\ & = & p'_V(m_V(v')) & \text{(definition of } u_V) \\ & = & m'_V(p_V(v')) & \text{(assumption)} \\ & = & m'_V(v) & \text{(substitution)} \end{array}$$

Case $v \in R_V - p_V(L_V)$. Then it holds that

$$\begin{array}{rcll} u_V(m^*_V(v)) & = & u_V((v,2)) & \text{(definition of } m^*_V\text{)} \\ & = & m'_V(v) & \text{(definition of } u_V\text{)} \end{array}$$

Now, to see that $u_V \circ p^*_V = p'_V$, note that

$$\begin{array}{rcll} u_V(p^*_V(v)) & = & u_V(([v]_{\equiv},1)) & \text{(definition of } p^*_V\text{)} \\ & = & p'_V(v) & \text{(definition of } u_V\text{)} \end{array}$$

It remains to show that u is in fact a graph morphism, that is to say, that u is compatible with source, target and labeling maps. However, note that this follows directly from the fact that u is constructed from p' and m'. In any case, we show here explicitly the compatibility of u w.r.t. the source map.

According to the construction of H we distinguish two cases:

Case $e = ([e']_{\equiv}, 1)$ with $e' \in G_E$. Then we have that

$$\begin{array}{rcll} s^{H'}(u_E(e)) & = & s^{H'}(u_E(([e']_{\equiv},1))) & \text{(substitution)} \\ & = & s^{H'}(p'_E(e')) & \text{(definition of } u_E\text{)} \\ & = & p'_V(s^G(e'))) & (p' \text{ is a graph morphism}) \\ & = & u_V(p^*_V(s^G(e'))) & \text{(assumption)} \\ & = & u_V(s^H(p^*_E((e'))) & (p^* \text{ is a graph morphism}) \\ & = & u_V(s^H(([e']_{\equiv},1))) & \text{(definition of } p^*_E\text{)} \\ & = & u_V(s^H(e)) & \text{(substitution)} \end{array}$$

Case $e = (e', 2)$ with $e' \in R_E - p_E(L_E)$. Then we have:

$$\begin{array}{rcll} s^{H'}(u_E(e)) & = & s^{H'}(u_E((e',2)) & \text{(substitution)} \\ & = & s^{H'}(m'_E((e',2))) & \text{(definition of } u_E\text{)} \\ & = & m'_V(s^R(e')) & (m' \text{ is a graph morphism}) \\ & = & u_V(m^*_V(s^R(e'))) & \text{(assumption)} \\ & = & u_V(s^H(m^*_E(e'))) & (m^* \text{ is a graph morphism}) \\ & = & u_V(s^H((e',2)) & \text{(definition of } m^*_E\text{)} \\ & = & u_V(s^H(e)) & \text{(substitution)} \end{array}$$

Thus u is a graph morphism and we have that $u \circ m^* = m'$ as well as $u \circ p^* = p'$. It still remains to show that u is the unique graph morphism such that the previous equations hold. Thus, suppose that there exists another $u' : H \to H'$ such that $u' \circ m^* = m' = u \circ m^*$ and $u' \circ p^* = p' = u \circ p^*$. Now, according to the first part of this theorem, m^* and p^* are jointly surjective. Therefore, by proposition A.2.11, we have that $u = u'$.

□

Corollary 2.1 (Pushouts in $\mathbf{Graph_L}$). *The derived graph H together with the graph morphisms $m^* : R \to H$ and $p^* : G \to H$ is a pushout of $p : L \to R$ and $m : L \to R$ in the category* $\mathbf{Graph_L}$*, written $(H, p^* : G \to H, m^* : R \to H)$.*

Proof: Directly from definition A.3.10 and theorem 2.2.6.

Remarks 2.2.7

1. In the derived graph H, we want to be sure that only items coming from L are glued together, and that G does not contain other items which does not come from G or R. These both requirements are expressed by the universal property of G, when compared with any other G' satisfying a similar commutativity as H.

2. Since pushouts are unique up to isomorphism (proposition A.3.11) it follows that H is also unique up to isomorphism.

3. It is advisable to use the explicit version in the sense of 2.2.1 for explicit constructions and the categorical version for proofs of theorems (see section 4).

We have now come to the point where we can generalize the well-know concepts of derivations and languages (as they are presented in formal language theory, e.g., [**?**]) to graphs.

Definition 2.2.8 (Direct derivation) *A direct derivation via a non-deleting rule $p : L \to R$ based on a match $m : L \to G$ is a pushout $(H, p^* : G \to H, m^* : R \to H)$ in the category* $\mathbf{Graph_L}$. *We say also that H is directly derived from G via p and denote it by $G \Longrightarrow_p H$.*

Definition 2.2.9 (Derivation sequence) *Let R be a set of rules. If a graph H is directly derived from a graph G with a rule from R then we write this as $G \Longrightarrow_R H$. A sequence of such derivations is called* **derivation sequence**, *i.e., $G_0 \Longrightarrow_R G_1 \ldots \Longrightarrow_R G_n$ is a derivation sequence of length n.*

Definition 2.2.10 (Graph grammar and graph language) *A* **graph grammar** *$GG = (R, G_0)$ consists of a set of rules R and a start graph G_0. $L(GG)$ is the graph language generated by this grammar, i.e., $L(GG) = \{H | G_0 \overset{*}{\Longrightarrow}_R H\}$, where $\overset{*}{\Longrightarrow}$ is the reflexive, transitive closure of the relation $\Longrightarrow$.*

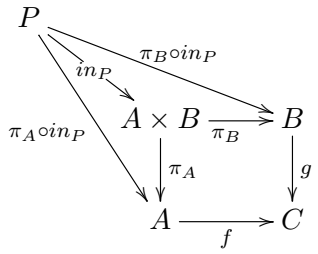
As a last result of this section, we show that $\mathbf{Graph_L}$ has also pullbacks (see A.3.15), a fact that we will need in the last section of this chapter.

Proposition 2.2.11 $\mathbf{Graph_L}$ *has pullbacks.*

Proof: By A.3.17, pullbacks can be constructed by products and equalizers. We construct first pullbacks of (sets of) edges and vertices in the category **Set**. Therefore, given graph morphisms $f : A \to C$ and $g : B \to C$ we define the graph pullback $P = (P_V, P_V, s^P, t^P, a^P, n^P)$ as follows:

$P_V = \{(v, v') \in A_V \times B_V | f_V(v) = g_V(v')\}$
$P_E = \{(e, e') \in A_E \times B_E | f_E(e) = g_E(e')\}$

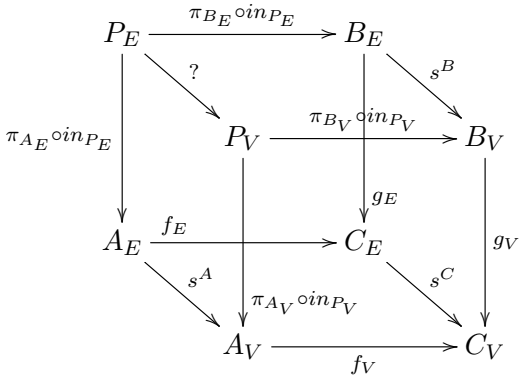
The equalizers in **Set** are just inclusions $in_{P_V} : P_V \to A_V \times B_V$ and $in_{P_E} : P_E \to A_E \times B_E$, which makes $in_P = (in_{P_V}, in_{P_E}) : P \to A \times B$ trivially a graph morphism. Now $\pi_A \circ in_P$ is a graph morphism from P to A and $\pi_B \circ in_P$ is a graph morphism from P to B such that the following diagram commutes.



The source and target maps are defined in the following way:

$$\begin{array}{lcll} s^P : P_E \to P_V & ::= & s^P(e,e') & = & (s^A \circ \pi_{A_E} \circ in_{P_E}(e,e'), s^B \circ \pi_{B_E} \circ in_{P_E}(e,e')) \\ t^P : P_E \to P_V & ::= & t^P(e,e') & = & (t^A \circ \pi_{A_E} \circ in_{P_E}(e,e'), t^B \circ \pi_{B_E} \circ in_{P_E}(e,e')) \end{array}$$

However, in order to be sure that P is a graph we must show well-definedness of these functions. We show here explicitly for s^P (a^P can be verified in a similar way). For well-definedness of s^P consider the following diagram, where front and back squares are pullbacks by construction and bottom and right squares commute because f and g are graph morphisms:

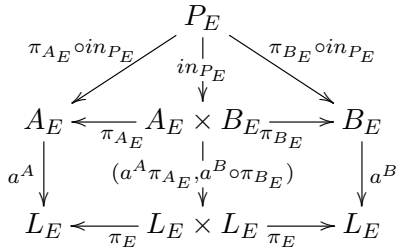


We calculate as follows:

$$\begin{array}{rcll} g \circ s^B \circ \pi_{B_E} \circ in_{P_E} & = & s^C \circ g_E \circ \pi_{B_E} \circ in_{P_E} & \text{(right square)} \\ & = & s^C \circ f_E \circ \pi_{A_E} \circ in_{P_E} & \text{(back square)} \\ & = & f_V \circ s^A \circ \pi_{A_E} \circ in_{P_E} & \text{(bottom square)} \end{array}$$

Now, since P_V is a pullback in **Set**, there is a unique $s^P : P_E \to P_E$ such that the left square commutes. Since our definition of s^P satisfies the commutativity requirement, well-definedness for the source mapping is established.

Now, consider the following diagram, which commutes by the universal properties of $A_E \times B_E$ and $L_E \times L_E$.:



We define $a^P : P_E \to L_E$ as

$$a^P(\langle e, e' \rangle) = a^A \circ \pi_{A_E} \circ in_{P_E} = a^B \circ \pi_{B_E} \circ in_{P_E}$$

Note that this definition really makes sense, since by definition, $\langle e, e' \rangle \in P_E$ iff $f_E(e) = g_E(e')$ and thus $a^A(e) = a^C(f_E(e)) = a^C(g_E(e')) = a^B(e')$.

The definition of n^P can be done in an analogous way. □

3 The double-pushout approach

In this section we admit for the first time "deleting" rules in our formal concept of graph derivation. For this purpose, we apply the so-called "double-pushout" approach for graph derivation. In the previous section, we have already described direct derivations by means of pushouts. However, at that time, we did not allow deleting rules. If we now admit rules that delete items, then we can describe this deletion by means of a second pushout. For this reason we have the name "double-pushout". Thus, a direct derivation happens in two steps: first deleting and second (as presented in the previous section) gluing and inserting.

On the other hand, by allowing deletion we have to face some problems. For instance, you cannot delete vertices without deleting at the same time their context (i.e., the edges which have these vertices as sources and/or targets), since otherwise, what remains is not a graph anymore. Besides, there are matches which can identify items that must be deleted with items that must remain. Should the corresponding item, in a derivation, remain or be deleted? Normally, one can proceed in two ways; either prohibiting the application of the match or allowing the application and later deciding how to cope with the conflict. In the double-pushout approach the first solution is adopted. With additional application conditions one can decide if the derivation can be carried out or not. We begin this section with the concepts of rules and matches and formulate the application conditions for a graph derivation. Adopting the same constructive approach of the last section, we first construct the result of a deletion, the so-called context graph. With these tools at our hands we finally define what a direct derivation in this approach is.

3.1 Rules and matches

Formally, a rule p that allows for deletion, is represented by a partial graph morphism $p : L \multimap R$, where the domain of definition of p (i.e., all $l \in L$ such that $p(l)$ is defined) represents

the items that must be preserved. Now, any partial graph morphism can be described by giving explicitly its domain of definition. Thus, let K denote denote the domain of definition of p. Then a partial graph morphism $p : L \multimap R$ can be represented as a span of total graph morphisms

$$L \overset{l}{\hookleftarrow} K \overset{r}{\to} R$$

where l is an inclusion and K represents exactly the items that we want to preserve. This discussion motivates the following

Definition 3.1.1 (Rule) *A* **rewriting rule** *(rule, for short) p, is a pair of total graph morphisms $l : K \to L$ and $r : K \to R$, where l is a injection. We call L the* **left side***, R the* **right side***, K the* **gluing graph** *(or* **interface** *graph), and write $p = (L \overset{l}{\leftarrow} K \overset{r}{\to} R)$. If r is also injective, then the rule is called* **injective***. If $L = R = K = \emptyset$, then the rule is called* **empty***.*

Remark 3.1.2 Each non-deleting rule (see 2.1.1) $q : L \to R$ can also be represented as a rule in the above sense, i.e., the corresponding rule is simply $p = (L \overset{id_L}{\leftarrow} L \overset{q}{\to} R)$.

Now that we have introduced the concept of a rule, we may start to clarify what a match in this approach should be. In the previous section, each total graph morphism from the left side to the graph in which we wanted to perform a derivation was considered to be a valid match. However, here we cannot allow for this possibility, because of the problems which were briefly discussed in the introduction of this section. To make this more clear, consider the rule $q = (L \overset{l}{\leftarrow} K \overset{r}{\to} R)$ and the graph G as presented in Figure 5. What should be the result of the direct derivation? The vertex "2" must be deleted. However, this is not possible without at the same time deleting the edge "3", since otherwise the result would not be a graph. In order to avoid such conflict, it is necessary to require that for each node to be deleted also the edge which has this node as a source or target be also deleted. Once this condition is satisfied, there will be no "dangling" edges. This discussion motivates the following

Definition 3.1.3 (Dangling condition) *A graph morphism $m : L \to G$ satisfies the* **dangling condition** *for a rule $p = (L \overset{l}{\leftarrow} K \overset{q}{\to} R)$ if $\forall v \in m_V(L_V - l_V(K_V))$ the following holds: if $v = s^G(e)$ or $v = t^G(e)$ for $e \in G_E$ then $e \in m_E(L_E - l_E(K_E))$.*

Remark 3.1.4 This condition says simply, that if a vertex of G shall be deleted and this vertex is a source or a target of a edge in G, then information about the deletion of this edge should also exist.

In the dangling condition we have formulated our first application condition. Unfortunately, this condition alone is not enough in order to define the derivation uniquely. We still have to avoid another possible conflict, which can happen only if the match m is not injective. In any case, we cannot simply prohibit non-injective matches since in many situations they may be necessary. The real problem here, is the identification of a vertex which shall be deleted with another one which should be kept. In this case, one may not know if a deletion

Figure 5: Example for a dangling edge

Figure 6: Example for a conflict

should or should not take place (see Figure 6). To avoid this dilemma, we formulate our next application condition in the following

Definition 3.1.5 (Identification condition) *A graph morphism $m : L \to G$ satisfies the* **identification condition** *for a rule $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$, if $(\forall v_1 \in (L_V - l_V(K_V)) \wedge v_2 \in L_V$, $m_V(v_1) = m_V(v_2) \Rightarrow v_1 = v_2) \wedge (\forall e_1 \in (L_E - l_E(K_E)) \wedge e_2 \in L_E$, $m_E(e_1) = h_E(e_2) \Rightarrow e_1 = e_2)$.*

Remark 3.1.6 The above condition simply states that two items shall only be identified if they must be kept by the application of the rule.

Having formulated our two application conditions, we are now in conditions to present our next

Definition 3.1.7 (Match for a rule and gluing condition) *A (total) graph morphism $m : L \to G$ is called a* **match** *for a rule $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ in a graph G if both the dangling condition and the identification condition are satisfied. The two application conditions together are named* **gluing condition**.

3.2 Derivations

In the same way we have proceeded in the last section, we want now to construct a direct derivation from a graph G with a rule $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ and a match $m : L \to G$. This construction follows from two steps. In the first step we simply perform the deletion, i.e., we construct a graph D, which results by removing all items from G which shall be deleted according to the rule p. D itself will be called *context graph*. Besides, as we are going to see, there is a relation between the graph D and the graphs K and G, which can be described by graph morphisms $l^* : D \to G$ and $d : K \to D$. In the second step, we need just to consider $r : K \to R$ as a non-deleting rule and $d : K \to D$ as a match, and then derive the graph H as in 2.2.1.

Definition and proposition 3.2.1 (Construction of context graph) *If $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ is a rule and $m : L \to G$ is a match for p in G, then the* **context graph** *of a direct derivation is constructed as follows:*

$$D_V = (G_V - m_V(L_V)) + m_V(l_V(K_V))$$
$$D_E = (G_E - m_E(L_E)) + m_E(l_E(K_E))$$
$$s^D = s^G|_{D_E}, t^D = t^G|_{D_E}, n^D = n^G|_{D_V}, a^D = a^G|_{D_E}$$

Thus, the mappings s^D, t^D, a^D, n^D are just restrictions of the corresponding original ones in G.

Proof: We have to show that $\forall e \in D_E$, $s^G(e) \in D_V$ and $t^G(e) \in D_V$. Now, suppose that this is not true (proof by contradiction). Without loss of generality, assume that there exists an edge $e \in (G_E - m_E(L_E)) + m_E(l_E(K_E))$ such that $s^G(e) \notin (G_V - m_V(L_V)) + m_V(l_V(K_V))$.

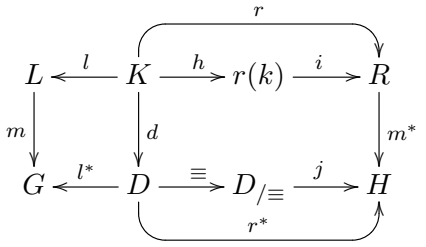
Now $e \in (G_E - m_E(L_E)) + m_E(l_E(K_E)) \Rightarrow e \notin m_E(l_E) \vee e \in m_E(l_E(K_E)) \Rightarrow e \notin m_E(L_E - l_E(K_E))$. Moreover, $s^G(e) \notin (G_V - m_V(L_V)) + m_V(l_V(K_V)) \Rightarrow s^G(e) \in m_V(L_V) \wedge s^G(e) \notin m_V(l_V(K_V)) \Rightarrow s^G(e) \in m_V(L_V - l_V(K_V))$. Now we have that $e \notin m_E(L_E - l_E(K_E)) \wedge s^G(e) \in m_V(L_V - l_V(K_V))$ which is a contradiction, since by assumption in the construction of D, $m : L \to G$ as match for p in G, must satisfy the dangling condition. The proof for t^G is analogous. □

Now, by construction of D, it is easy to see that D, K and G are somehow related. First, D is by construction a subgraph of G. Moreover, K is related to G by the composition $m \circ l$. Now, by construction of D, we have that $m_V(l_V(K_V)) \in D_V$ (resp. $m_E(l_E(K_E)) \in D_E$) and hence D and K are also related by the composition $m \circ l$. This discussion motivates the following

Proposition 3.2.2 *If $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ is a rule and $m : L \to G$ is a match for p in G, then there exists graph morphisms $d : K \to D$ and $l^* : D \to G$ such that $m \circ l = l^* \circ d$.*

Proof: Let l^* be the inclusion of D in G and $d = (d_V, d_E)$ be defined by $d_V ::= d_V(v) = (m \circ l)_V(v)$ and $d_E ::= d_E(v) = (m \circ l)_E(e)$. Moreover, l^* and d defined in this way are trivially graph morphisms. The commutativity also holds, since, for instance, $l^*_V \circ d_V(v) = l^*_V \circ (m \circ l)_V(v) = l^*_V(m_V(l_V(v))) = m_V \circ l_V(v)$ (and the same for $l^*_E \circ d_E(e) = m_E \circ l_E(e)$).□

Summarizing everything we have seen up to know, we have the commutativity of the following diagram:



Proposition 3.2.3 *The context graph D as constructed in 3.2.1 is a pushout complement, i.e., given a rule $p = (L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R)$ and a match $m : L \to G$, then $(G, l^* : D \to G, m : L \to G)$ is a pushout of l and d in $\mathbf{Graph_L}$.*

Proof idea: Construct a pushout G' of l and d according to 2.2.1 and show that G and G' are isomorphic. See also [**?**]. □

In other words, if the gluing condition is satisfied then the context graph D is a pushout complement in the sense of 3.2.3. On the other hand, the pushout properties of G already imply that the gluing condition is satisfied, as shown by the following

Proposition 3.2.4 (Characterization of gluing condition) *If $(G, l^* : D \to G, m : L \to G)$ is a pushout of l and d in $\mathbf{Graph_L}$, then m satisfies de gluing condition for l.*

Proof: First we show that m satisfies the dangling condition. We prove this by contradiction. Thus, let $v = m_V(v_1)$ with $v_1 \notin l_V(K_V)$ and suppose, without loss of generality, that $v = s^G(e)$ for $e \in G_E$ and $e \notin m_E(L_E - l_E(K_E)$. Then we have to distinguish two cases:

Case1 : $e \in m_E(l_E(K_E))$, i.e., $e = m_E(e_1)$ for $e_1 \in L_E$ and $e_1 = l_E(e_2)$ for $e_2 \in K_E$. Then it holds that

$$\begin{array}{rcll} m_V(l_V(s^K(e_2))) & = & m_V(s^L(l_E(e_2))) & (l \text{ is a graph morphism}) \\ & = & m_V(s^L(e_1)) & \text{(substitution)} \\ & = & s^G(m_E(e_1)) & (m \text{ is a graph morphism}) \\ & = & s^G(e) & \text{(substitution)} \\ & = & v & \text{(assumption)} \end{array}$$

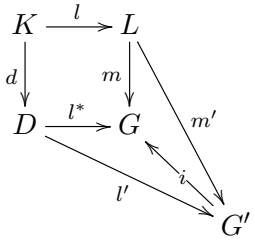
Thus, $v \in m_V(l_V(K_V))$ which is a contradiction to the assumption that $v \in m_V(L_V - l_V(K_V))$.

Case2 : $e \notin m_E(L_E)$. According to 2.2.6.1, m and l^* are jointly surjective. Therefore, there must exist $e_3 \in D_E$ with $l^*_E(e_3) = e$ (since by assumption $e \notin m_E(L_E)$). Now observe that

$$\begin{array}{rcll} l^*_V(s^D(e_3)) & = & s^G(l^*_E(e_3)) & (l \text{ is a graph morphism}) \\ & = & s^G(e) & \text{(substitution)} \\ & = & v & \text{(assumption)} \end{array}$$

Now v is in the image of both m_V (by assumption $v = m_V(v_1)$) and l^*_V (calculation above), i.e., $m_V(v_1) = v = l^*_V(s^D(e_3))$. By construction of l^* and m in 2.2.4 (taking $l^* = p^*$ and $m = m^*$), there exists v_2 in K_V with $l_V(v_2) = v_1$ and $d_V(v_2) = s^D(e_3)$, i.e., $m_V(l_V(v_2)) = l^*_V(d_V(v_2))$. Now $l_V(v_2) = v_1 \wedge v_1 \notin l_V(K_V)$ is the desired contradiction.

We have now to show that m also satisfies the identification condition. We show here explicitly the proof for the identification of vertices, since the case for the identification of edges can be treated similarly. Therefore, consider the following diagram:



Let $(G', m' : L \to G', l' : D \to G)$ be a pushout of l and d according to 2.1. Since pushouts are unique up to isomorphism there exists an isomorphism $i : G' \to G$ with $i \circ m' = m$ and $i \circ l' = l^*$. The identification condition states that $\forall v_1 \in (L_V - l_V(K_V))$ and $v_2 \in L_V$, $m_V(v_1) = m_V(v_2) \Rightarrow v_1 = v_2$. Now observe that $m_V(v_1) = m_V(v_2) \Rightarrow i_V \circ m'_V(v_1) = i_V \circ m'_V(v_2)$ (since $i \circ m' = m$). Now, since i is an isomorphism then it is also an monomorphism (A.2.7) and hence $m'_V(v_1) = m'_V(v_2)$. By construction of m' in 2.2.4 we have (since $v_1 \notin l_V(K_V)$) that $m'_V(v_1) = (v_1, 2) = (v_2, 2) = m'_V(v_2)$ and hence that $v_1 = v_2$.

□

The last meaningful result of this section is the abstract characterization of the context graph.

Proposition 3.2.5 (Uniqueness of the context graph) *If $p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ is a rule and $m : L \rightarrow G$ is a match, then the pushout complement of l and m $(D, l^* : D \rightarrow G, d : K \rightarrow D)$ is unique up to isomorphism.*

Proof: Omitted. □

Now we can characterize both the first step of the derivation, namely the deletion, as well the second one, i.e., gluing and insertion, in a abstract way or in other words, uniquely up to isomorphism. This abstract characterization is used in the following

Definition 3.2.6 (Direct derivation) *A* **direct derivation** *via rule $p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ based on a match $m : L \rightarrow G$ consists of the following two pushouts $(PO)_1$ and $(PO)_2$ in the category* $\mathbf{Graph_T}$. *We again write $G \Longrightarrow_p H$ to say that H is directly derived from G via p.*

$$\begin{array}{ccccc} L & \xleftarrow{l} & K & \xrightarrow{r} & K \\ {\scriptstyle m}\downarrow & (PO)_1 & \downarrow{\scriptstyle d} & (PO)_2 & \downarrow{\scriptstyle d^*} \\ G & \xleftarrow[l^*]{} & D & \xrightarrow[r^*]{} & H \end{array}$$

The concepts of derivation sequence, graph grammar and graph language introduced in the last section, can be transfered directly, without any changes.

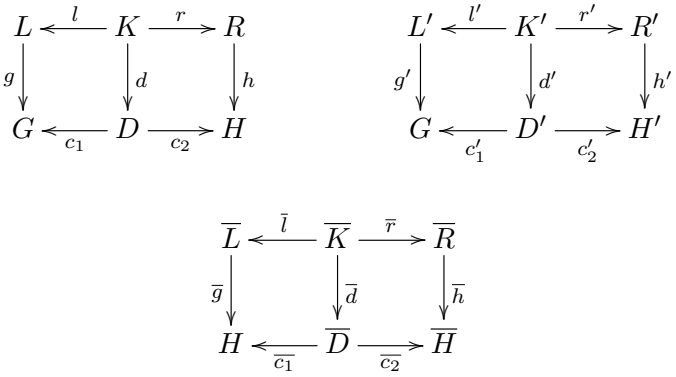
4 Results

In this section we consider Church-Rosser properties of graph derivations and parallel productions and derivations. We start with the notions of parallel and sequential independence in two different formulations. The first one is more intuitive, the second one is categorical and will be used to prove the Church-Rosser-Property I in 4.2.1.

4.1 Independence of derivations

Definition 4.1.1 Independence. *Let $G \Longrightarrow_p H$, $G \Longrightarrow_{p'} H'$ and $H \Longrightarrow_{\overline{p}} \overline{H}$ be direct derivations according to the following gluing diagrams (where p and p' are injective productions*[3] *in the sense of 3.1.1):*

[3] Also called **fast** productions in the literature.



1. *Two derivations $G \Longrightarrow H$ and $G \Longrightarrow H'$ via p and p' are said to be* **parallel independent** *if the intersection of L and L' in G (which are the occurrences of p and p' in G) consists of common gluing points. This means precisely that*

$$g(L) \cap g'(L') \subseteq g(l(K)) \cap g'(l'(K))$$

2. *The derivation sequence $G \Longrightarrow H \Longrightarrow \overline{H}$ via $(p, \overline{p})$ based on g and $\overline{g}$ is called* **sequential independent** *if*

$$h(R) \cap \overline{g}(\overline{L}) \subseteq h(r(K)) \cap \overline{g}(\overline{l}(\overline{K}))$$

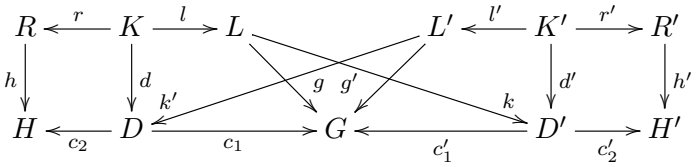
Remarks 4.1.2

1. When we say that two productions p and p' are parallel independent, we mean that the intersection of the two left sides of p and p' in G must be included in the intersection of the two corresponding interface graphs K, K' in G. This means that the occurrence L of p (respectively L' of p') in G will not be destroyed by the other derivation $G \Longrightarrow H'$ via p' (respectively $G \Longrightarrow H$ via p), since the gluing points are preserved during the derivation process. Hence, we can also apply p in H' (respectively p' in G). In other words, L is included in the context graph D' of $G \Longrightarrow H'$ (respectively L' is included in the context D of $G \Longrightarrow H$).

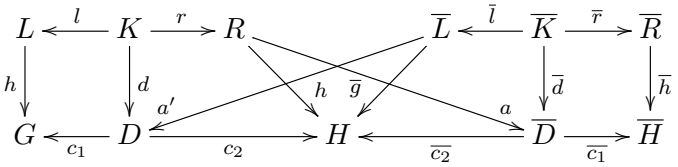
2. When we say that two productions p and $\overline{p}$ are sequential independent, this means that the intersection of the right side of the first production with the left side of the second production in H consists only of gluing points. This means that $\overline{p}$ does not need the items which are generated by p_1 and that $\overline{p}$ does not remove items which are used by p_1.

Proposition 4.1.3 (Categorical formulation of independence). *Let p, p' and $\overline{p}$ be productions according to 4.1.1.*

1. *Two derivations $G \Longrightarrow H$ and $G \Longrightarrow H'$ via p and p' based on g and g' are said to be* **parallel independent** *if and only if there are graph morphisms $k : L \to D'$ and $k' : L' \to D$ such that $g = c'_1 \circ k$ e $g' = c_1 \circ k'$ (see the next diagram).*



2. *A derivation sequence $G \Longrightarrow H \Longrightarrow \overline{H}$ via $(p, \overline{p})$ based on g and $\overline{g}$ is called* **sequential independent** *if there are graph morphisms $a : R \to \overline{D}$ and $a' : \overline{L} \to D$ such that $h = \overline{c_2} \circ a$ and $\overline{g} = c_2 \circ a'$ (see next diagram).*

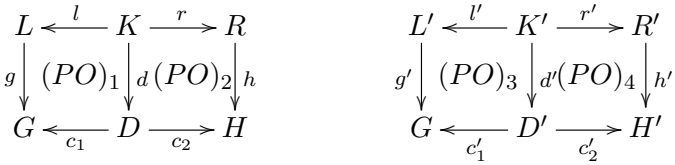


Proof sketch : Consider the case of parallel independence (the reasoning for sequential independence is similar). If the occurrences of L and L' in G are disjoint then clearly D' contains $g(L)$ since removing $g(L')$ won't remove L. In case the occurrences are not disjoint but their intersection consists of common gluing points, then we still have $g(L)$ contained in D' since D', by construction, contains the gluing points of p'. Now we define $k(x) = g(x)$ (where x stands for and edge or vertex of L) and we have $c'_1(k'(x)) = c'_1(g(x)) = g(x)$. The same argument applies to show that L' is contained in D. □

4.2 The Church-Rosser theorems

Theorem 4.2.1 (Church-Rosser-Property I). *Let p and p' be injective rules and $G \Longrightarrow H$ and $G \Longrightarrow H'$ parallel direct derivations via p respectively p'. Then there exists a graph X such that both sequences in Figure 7 are sequential independent.*

Proof: By definition 3.2.6, the direct derivations $G \Longrightarrow H$ via p and $G \Longrightarrow H'$ via p' are represented by the following diagrams:



Combining $(PO)_1$ with $(PO)_3$ and using the categorical formulation of parallel independence in 4.1.3.1, we have the following diagram:

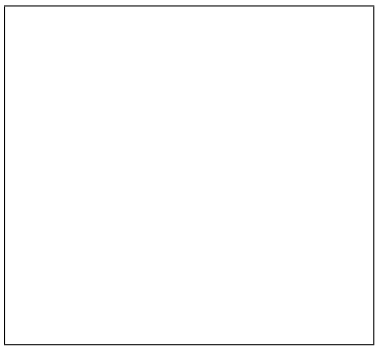
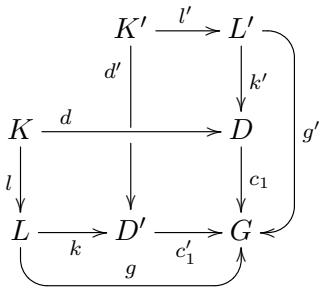
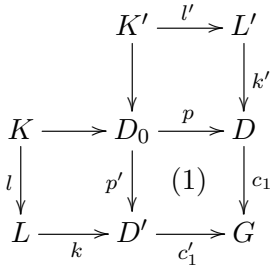


Figure 7: Church-Rosser Property



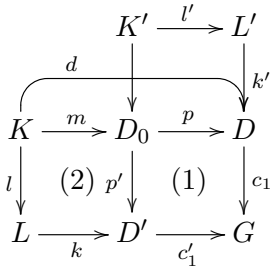
Now we construct the pullback of c_1 and c_1' (D_0) in the following diagram:



where D_0 is the intersection of D and D' (since c_1 and c_1' are just inclusions). Now, note that

$c_1 \circ d =$
$= g \circ l$ (by $(PO)_1$)
$= (c_1' \circ k) \circ l$ (by 4.1.3.1)
$= c_1' \circ (k \circ l)$ (associativity)

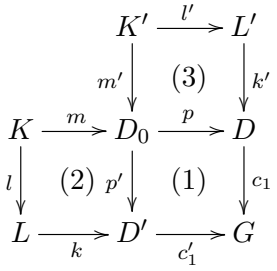
However, by the universal property of the pullback (1), there is a unique $m : K \to D_0$ such that $d = p \circ m$ and $k \circ l = p' \circ m$. The last equality implies that (2) commute in the following diagram:



Reasoning in the same way, we have that

$c'_1 \circ d' =$
$= g' \circ l'$ (by $(PO)_3$)
$= (c_1 \circ k') \circ l'$ (by 4.1.3.1)
$= c_1 \circ (k' \circ l')$ (associativity)

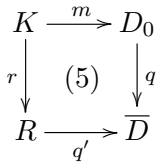
Now, by the universal property of the pullback (1), we have that there exists a unique $m' : K' \to D_0$ such that $d' = p' \circ m'$ and $k' \circ l' = p \circ m'$. The last equality implies that (3) commutes in the following diagram:

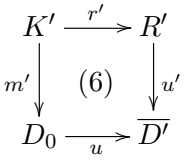


In this way, the pushouts $(PO)_1$ and $(PO)_3$ are decomposed by the diagrams (1) and (2), respectively (1) and (3). We will show separately that $(1), (2)$ and (3) are pushouts.

Since c_1 and c'_1 are injective (inclusions), the pullback (1) is also a pushout by A.3.16. Since c_1 and c'_1 are injective and D_0 is the intersection of D and D' it follows that p and p' are also injective. Now, since $(2) + (1) = (PO)_1$ is a pushout, (1) is a pushout and p is injective, it follows, from A.3.14, that (2) is also a pushout. Moreover, since $(3) + (1) = (PO)_3$ is a pushout, (1) is a pushout and c_1 is injective, it follows also from A.3.14 that (3) is also a pushout.

We now construct a similar "triple-PO-diagram", using the pushouts $(PO)_2$ and $(PO)_4$ instead of the pushouts $(PO)_1$ and $(PO)_3$. So, given the morphisms $m : K \to D_0, m' : K' \to D_0, r : K \to R$ and $r' : K' \to R'$, we construct the pushouts of m and r, respectively of m' and r', resulting in the following diagrams:





Then we construct the pushout X of $q : D_0 \to \overline{D}$ and $u : D_0 \to \overline{D'}$ obtaining the following diagram:

$$\begin{array}{ccc} D_0 & \xrightarrow{u} & \overline{D'} \\ {\scriptstyle q}\downarrow & (4) & \downarrow{\scriptstyle t} \\ \overline{D} & \xrightarrow[s]{} & X \end{array}$$

Combining the diagrams (4), (5) and (6), we obtain the following "triple-PO-diagram":

$$\begin{array}{ccccc} & & K' & \xrightarrow{r'} & R' \\ & & {\scriptstyle m'}\downarrow & (6) & \downarrow{\scriptstyle u'} \\ K & \xrightarrow{m} & D_0 & \xrightarrow{u} & \overline{D'} \\ {\scriptstyle r}\downarrow & (5) & {\scriptstyle q}\downarrow & (4) & \downarrow{\scriptstyle t} \\ R & \xrightarrow[q']{} & \overline{D} & \xrightarrow[s]{} & X \end{array}$$

Now note that the pushout $(PO)_2$ can be decomposed by the pushout (5) and by the factorization $d = p \circ m$ shown previously.

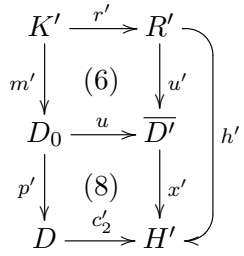
$$\begin{array}{ccc} K & \xrightarrow{r} & R \\ {\scriptstyle m}\downarrow & (5) & \downarrow{\scriptstyle q'} \\ D_0 & \xrightarrow{q} & \overline{D} \\ {\scriptstyle p}\downarrow & (7) & \downarrow{\scriptstyle x} \\ D & \xrightarrow{c_2} & H \end{array} \qquad h : R \to H$$

Now see that

$$\begin{aligned} & c_2 \circ p \circ m = \\ & = c_2 \circ d && \text{(definition of } d\text{)} \\ & = h \circ r && \text{(by } (PO)_2\text{)} \end{aligned}$$

But then, by the universal property of the pushout (5), we have that there is a unique $x : \overline{D} \to H$ such that $x \circ q' = h$ and $c_2 \circ p = x \circ q$. By construction, we know that (5) + (7) = $(PO)_2$ and (5) are pushouts. By proposition A.3.13.2 we have that (7) is also a pushout.

In the same way, we have that the $(PO)_4$ can be decomposed by (6) and by the factorization of $d' = p' \circ m'$ shown previously.



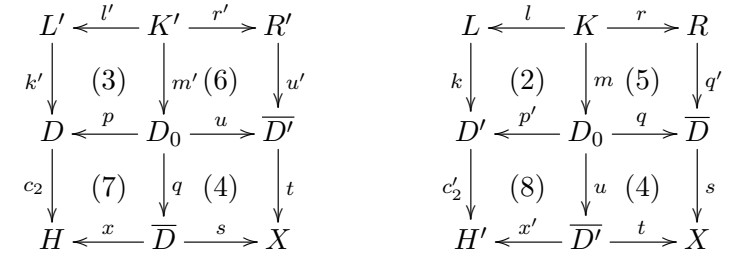
Now see that

$c_2' \circ p' \circ m' =$
$= c_2' \circ d'$ (definition of d')
$= h' \circ r'$ (by $(PO)_4$)

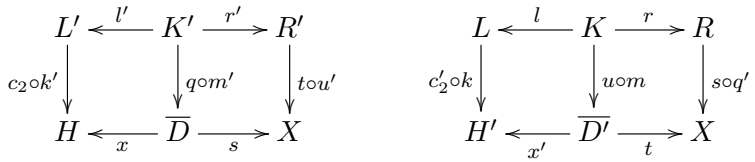
But then, by the universal property of pushout (6), there is a unique $x' : \overline{D}' \to H'$ such that $x' \circ u' = h'$ and $c_2' \circ p' = x' \circ u$. The last equality implies that (8) commutes.

By construction, we know that (6) + (8) = $(PO)_4$ and (6) are pushouts. By proposition A.3.13.2 we have that (8) is also a pushout.

Now we have all the pushout pieces to construct the direct derivations $H \Longrightarrow X$ via p' and $H' \Longrightarrow X$ via p. Therefore, consider the (two) following diagrams:



Now observe that, since all the squares in the previous two diagrams are pushouts, we have that the following diagrams are direct derivations $H \Longrightarrow X$ via p' and $H' \Longrightarrow X$ via p in the sense of 3.2.6.



It remains now only to show that the sequences $G \Longrightarrow H \Longrightarrow X$ via (p, p') and $G \Longrightarrow H' \Longrightarrow X$ via (p', p) are sequential independent. To see this, note that (5) + (7) together with (3) + (7) imply, by proposition 4.1.3.2, that the sequence $G \Longrightarrow H \Longrightarrow X$ is sequential independent, since there are $q' : R \to \overline{D}$ such that $h = x \circ q'$ and $k' : L \to D$ such that (trivially) $c_2 \circ k' = c_2 \circ k'$. In the same way, (6) + (8) together with (2) + (8) imply, by proposition 4.1.3.2, that the sequence $G \Longrightarrow H' \Longrightarrow X$ is also sequential independent.

Figure 8: Productions to exemplify the Church-Rosser-Property I

□

Remark 4.2.2 The proof of this theorem makes clear the importance of having a abstract characterization of a graph derivation by means of a colimit construction. Without this, we would have do deal explicitly with gluing constructions and proof obligations in the sense of 2.2.1 and 2.2.4 which would make the proof enormous and error-prone, leading to lack of clarity and unreadability. Moreover, without the universal properties of colimits, we would not be able to use the composition and decomposition properties of pushouts which were of immeasurable help in this proof.

Example 4.2.3 Consider the productions p and p' in Figure 8 and the graph G in Figure 9. According to 4.1.3.1, the derivations $G \Longrightarrow H$ via p and $G \Longrightarrow H'$ via p' are parallel independent. Now, the existence of derivations $H \Longrightarrow X$ via p' and $H' \Longrightarrow X$ via p such that $G \Longrightarrow H \Longrightarrow X$ and $G \Longrightarrow H' \Longrightarrow X$ become sequential independent follows directly from the above theorem.

Dualizing the previous result lead us to the following

Theorem 4.2.4 (Church-Rosser-Property II) *Given a sequentially independent sequence $G \Longrightarrow_p H \Longrightarrow_{p'} X$ with p and p' injective, there exists another sequentially independent sequence $G \Longrightarrow_{p'} H' \Longrightarrow_p X$ between the same graphs X. Moreover, $G \Longrightarrow_p H$ and $G \Longrightarrow_{p'} H'$ become parallel independent.*

Proof: See [**?**] □

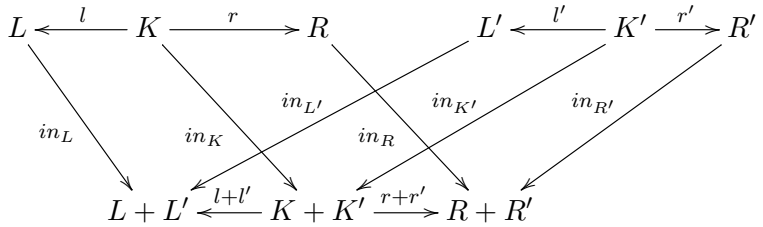
4.3 The parallelism theorem

Definition 4.3.1 *Let $p = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ and $p' = (L' \stackrel{l'}{\leftarrow} K' \stackrel{r'}{\rightarrow} R')$ be injective rules. Then the parallel production $p + p'$ is defined by*

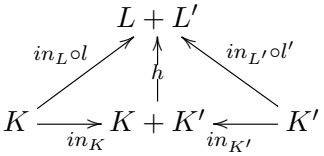
$$p + p' = (L + L' \stackrel{l+l'}{\leftarrow} K + K' \stackrel{r+r'}{\rightarrow} R + R')$$

where $+$ denotes disjoint union of graphs resp. graphs morphisms. A derivation $G \Longrightarrow X$ via $p + p'$ is called **parallel derivation**.

Remark 4.3.2 The above definition of parallel production can be given a categorical characterization by considering the following diagram:



Note that $l + l'$ and $r + r'$ are uniquely defined by the couniversal property of coproducts. For instance, well-definedness of $l + l'$ is shown by the following commutative diagram (similarly for $r + r'$):



where $h = in_L \circ l + in_{L'} \circ l' = l + l'$.

Parallel derivations and independent sequences are related in the following way:

Theorem 4.3.3 (Parallelism Theorem)

1. *ANALYSIS: Given a parallel derivation $G \Longrightarrow X$ via $p+p'$, there is a canonical analysis into two sequential independent sequences $G \Longrightarrow H \Longrightarrow X$ via (p, p') and $G \Longrightarrow H' \Longrightarrow X$ via (p', p).*

Figure 9: Example of the Church-Rosser-Property I

2. *SYNTHESIS: Given a sequentially undependent sequence* $G \Longrightarrow H \Longrightarrow X$ *via* (p, p')*, there is a canonical synthesis leading to one parallel derivation* $G \Longrightarrow X$ *via* $p + p'$*.*

3. *The operations ANALYSIS and SYNTHESIS are inverse of each other in the following sense: Given* p, p' *and* $p + p'$ *there is a bijective correspondence between sequentially independent sequences* $G \Longrightarrow H \Longrightarrow X$ *via* (p, p') *and parallel derivations* $G \Longrightarrow X$ *via* $p + p'$*.*

Proof: See [**?**], where it is shown that the parallelism theorem is just a special case of the concurrency theorem (see remarks below). □

Example 4.3.4 The parallel production $p + p'$ given by the disjoint union of the left hand sides, interfaces and right hand sides of p and p' in Figure 8 (which are sequential independent) leads in one direct derivation step from the left graph G to the right graph X in Figure 9.

Remark 4.3.5 Initially, in making clear what topics should be covered in this chapter, we had the intention to cover also aspects of concurrency. However, because of space limitations, we refer the reader to the literature in next section below, since a careful presentation of examples and a detailed explanation of the very technical constructions would require a chapter on its own. Therefore, we hope that the constructive approach adopted in this chapter may give to, the interested reader, enough confidence in order to grasp successfully further (and more involving) technical results.

5 Bibliographic Notes

This chapter presents only a small part of the whole set of concepts and results of the for graph grammars, although enough to show the enormous benefits from the use of category theory in this area. Especially, the DPA [**?**] has been developed for many years and presents results also for concurrency [**?**]), synchronization mechanisms [**?**], and implementation of abstract data types [**?**]. Theoretical Computer Science, v.109, n.1-2, 1992, is entirely devoted to the algebraic approach of graph grammars, especially the **single pushout approach**, a generalization of the double-pushout approach, which we have not covered in this chapter.

Acknowledgments: This work was developed within a German-Brazilian cooperation on Information Technology, in the context of GRAPHIT project, and partially supported by a CNPq-grant 200529/94-3 for Álfio Martini. This is part of a work which was initiated by Prof. Dr. Harmut Ehrig's Kloosterman Lecture "On the role of Category Theory in Computer Science" in Leiden, Holland, September, 1993, as well as the encouragement of Prof. Dr. Daltro Nunes concerning categorical methods in computer science. A very special "thank you" goes to Uwe Wolter, who carefully read this report, found numerous significant errors, and made very useful suggestions on how to correct them.

A Categorical Background

In this appendix we summarize some basic notions from category theory which are used in the previous chapters as well as a set of elementary results (propositions and theorems) concerning categorical constructions that are referenced in the proofs presented in this report. The complete proofs themselves, concerning each of this results, may be found, for instance, in [**?**]. Detailed introductions to this subject may be found in [**?**], [**?**], and [**?**].

A.1 Categories and diagrams

Definition A.1.1 *A category* $\mathbf{C}$ *comprises*

1. *a collection* $Ob(\mathbf{C})$ *of* **objects***;*
2. *a collection* $Mor(\mathbf{C})$ *of morphisms;*
3. *two operations* $dom, cod : Mor(\mathbf{C}) \to Ob(\mathbf{C})$ *assigning to each morphism* f *two objects, called repsectively* **domain** *and* **codomain** *of* f*;*
4. *a composition operator* $\circ : Mor(\mathbf{C}) \times Mor(\mathbf{C}) \to Mor(\mathbf{C})$ *assigning to each pair of morphisms* $\langle f, g\rangle$ *with* $dom(g) = cod(f)$ *a composite morphism* $g \circ f : dom(f) \to cod(g)$*, such that the following* **associative law** *holds:*

   For any morphisms f, g, h *in* $Mor(\mathbf{C})$ *such that* $cod(f) = dom(g) \wedge cod(g) = dom(h)$*:*

   $$h \circ (f \circ g) = (h \circ g) \circ h$$

5. *for each object* A*, an identity morphism* $id_A : A \to A$*, such that the following* **identity law** *holds:*

   For any morphism f *such that* $dom(f) = A \wedge cod(f) = B$*:*

   $$id_B \circ f = f \wedge f \circ id_A = f$$

Remarks A.1.2

1. Category theory is based on composition as a fundamental operation, in much the same way that classical set theory is based on the "element of" or membership relation
2. Categories will be denoted by uppercase boldface letters $\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$ from the beginning of the alphabet.
3. We use letters $A, B, \ldots, Y, Z$ from the alphabet (with subscripts when appropriate) to denote objects and lowercase letters $a, b, c, \ldots, f, g, h, \ldots, y, z$ (occasionally with subscripts) to denote morphisms in any category.
4. If $dom(f) = A \wedge cod(f) = B$ we write $f : A \to B$ to denote that f is a morphism from A to B.

CATEGORY	*OBJECTS*	*MORPHISMS*
Set	sets	total functions
FinSet	finite sets	total functions
Pfn	sets	partial functions
Rel	sets	binary relations
Mon	monoids	monoid homomorphisms
Poset	posets	monotonic functions
Grp	groups	group homomorphisms
Σ-Alg	Σ-Alg	Σ-homomorphisms
Cat$(SPEC)$	$SPEC$-algebras	$SPEC$-homomorphisms
Aut	finite automata	automata homomorphisms
Graph	directed graphs	graph morphisms

Table 1: Examples of categories

5. The collection of all morphisms with domain A and codomain B will be written as $\mathbf{C}(A, B)$ or as $Hom_{\mathbf{C}}(A, B)$.

6. Given a category $\mathbf{C}$, we may write "$\mathbf{C}$-object A" (resp. "$\mathbf{C}$-morphism $f : A \to B$") or "$A \in Ob(\mathbf{C})$" (resp. "$f \in \mathbf{C}(A, B)$") to denote that A is a object of $\mathbf{C}$ (resp. f is a morphism from A to B in $\mathbf{C}$).

Example A.1.3 Table 1 lists some categories by specifying their objects and morphisms.

Definition A.1.4 *A* **diagram** *in a category* $\mathbf{C}$ *is a collection of vertices and directed edges, consistently labeled with objects and morphisms of* $\mathbf{C}$*, where "consistently" means that if an edge in the diagram is labeled with a morphism* f*, and* f *has domain* A *and codomain* B*, then the endpoints of this edge must be labeled with* A *and* B*.*

Definition A.1.5 *A diagram in a category* $\mathbf{C}$ *is said to* **commute** *(or is said to be* **commutative***) if, for each pair of vertices* v_1 *and* v_2 *labeled by* X *and* Y*, all paths in the diagram from* X *to* Y *are equal, in the sense that each path in the diagram determines a composite morphism and these composites are equal in* $\mathbf{C}$*.*

As an example, saying that the diagram below commutes is exactly the same as saying that $f \circ g' = g \circ f'$.

$(0,700)(1,0)[X`Z`f']6001a(0,700)(0,-1)[\quad `W`g']6001l(600,700)(0,-1)[\quad `Y`g]6001r(0,100)(1,0)[\quad `\quad `f]600$

A.2 Special kinds of morphisms

Definition A.2.1 *Let* $\mathbf{C}$ *be a category. A morphism* $f \in \mathbf{C}(B, C)$ *is a* **monomorphism** *(or is* **monic***) if for any pair of morphisms* $g \in \mathbf{C}(A, B)$ *and* $h \in \mathbf{C}(A, B)$*, the equality* $f \circ g = f \circ h$ *implies* $g = h$ *(i.e.,* f *is left-cancelable with respect to composition).*

Example A.2.2 In **Set** a function is a monomorphism iff it is injective.

Definition A.2.3 *Let* $\mathbf{C}$ *be a category. A morphism* $f \in \mathbf{C}(A, B)$ *is an epimorphism (or* **comonomorphism** *or only* **epic***) if for any pair of morphisms* $g \in \mathbf{C}(B, C)$ *and* $h \in \mathbf{C}(B, C)$*, the equality* $g \circ f = h \circ f$ *implies* $g = h$ *(i.e.,* f *is right-cancelable with respect to composition).*

Example A.2.4 In **Set** a function is an epimorphism iff it is surjective.

Definition A.2.5 *Let* $\mathbf{C}$ *be a category. A morphism* $f \in \mathbf{C}(A, B)$ *is an* **isomorphism** *(or is* **iso***) if there exists a morphism* $f^{-1} \in \mathbf{C}(B, A)$*, called* inverse *of* f*, such that* $f \circ f^{-1} = id_B$ *and* $f^{-1} \circ f = id_A$*. In this case, the objects* A *and* B*, are said to be isomorphic, which we represent as* $A \cong B$*.*

Example A.2.6 In **Set** a function is an isomorphism iff it is bijective.

Proposition A.2.7 *In any category* $\mathbf{C}$ *if* $f : A \to B$ *is an isomorphism then* f *is also a monomorphism and an epimorphism.*

Definition A.2.8 *A* **sink** *is a pair* $((f_i)_{i \in I}, A)$ *consisting of an object* A *(the* **codomain** *of the sink) and a family of morphisms* $f_i : A_i \to A$ *indexed by some class* I*. The family* $(A_i)_{i \in I}$ *is called the domain of the sink.*

Definition A.2.9 *For a sink* $\mathcal{S} = ((f_i)_{i \in I}, A)$ *and a morphism* $f : A \to B$*, then the composition* $f \circ \mathcal{S}$ *is defined as* $f \circ f_i$ *for each* $i \in I$*.*

Definition A.2.10 *A* **sink** $\mathcal{S} = ((f_i)_{i \in I}, A)$ *is called an* **epi-sink** *provided it can be cancelled from the right, i.e., for any pair of morphisms* $h, k : A \to B$ *the equation* $h \circ \mathcal{S} = k \circ \mathcal{S}$ *(i.e.,* $h \circ f_i = k \circ f_i$ *for each* $i \in I$*) implies* $h = k$*.*

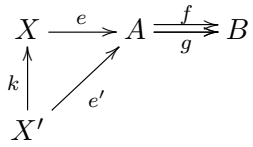
Proposition A.2.11 *In any construct*[4] *all jointly surjective sinks are epi-sinks.*

A.3 Universal constructions

Definition A.3.1 *A morphism* $e \in \mathbf{C}(X, A)$ *is an equalizer of a pair of morphisms* $f \in \mathbf{C}(A, B)$ *and* $g \in \mathbf{C}(A, B)$ *if*

1. $f \circ e = g \circ e$*;*
2. *whenever there is a morphism* $e' \in \mathbf{C}(X', A)$ *which satisfies* $f \circ e' = g \circ e'$*, then there exists a unique morphism* $k \in \mathbf{C}(X', X)$ *such that* $e \circ k = e'$*, as shown by the following diagram:*

[4] Categories of structured sets and structure-preserving function between them.



Example A.3.2 In **Set**, equalizers can be constructed in the following way:

Given two functions $f, g : A \to B$, we can construct the subset E of A in which f and g are equal, i.e.:

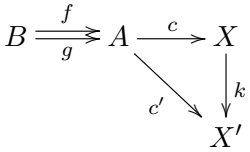
$$E = \{a : a \in A \wedge f(a) = g(a)\}$$

Then, the inclusion function $in_E : E \to A$ equalizes f and g, i.e., $f \circ in_E(a) = f(in_E(a)) = f(a) = g(a) = g(in_E(a)) = g \circ in_E(a)$ for all $a \in E$.

The dual notion for equalizers is represented in the following

Definition A.3.3 *A morphism $c \in \mathbf{C}(A, X)$ is a* **coequalizer** *of a pair of morphisms $f \in \mathbf{C}(B, A)$ and $g \in \mathbf{C}(B, A)$ if*

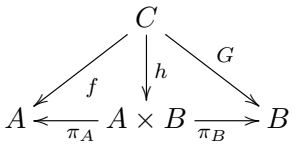
1. *$c \circ f = c \circ g$;*
2. *whenever $c' \in \mathbf{C}(A, X')$ satisfies $c' \circ f = c' \circ g$, then there exists a unique morphism $k \in \mathbf{C}(X, X')$ such that $k \circ c = c'$, as shown in the next diagram.*



Example A.3.4 In **Set** coequalizers can be constructed by using equivalence relations. Given two functions $f, g : X \to A$, the intuitive idea is to identify $f(x)$ with $g(x)$ for all $x \in X$. First we construct the relation $R = \{\langle f(x), g(x)\rangle\}$ for all $x \in X$. Next we define R' as being the least equivalence relation which contains R' (i.e., the reflexive, symmetric and transitive closure of R). Then we define the coequalizer $nat : A \to A/R'$ by $nat(a) = [a]$ (also called "natural" mapping).

Proposition A.3.5 *Every coequalizer is an epimorphism.*

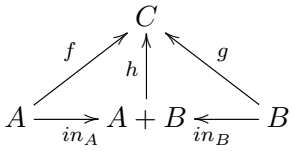
Definition A.3.6 . *A* **product** *of two objects A and B in a category $\mathbf{C}$, is a $\mathbf{C}$-object $A \times B$, together with two projection morphisms $\pi_A \in \mathbf{C}(A \times B, A)$ and $\pi_B \in \mathbf{C}(A \times B, B)$ such that, for any object C and pair of morphisms $f \in \mathbf{C}(C, A)$ and $g \in \mathbf{C}(C, B)$, there exists exactly one morphism $h \in \mathbf{C}(C, A \times B)$ such that the following diagram commutes, i.e., such that $\pi_A \circ h = f$ and $\pi_B \circ h = g$.*



Example A.3.7 In **Set** the product of two sets A and B is given by the cartesian product $A \times B$.

The dual notion for a product is a coproduct or sum of objects, which we present in the following

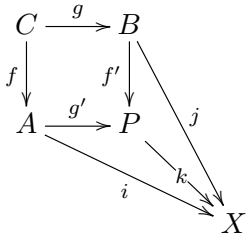
Definition A.3.8 A **coproduct** of two objects A and B in a category $\mathbf{C}$ is a $\mathbf{C}$-object $A+B$ together with two **injection morphisms** $in_A \in \mathbf{C}(A, A+B)$ and $in_B \in \mathbf{C}(B, A+B)$, such that for any $\mathbf{C}$-object C and pair of morphisms $f \in \mathbf{C}(A, C)$ and $g \in \mathbf{C}(B, C)$, there exists exactly one morphism $h \in \mathbf{C}(A+B, C)$, such that $h \circ in_A = f$ and $h \circ in_B = g$ (see next diagram).



Example A.3.9 In **Set**, the coproduct of two sets A and B is given by the disjoint union $A + B$.

Definition A.3.10 *A* **pushout** *of a pair of morphisms $f \in \mathbf{C}(C, A)$ and $g \in \mathbf{C}(C, B)$ is a $\mathbf{C}$-object P and a pair of morphisms $g' \in \mathbf{C}(A, P)$ and $f' \in \mathbf{C}(B, P)$, such that*

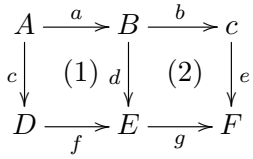
1. *$g' \circ f = f' \circ g$;*
2. *for any other $\mathbf{C}$-object X and pair of morphisms $i \in \mathbf{C}(A, X)$, $j \in \mathbf{C}(B, X)$ with $i \circ f = j \circ g$, there exists only one morphism $k : P \to X$ such that $i = k \circ g'$ and $j = k \circ f'$, as shown by the following diagram:*



Proposition A.3.11 *Pushouts are unique up to isomorphism.*

Proposition A.3.12 *Pushouts can be constructed from coproducts and coequalizers.*

Proposition A.3.13 *Consider the following diagram:*

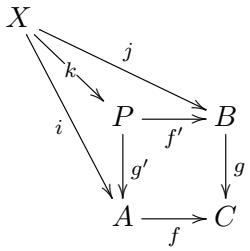


1. *If both squares are pushouts, then the outer rectangle is also a pushout.*
2. *If the outer rectangle is a pushout, (2) commutes and (1) is also a pushout, then (2) is also a pushout.*

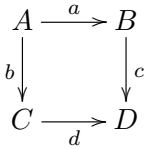
Proposition A.3.14 Consider the above diagram. In $\mathbf{Graph_L}$ and **Set**, if (1)+(2) and (2) are pushouts, $b : B \to C$ is injective, then (1) is a pushout (see [**?**] and [**?**], for instance).

Definition A.3.15 *A* **pullback** *of a pair of morphisms $f \in \mathbf{C}(A, C)$ and $g \in \mathbf{C}(B, C)$ is a $\mathbf{C}$-object P and a pair of morphisms $g' \in \mathbf{C}(P, A)$ and $f' \in \mathbf{C}(P, B)$, such that*

1. *$f \circ g' = g \circ f'$;*
2. *for any other $\mathbf{C}$-object X and morphisms $i \in \mathbf{C}(X, A)$, $j \in \mathbf{C}(X, B)$ with $f \circ i = g \circ j$, there exists only one morphism $k : X \to P$ such that $i = g' \circ k$ e $j = f' \circ k$ as shown by the following diagram:*



Proposition A.3.16 In categories like **Set** and $\mathbf{Graph_L}$ the following pullback diagram



(where c and d are injective) is also a pushout (where a and b are also injective). These diagrams are normally called **Doolittle** diagrams.

Proposition A.3.17 *Pullbacks can be constructed from products and equalizers.*

A.4 General limit and colimit constructions

Definition A.4.1 *Let* **C** *be a category and* **D** *a diagram in* **C**. *A* **cone** *for* **D** *is a* **C***-object* X *and morphisms* $f_i : X \to D_i$ *(one for each* D_i *in* **D***), such that for each* **C***-morphism* $g : D_i \to D_j$ *in* **D**, *the diagram*

$(700,700)(-1,-1)[X\text{`}\quad\text{`}f_i]6001l(700,700)(1,-1)[\quad\text{`}\quad\text{`}f_j]6001r(100,100)(1,0)[D_i\text{`}D_j\text{`}g]12001b$

commutes. We use the notation $\{f_i : X \to D_i\}$ for cones.

Definition A.4.2 *A* **limit** *for a diagram* **D** *is a cone* $\{f_i : X \to D_i\}$ *with the property that if* $\{f'_i : X \to D_i\}$ *is another cone for* **D**, *then there is a unique morphism* $k : X' \to X$ *such that the diagram*

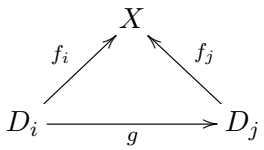
$(0,700)(1,0)[X'\text{`}X\text{`}k]12001a(0,700)(1,-1)[\quad\text{`}\quad\text{`}f'_i]6001l(1200,700)(-1,-1)[\quad\text{`}D_i\text{`}f_i]6001r$

commutes for every D_i *in* **D**.

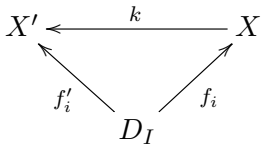
Proposition A.4.3 *Limits are unique up to isomorphism.*

Example A.4.4 Products, equalizers and pullbacks are all special cases of limit constructions.

Definition A.4.5 *A* **cocone** *for a diagram* **D** *in a category* **C** *is a* **C***-object* X *and a collection of* **C***-morphisms* $f_i : D_i \to X$ *(one for each* D_i *in* **D***) such that the following diagram commutes:*



Definition A.4.6 *A* **colimit** *or* **inverse limit** *for* **D** *is then a cocone* $\{f_i : D_i \to X\}$ *with the couniversal property that for any other cocone* $\{f'_i : D_i \to X'\}$ *there is a unique morphism* $k : X \to X'$ *such that the diagram*



commutes for every D_i *in* **D**.

Proposition A.4.7 *Colimits are unique up to isomorphism.*

Example A.4.8 Coproducts, coequalizers and pushouts are all special cases of colimit constructions.

References