Electronic Communications of the EASST
Volume 39 (2011)

Graph Computation Models
Selected Revised Papers from the
Third International Workshop on
Graph Computation Models (GCM 2010)

Construction of Pushout Complements in the Category of Hypergraphs

Marvin Heumüller, Salil Joshi, Barbara König, Jan Stückrath

20 pages

Guest Editors: Rachid Echahed, Annegret Habel, Mohamed Mosbah
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Construction of Pushout Complements in the Category of Hypergraphs

Marvin Heumüller[1], **Salil Joshi**[2], **Barbara König**[1], **Jan Stückrath**[1]

[1] Abteilung für Informatik und Angewandte Kognitionswissenschaft, Universität Duisburg-Essen, Germany

[2] Carnegie Mellon University, USA

Abstract: We describe a concrete construction of all pushout complements for two given morphisms $f\colon A \to B$, $m\colon B \to D$ in the category of hypergraphs, valid also for the case where f, m are non-injective. It is based on the generation of suitable equivalence relations. We also give a combinatorial interpretation and show how well-known coefficients from combinatorics, such as the Bell numbers, can be recovered. Furthermore we present a formula that can be used to compute the number of pushout complements for two given morphisms.[1]

Keywords: graph transformation, pushout complements, combinatorics

1 Introduction

Pushout complements are an integral part of double-pushout rewriting [2, 4, 5]: they implement the deletion of elements, whereas the creation of new elements is implemented via a pushout. Hence the construction of pushout complements is needed for many tools based on double-pushout graph rewriting. Most of the time the left leg of a rule is considered to be injective and thus the construction of pushout complements is greatly simplified compared to the general case, where both morphisms might be non-injective. A thorough study of the expressiveness of injective and non-injective rules and matches can be found in [6].

In [7] we considered a backwards analysis technique for graph transformation systems where rewriting steps have to be applied backwards. That is we are interested in *all* predecessors of a given graph, which is a common scenario in verification techniques. In this setting pushout complements have to be constructed for the right leg of a rule and in many applications this morphism is *not* injective, especially in cases where graph nodes and edges are fused by rewriting. In [7] we considered in fact single-pushout rewriting [3] with pushouts in the category of partial morphisms. The problem of computing such pushout complements can be reduced to the construction of pushout complements for total morphisms, hence the construction given in this paper can also be adapted to the scenario in [7].

Taking pushout complements for morphisms which are non-injective means—intuitively—to "unmerge" or split nodes in all possible ways, which can lead to a combinatorial explosion and serious efficiency problems.

[1] Research supported by the DFG project GaReV.

In the literature the general case has so far received little attention. In the 70s the papers introducing and studying the notion of pushout complement [4, 5, 9] restricted themselves to cases where either a vertical or a horizontal morphism is injective. Furthermore there are some investigations into taking pushout complements in more general categories [1, 8], but they usually assume that the first morphism is a mono or consider only the minimal pushout complement. Since a construction of general pushout complements does not seem to be available in the literature, we specified this construction ourselves and found it surprisingly complex. Hence we believe that it is of general interest.

We will in the following define the construction which computes all pushout complements for two given morphisms $f\colon A \to B$, $m\colon B \to D$. This is done by defining an auxiliary graph $A \oplus \tilde{D}$ which is the disjoint union of A and a disjoint collection of all nodes and edges of D, which are not in the image of m. Then we enumerate all equivalences on $A \oplus \tilde{D}$ satisfying certain conditions and factor through these equivalences. In this way we obtain all pushout complements and our main theorem proves this fact. Furthermore—since the enumeration of all equivalences on $A \oplus \tilde{D}$ is very costly and there are serious issues with efficiency—we consider optimizations. We show how some coefficients from combinatorics, such as Bell numbers or Stirling numbers of the second kind arise as the number of pushout complements for certain pairs of arrows. This also shows that there can be a combinatorial explosion in the number of constructed pushout complements. Finally we present a general formula that can be used to compute the number of pushout complements for two given morphisms.

2 Preliminaries

We first define the usual notions of hypergraph and hypergraph morphism.

Definition 1 (Hypergraph) Let Λ be a finite set of labels and a function $ar\colon \Lambda \to \mathbb{N}_0$ that assigns an *arity* to each label.

A *(Λ-)hypergraph* is a tuple (V_G, E_G, c_G, l_G) where V_G is a finite set of nodes, E_G is a finite set of edges, $c_G\colon E_G \to V_G^*$ is a connection function and $l_G\colon E_G \to \Lambda$ is the labelling function for edges. We require that $|c_G(e)| = ar(l_G(e))$ for each edge $e \in E_G$.

Definition 2 (Hypergraph morphism) Let G, G' be (Λ-)hypergraphs. A *hypergraph morphism* (or simply *morphism*) $\varphi\colon G \to G'$ consists of a pair of functions $(\varphi_V : V_G \to V_{G'}, \varphi_E : E_G \to E_{G'})$ such that for every $e \in E_G$ it holds that $l_G(e) = l_{G'}(\varphi_E(e))$ and $\varphi_V(c_G(e)) = c_{G'}(\varphi_E(e))$.

In the following, we will simply use *graph* to denote a hypergraph.

We will work extensively with equivalence relations and one required operation is equivalence closure that turns an arbitrary relation into an equivalence.

Definition 3 (Equivalence closure) Let A be a set and $\mathcal{R}$ be a relation $\mathcal{R} \subseteq A \times A$. The equivalence closure $\overline{\mathcal{R}}$ of $\mathcal{R}$ is the smallest equivalence containing $\mathcal{R}$.

In the following, equivalence closure is mainly used if $\mathcal{R}$ is the union of two equivalences $\equiv_1, \equiv_2$ on A, i.e., $\mathcal{R} = \equiv_1 \cup \equiv_2$. In this case $\overline{\mathcal{R}}$ is simply the transitive closure of $\equiv_1 \cup \equiv_2$ and can be written as

$$\begin{aligned}\overline{\mathcal{R}} \;=\; & \{(x,y) \in A \times A \mid \exists x_1, y_1, \ldots, x_n, y_n \in A : \\ & \quad x = x_1 \equiv_1 y_1 \equiv_2 x_2 \equiv_1 \cdots \equiv_1 y_{n-1} \equiv_2 x_n \equiv_1 y_n = y\}\end{aligned}$$

Definition 4 (Pushout) Let A, B, C be graphs with graph morphisms $f\colon A \to B$ and $n\colon A \to C$.

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "n"'] & B \arrow[d, "m"'] \arrow[rdd, bend left, "\overline{m}"] & \\
C \arrow[r, "g"] \arrow[rrd, bend right, "\overline{g}"'] & D \arrow[rd, dotted, "h"] & \\
 & & D'
\end{tikzcd}$$

The graph D together with $g\colon C \to D$ and $m\colon B \to D$ is a pushout of f, n if the following conditions are satisfied:

(1) $m \circ f = g \circ n$.

(2) For all $\overline{m}\colon B \to D'$, $\overline{g}\colon C \to D'$ satisfying $\overline{m} \circ f = \overline{g} \circ n$ there exists a unique morphism $h\colon D \to D'$ such that $h \circ m = \overline{m}$ and $h \circ g = \overline{g}$.

There is a well-known construction of pushouts [5] in the category of hypergraphs, where pushouts are obtained by taking the disjoint union of B and C and factoring through an equivalence obtained from the morphisms f, n.

Proposition 1 (Pushout via equivalence classes) *Let A, B, C be graphs with graph morphisms $f\colon A \to B$, $n\colon A \to C$. We also assume that all node and edge sets are disjoint.*[2]

Let $\equiv$ be the equivalence closure of the relation $\hat{=}$ on $V_B \cup E_B \cup V_C \cup E_C$ which is defined as $f(x) \hat{=} n(x)$ for all $x \in V_A \cup E_A$. The gluing of B, C over A (written as $D = (B \oplus C)/\equiv$) is defined as $D = (V_D, E_D, c_D, l_D)$ with:

- $V_D = (V_B \cup V_C)/\equiv$,
- $E_D = (E_B \cup E_C)/\equiv$,
- $c_D\colon E \to V^*$ *where* $c_D([e]_\equiv) = [v_1]_\equiv \ldots [v_k]_\equiv$ *and* $v_1 \ldots v_k = \begin{cases} c_B(e) & \textit{if } e \in E_B \\ c_C(e) & \textit{if } e \in E_C \end{cases}$
- $l_D\colon E \to \Lambda$ *where* $l_D([e]_\equiv) = \begin{cases} l_B(e) & \textit{if } e \in E_B \\ l_C(e) & \textit{if } e \in E_C \end{cases}$

The resulting morphisms are $m\colon B \to D$, $g\colon C \to D$ with:

$$g(x) = [x]_\equiv \qquad\qquad m(x) = [x]_\equiv$$

Then D together with the morphisms g, m is the pushout of f, n.

[2] Disjointness can be achieved easily by renaming.

Note that the functions c_D and l_D in Proposition 1 are well-defined due to the morphism properties of Definition 2.

Definition 5 (Pushout complement) Given morphisms $f\colon A \to B$, $m\colon B \to D$ a *pushout complement* of f,m is a graph C and a pair of morphisms $n\colon A \to C$, $g\colon C \to D$ such that g,m form the pushout of f,n. We say that two pushout complements C_i with $n_i\colon A \to C_i$, $g_i\colon C_i \to D$ for $i = 1,2$ are *isomorphic* if there exists an isomorphism $j\colon C_1 \to C_2$ with $j \circ n_1 = n_2$ and $g_2 \circ j = g_1$.

There is a well-known characterization of the existence of pushout complements (see for instance Proposition 3.3.4 of [2]).

Proposition 2 (Existence of pushout complements) *A pushout complement of f,m exists if and only if the following two conditions are satisfied:*

- Identification condition: *for all $x,y \in V_B \cup E_B$ with $m(x) = m(y)$ there exist $x',y' \in V_A \cup E_A$ with $f(x') = x$, $f(y') = y$.*
- Dangling condition: *for every node $v \in V_B$ where $m(v)$ is attached to an edge $e \in E_D$ which is not in the range of m, there exists a node $v' \in V_A$ with $f(v') = v$.*

3 Construction of pushout complements

In this section we will give a concrete construction for pushout complements, i.e., given morphisms $f\colon A \to B$ and $m\colon B \to D$, we construct *all* pairs of morphisms $n\colon A \to C$, $g\colon C \to D$ and graphs C (up to isomorphism) such that the resulting square is a pushout.

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, dotted, "n"'] & B \arrow[d, "m"] \\ C \arrow[r, dotted, "g"] & D \end{tikzcd}$$

We use the following abbreviations: since it is often not necessary to distinguish between edges and nodes of a graph, we will use $x \in A$ as shorthand for ($x \in E_A$ or $x \in V_A$) and $f(x)$ as shorthand for $f_V(x)$ if $x \in V_A$ and $f_E(x)$ if $x \in E_A$ respectively.

Construction 1 (Pushout complements)

(1) Construct a graph $\tilde{D}$ as follows:

- For every node $v \in V_D$ that is not in the range of m, add a copy of v to $\tilde{D}$. The copy of v will be denoted by v'.
- For every edge $e \in E_D$ that is not in the range of m, add a copy of e with the same arity, attached to fresh nodes, to $\tilde{D}$. (This is done also if some of the nodes attached to e are in the range of m.) The copy of e will be denoted by e' and the fresh nodes by (e',i), $i \in \{1,\ldots,ar(l_D(e))\}$.

This means that $\tilde{D}$ is a collection of disjoint nodes and edges.

(2) Now construct $A \oplus \tilde{D}$, the disjoint union of A and $\tilde{D}$, with morphisms $n' \colon A \to A \oplus \tilde{D}$, $g' \colon A \oplus \tilde{D} \to D$ as follows:

- n' is the canonical embedding of A into $A \oplus \tilde{D}$.
- For an item x of $A \oplus \tilde{D}$ we define $g'(x) = m(f(x))$ if x is contained in A. If $x = y'$ for some item y of D we define $g'(x) = y$. Finally if $x = (e', i)$ for some edge e of D we have $g'((e', i)) = [c_D(e)]_i$.[3] (See Step (1) of this construction where items of the form y' were created.)

Clearly $g' \circ n' = m \circ f$.

(3) Define two equivalences on the items of $A \oplus \tilde{D}$:

- $x \equiv_{g'} y$ if and only if $g'(x) = g'(y)$.
- $x \equiv_f y$ if either $x = y$ or x, y are both items of A and $f(x) = f(y)$.

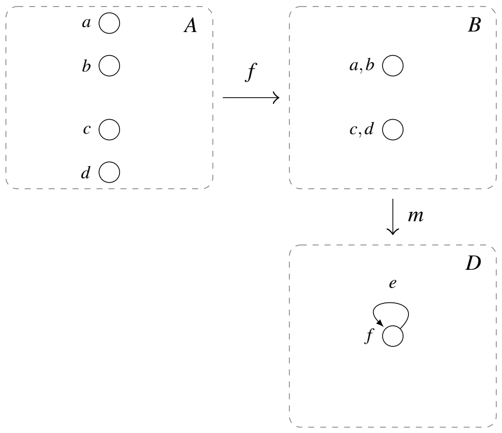
It can easily be seen that $\equiv_f$ is a refinement of $\equiv_{g'}$, i.e., $x \equiv_f y$ implies $x \equiv_{g'} y$.

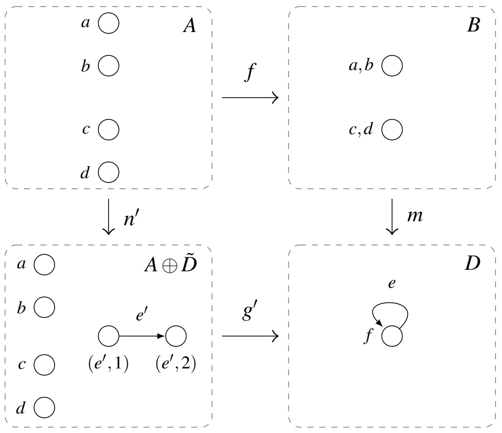
(4) Now consider all equivalences $\equiv$ on $A \oplus \tilde{D}$ such that $\equiv_{g'}$ is the equivalence closure of $\equiv_f \cup \equiv$. Furthermore whenever $e_1 \equiv e_2$ for two edges e_1, e_2, we require that $[c_{A\oplus\tilde{D}}(e_1)]_i \equiv [c_{A\oplus\tilde{D}}(e_2)]_i$ for all $1 \le i \le ar(l_G(e_1)) = ar(l_G(e_2))$. For each such equivalence $\equiv$ construct the graph $C = (A \oplus \tilde{D})/\equiv$ with morphisms $n \colon A \to C$, $g \colon C \to D$ as specified below:

$$n(x) = [n'(x)]_\equiv \qquad\qquad g([x]_\equiv) = g'(x)$$

Note that g is well-defined since $\equiv$ refines $\equiv_{g'}$.

Example 1 *Consider for instance the situation on the left below. We have a single binary edge, which is unlabeled (labels do not play a role for this example). The identities of nodes and edges are illustrated by the letters next to them.*





[3] For a sequence s we denote by $[s]_i$ the i-th element of s.

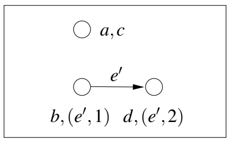
On nodes we have the equivalences $\equiv_{g'}$, $\equiv_f$, represented by their equivalence classes:

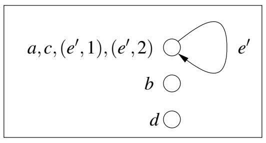
- $\equiv_{g'}$: $\{a,b,c,d,(e',1),(e',2)\}$
- $\equiv_f$: $\{a,b\}$, $\{c,d\}$, $\{(e',1)\}$, $\{(e',2)\}$

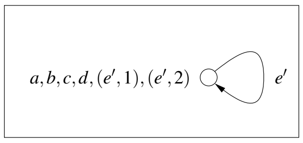
Now there are many possible equivalences $\equiv$. First, we have to relate at least one node from $\{a,b\}$ to one node from $\{c,d\}$. Furthermore we have to relate each of the two nodes $(e',1)$, $(e',2)$ to an equivalences class containing one of a,b,c,d. For instance the following three equivalences $\equiv$ are all permissible:

- $\{a,c\},\{b,(e',1)\},\{d,(e',2),\}$
- $\{a,c,(e',1),(e',2)\},\{b\},\{d\}$
- $\{a,b,c,d,(e',1),(e',2)\}$

This results in the following three graphs:







But there are many more possibilities. In order to enumerate them more systematically we consider all 15 equivalences on the set $\{a,b,c,d\}$, given by equivalence classes (a better way of enumeration is describes in Section 4.2). The ones that do not satisfy the requirement above are crossed out.

$\{a,b,c,d\}$ $\{a\},\{b,c,d\}$ $\{b\},\{a,c,d\}$ $\{c\},\{a,b,d\}$ $\{d\},\{a,b,c\}$
~~$\{a,b\},\{c,d\}$~~ $\{a,c\},\{b,d\}$ $\{a,d\},\{b,c\}$ ~~$\{a,b\},\{c\},\{d\}$~~ $\{a,c\},\{b\},\{d\}$
$\{a,d\},\{b\},\{c\}$ $\{b,c\},\{a\},\{d\}$ $\{b,d\},\{a\},\{c\}$ ~~$\{c,d\},\{a\},\{b\}$~~
~~$\{a\},\{b\},\{c\},\{d\}$~~

Now for k equivalence classes there are k^2 possibilities to associate $(e',1)$ and $(e',2)$ to these equivalence classes. Hence in total there are $1+6\cdot 2^2+4\cdot 3^2=61$ equivalences. Some of them result in isomorphic graphs, however they are all non-isomorphic in the sense of Definition 5 (see also Proposition 4). How the equivalences can be counted in the arbitrary case is further discussed in Section 5.

We now show that every graph C constructed as specified in Construction 1 is a pushout complement and that all pushout complements can be obtained in this way.

Proposition 3 *Assume that $f\colon A\to B$, $m\colon B\to D$ are given and that the conditions of Proposition 2 are satisfied. Then every equivalence relation $\equiv$ created by Construction 1 generates a pushout complement.*

Proof. Assume that $\equiv$ is one of the equivalences of Construction 1 and that C and n,g have been obtained by factoring $A \oplus \tilde{D}$ through this equivalence.

As a first step we show that $m \circ f = g \circ n$, i.e., the resulting square commutes: because n' is the canonical embedding of A into $A \oplus \tilde{D}$ (and therefore injective) and $g'(x)$ is defined as $m(f(x))$ if $x \in A$, $m(f(x)) = g'(n'(x))$ holds. Furthermore by definition of n,g we have:

$$m(f(x)) = g'(n'(x)) = g([n'(x)]_{\equiv}) = g(n(x))$$

Now we show that C is indeed a pushout complement by verifying that the second condition of Definition 4 are satisfied: we have to prove that for every other commuting pair of morphisms $\overline{g}\colon C \to D'$, $\overline{m}\colon B \to D'$ there is a unique morphism $h\colon D \to D'$ such that $h \circ g = \overline{g}$ and $h \circ m = \overline{m}$.

$$\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "n"'] & B \arrow[d, "m"'] \arrow[ddr, bend left, "\overline{m}"] & \\
C \arrow[r, "g"] \arrow[drr, bend right, "\overline{g}"'] & D \arrow[dr, dotted, "h"] & \\
& & D'
\end{tikzcd}$$

We define the required morphism h as follows:

$$h(x) = \begin{cases} \overline{g}(\tilde{x}) & \text{if } \exists \tilde{x} \in C : g(\tilde{x}) = x \\ \overline{m}(\tilde{x}) & \text{if } \exists \tilde{x} \in B : m(\tilde{x}) = x \end{cases}$$

By definition of g every element of D has a preimage either under g or m. It remains to be shown that h is a well-defined morphism, and that it is the unique morphism such that the triangles commute.

Commutativity. By definition $h(m(x)) = \overline{m}(x)$ and $h(g(x)) = \overline{g}(x)$ hold.

Uniqueness. Let h' be another morphism with $h' \circ g = \overline{g}$ and $h' \circ m = \overline{m}$. Each element of D has a preimage either under g or m:

(1) if $x = g(x')$ then $h'(x) = h'(g(x')) = \overline{g}(x') = h(g(x')) = h(x)$

(2) if $x = m(x')$ then $h'(x) = h'(m(x')) = \overline{m}(x') = h(m(x')) = h(x)$

Well-definedness. As seen before h is defined for all elements of D. To show well-definedness it is therefore only necessary to prove that different $\tilde{x}$ having the same image under g or m also have the same image under $\overline{g}$ or $\overline{m}$.

Every element of C is an equivalence class of $\equiv$. Therefore, let $x = [x']_{\equiv}$ and $y = [y']_{\equiv}$. In the following we do not strictly distinguish between an element of A and its image under n' because n' is a canonical embedding. Hence for $x' \in A \oplus \tilde{D}$ the property $x' \in A$ holds if and only if x' has a preimage under n'.

The *first property* we show is that $g(x) = g(y) \Rightarrow \overline{g}(x) = \overline{g}(y)$ holds for all $x, y \in C$. For $x \neq y$ there are two cases which have to be considered:

(1) $x', y' \in A$, i.e., we assume that the equivalence classes x, y have representatives in A (which also implies $n(x') = x$ and $n(y') = y$).

$x' \equiv_{g'} y'$ holds because of $g'(x') = g([x']_\equiv) = g(x) = g(y) = g([y']_\equiv) = g'(y')$. Due to this equivalence there are $x_1, y_1, \ldots, x_n, y_n \in A$ such that $x' \equiv_f x_1$, $x_i \equiv y_i$, $y_i \equiv_f x_{i+1}$ and $y_n \equiv_f y'$ for $1 \leq i < n$. Using the definition of n and the fact that x_i and y_i are elements of A it can be shown that the equivalence $x_i \equiv y_i$ implies $n(x_i) = [n'(x_i)]_\equiv = [n'(y_i)]_\equiv = n(y_i)$. These properties lead to the following equality

$$\overline{m}(f(x_i)) = \overline{g}(n(x_i)) = \overline{g}(n(y_i)) = \overline{m}(f(y_i)) = \overline{m}(f(x_{i+1}))$$

for every i. Together with the equalities $\overline{g}(n(x')) = \overline{m}(f(x')) = \overline{m}(f(x_1))$ and $\overline{g}(n(y_n)) = \overline{m}(f(y_n)) = \overline{m}(f(y')) = \overline{g}(n(y'))$ it follows that $\overline{g}(x) = \overline{g}(y)$.

(2) x contains no elements of A (implying $x' \notin A$)
Because x contains no elements of A, it also has no preimage under n. As already shown $g([x']_\equiv) = g([y']_\equiv)$ implies $x' \equiv_{g'} y'$. Because of this equivalence there are $x_1, y_1, \ldots, x_n, y_n \in A$ satisfying $x' \equiv_f x_1$, $x_i \equiv y_i$, $y_i \equiv_f x_{i+1}$, $y_n \equiv_f y'$ for $1 \leq i < n$. Due to the definition of $\equiv_f$ it holds that $x' = x_1$ because x' is not in A. Also y_1 can not be an item of A because otherwise $[x']_\equiv$ would contain items of A. This property can be extended to $y_i = x_{i+1}$ and $y_n = y'$, which leads to $x_i \equiv x_{i+1}$. Because of $x' = x_1$ and $x_n \equiv y'$, x' and y' are equivalent according to $\equiv$ and hence x and y must be equal. This clearly implies $g(x) = g(y) \Rightarrow \overline{g}(x) = \overline{g}(y)$.

The *second property* needed for well-definedness is $m(x) = m(y) \Rightarrow \overline{m}(x) = \overline{m}(y)$. The identification condition (see Proposition 2) states that because of $m(x) = m(y)$ there are $x', y' \in A$ such that $f(x') = x$ and $f(y') = y$. Using this and the first property the desired equality can easily be shown by:

$$\begin{array}{ccccccc} m(x) = m(y) & \Rightarrow & m(f(x')) = m(f(y')) & \Rightarrow & g(n(x')) = g(n(y')) & \Rightarrow \\ \overline{g}(n(x')) = \overline{g}(n(y')) & \Rightarrow & \overline{m}(f(x')) = \overline{m}(f(y')) & \Rightarrow & \overline{m}(x) = \overline{m}(y) \end{array}$$

The *last property* to show is $g(x) = m(y) \Rightarrow \overline{g}(x) = \overline{m}(y)$. We first show that $g(x) = m(y)$ implies that there is a y' with $f(y') = y$: the only items of D which are in the range of both g and m are the images of elements of A and nodes in the range of m which are attached to edges which are not in the range of m. However, due to the dangling condition (see Proposition 2) such nodes must have a preimage in A. Together with the first property this implies:

$$\begin{array}{ccccccc} g(x) = m(y) & \Rightarrow & g(x) = m(f(y')) & \Rightarrow & g(x) = g(n(y')) & \Rightarrow \\ \overline{g}(x) = \overline{g}(n(y')) & \Rightarrow & \overline{g}(x) = \overline{m}(f(y')) & \Rightarrow & \overline{g}(x) = \overline{m}(y) \end{array}$$

Morphism. Finally it is straightforward to prove that h satisfies indeed the morphism properties. For instance in order to show that $h(c_D(e)) = c_{D'}(h(e))$ for an edge $e \in D$ we have to distinguish two cases: if there exists an edge $\tilde{e} \in C$ with $g(\tilde{e}) = e$, then—since g is a morphism—we have $g(c_C(\tilde{e})) = c_D(e)$. Hence $h(c_D(e)) = h(g(c_C(\tilde{e}))) = \overline{g}(c_C(\tilde{e})) = c_{D'}(\overline{g}(\tilde{e})) = c_{D'}(h(e))$ by definition of h. The case $\tilde{e} \in B$ with $m(\tilde{e}) = e$ is analogous.

This proves that every diagram formed by an equivalence generated in the given construction is a pushout diagram. □

Proposition 4 *Assume that $f\colon A \to B$, $m\colon B \to D$ are given. Then every pushout complement $n\colon A \to C$, $g\colon C \to D$ of f,m can be obtained via Construction 1. Furthermore two isomorphic pushout complements give rise to the same equivalence $\equiv$.*

Proof. Assume that C with morphisms n,g is a pushout complement of f,m. We will show that there is an equivalence $\equiv$, as specified by Construction 1, such that C is obtained by factoring $A \oplus \tilde{D}$ through this equivalence.

$$\begin{tikzcd}
& A \arrow[r, "f"] \arrow[d, "n"'] \arrow[dl, bend right, "n'"'] & B \arrow[d, "m"] \\
& C \arrow[r, "g"] & D \\
A \oplus \tilde{D} \arrow[ur, dotted, "k"] \arrow[urr, bend right, "g'"'] & &
\end{tikzcd}$$

For the given pushout of f,n we will define a surjective morphism $k\colon A \oplus \tilde{D} \to C$ (see diagram above). Our next step is then to define an equivalence relation $\equiv$ where $x,y \in A \oplus \tilde{D}$ are equivalent if and only if $k(x) = k(y)$. The factorization of $A \oplus \tilde{D}$ through $\equiv$ then results in C and it has to be shown that the equivalence relation $\equiv$ is one of the equivalence relations obtained by the presented construction.

Let $\equiv^*$ be the equivalence closure of the relation $\hat{\equiv}$ where $f(a) \hat{\equiv} n(a)$ for all $a \in A$. Due to the construction of pushouts using equivalence classes we can assume without loss of generality that $D = (B \oplus C)/\equiv^*$ (see Proposition 1). Furthermore for $b \in B$ we have $m(b) = [b]_{\equiv^*}$ and for $c \in C$ we have $g(c) = [c]_{\equiv^*}$.

We define k as follows: if $x \in A$, then $k(x) = n(x)$. If x is of the form y' for some item y of D, then — since y is not in the image of m — there must be a $c \in C$ with $g(c) = y$. In this case we define $k(x) = c$. If x is of the form (e',i) for some edge e of D, then $k(x) = [c_C(k(e))]_i$.

Well-definedness. Problems with well-definedness may arise only in the second case of the definition of k, where x is of the form y' for some item y of D. In this case y is not in the range of m due to the construction of $A \oplus \tilde{D}$. Therefore y as an equivalence class does not contain elements of B. Because of the definition of $\equiv^*$ every equivalence class containing elements of either B or C (but not both) only contains one element, hence y contains exactly one element c of C. Because $g(c) = [c]_{\equiv^*} = y$ the preimage of y under g is unique and therefore $k(x)$ is well-defined in this case.

Morphism. Note that k is obviously a morphism on the elements of A. Furthermore $\tilde{D}$ is a disjoint collection of nodes and edges and the third case in the definition of k ensures that it is indeed a valid morphism.

Surjectivity. We now show that k is surjective. Let therefore $c \in C$ be any element of C and we distinguish the following two cases:

(1) $\exists y \in A\colon n(y) = c$: By definition $k(y) = n(y) = c$.

(2) $\nexists y \in A \colon n(y) = c$: Without a preimage under n the equivalence class $[c]_{\equiv^*}$ contains only c because c is not equivalent to any element of B according to $\equiv^*$. Therefore $[c]_{\equiv^*}$ is not in the range of m since otherwise the equivalence class would contain elements of B. Because of the definition of k there is a $y' \in \tilde{D}$ with $g'(y') = y = [c]_{\equiv^*} = g(c)$, hence $k(x) = c$.

Commutativity. We have to show that both triangles commute:

(1) We first check that $k(n'(x)) = n(x)$ for any $x \in A$:

As already seen $n'(x) = x$ if $x \in A$. Using the definition of k we obtain $k(n'(x)) = k(x) = n(x)$.

(2) Now we show that $g(k(x)) = g'(x)$ for any $x \in A \oplus \tilde{D}$. There are two cases:

(a) $x \in A$: Using $k(x) = n(x)$ if $x \in A$ and $m \circ f = g' \circ n'$ due to the definition of g' and n' it can be shown that:
$g(k(x)) = g(n(x)) = m(f(x)) = g'(n'(x)) = g'(x)$

(b) $x \in \tilde{D}$: In this case $k(x) = c$ and $g(c) = g'(x)$, therefore $g(k(x)) = g(c) = g'(x)$.

The equivalence $\equiv$ is generated. We will now show that the equivalence $\equiv$, where two elements $x, y \in A \oplus \tilde{D}$ are equivalent if and only if $k(x) = k(y)$, is generated by the given construction. Hence we have to show that the equivalence closure of $\equiv \cup \equiv_f$ is $\equiv_{g'}$, i.e., that $\overline{\equiv \cup \equiv_f} = \equiv_{g'}$.

- $\overline{\equiv \cup \equiv_f} \subseteq \equiv_{g'}$:

  As already mentioned in Construction 1, $\equiv_f$ implies $\equiv_{g'}$, i.e. $\equiv_f$ is clearly a subset of $\equiv_{g'}$.

  The equivalence $\equiv$ is also a subset of $\equiv_{g'}$ because of:

$$x \equiv y \Rightarrow k(x) = k(y) \Rightarrow g'(x) = g(k(x)) = g(k(y)) = g'(y)$$

- $\overline{\equiv \cup \equiv_f} \supseteq \equiv_{g'}$:

  Let x, y be elements of $A \oplus \tilde{D}$ with $x \equiv_{g'} y$, hence $g'(x) = g'(y)$. As shown above the equivalence classes $g'(x)$ and $g'(y)$ of $\equiv^*$ contain $k(x)$ and $k(y)$ respectively, therefore $k(x) \equiv^* k(y)$. Hence there are $c_0, b_1, c_1, \dots b_m, c_m$ such that $b_i \stackrel{\sim}{\equiv} c_i$ for $1 \leq i \leq m$ and $b_{j+1} \stackrel{\sim}{\equiv} c_j$ for $0 \leq j < m$ with $k(x) = c_0$ and $k(y) = c_m$. Using the definition of $\stackrel{\sim}{\equiv}$ leads to the following properties:

$$\begin{array}{ccc} b_i \stackrel{\sim}{\equiv} c_i & \Rightarrow & \exists a_i \in A \colon f(a_i) = b_i \wedge n(a_i) = c_i \\ b_{i+1} \stackrel{\sim}{\equiv} c_i & \Rightarrow & \exists a'_i \in A \colon f(a'_i) = b_{i+1} \wedge n(a'_i) = c_i \end{array}$$

  This implies that a_{i+1} and a'_i have the same image under f, hence $a_{i+1} \equiv_f a'_i$, and that a_i and a'_i have the same image under n, hence $a_i \equiv a'_i$. This leads to $x \equiv a'_0 \equiv_f a_1 \equiv a'_1 \equiv_f \cdots \equiv a'_{m-1} \equiv_f a_m \equiv y$, hence $x \, \overline{\equiv \cup \equiv_f} \, y$.

This proves that every pushout complement can be obtained by using the given construction.

Isomorphism of pushout complements. It is left to show that, given two isomorphic pushout complements $n_i\colon A \to C_i$, $g_i\colon C_i \to D$ with $i = 1,2$ and an isomorphism $j\colon C_1 \to C_2$ with $j \circ n_1 = n_2$, $g_2 \circ j = g_1$, the corresponding equivalences $\equiv$ are the same. For this it is sufficient to show that j commutes with the morphisms k_1,k_2, where $k_i\colon A \oplus \tilde{D} \to C_i$ and k_1,k_2 are constructed analogously to the morphism k above. That is, we have to show that $j \circ k_1 = k_2$. Then k_1,k_2 give rise to the same equivalence $\equiv$.

We distinguish the following cases (as in the definition of k): if $x \in A$, then $j(k_1(x)) = j(n_1(x)) = n_2(x) = k_2(x)$. If x is of the form y' for some item y of D, then we define $k_i(x) = c_i$ for c_i with $g_i(c_i) = y$. Since $g_2(j(c_1)) = g_1(c_1) = y$ we obtain $c_2 = j(c_1)$. Hence $j(k_1(x)) = j(c_1) = c_2 = k_2(x)$. Finally, if x is of the form (e',ℓ) for some edge e of D, then $k_i(x) = [c_C(k_i(e))]_\ell$ and so $j(k_1(x)) = j([c_C(k_1(e))]_\ell) = [c_C(j(k_1(e)))]_\ell = [c_C(k_2(e))]_\ell = k_2(x)$. This completes the proof. □

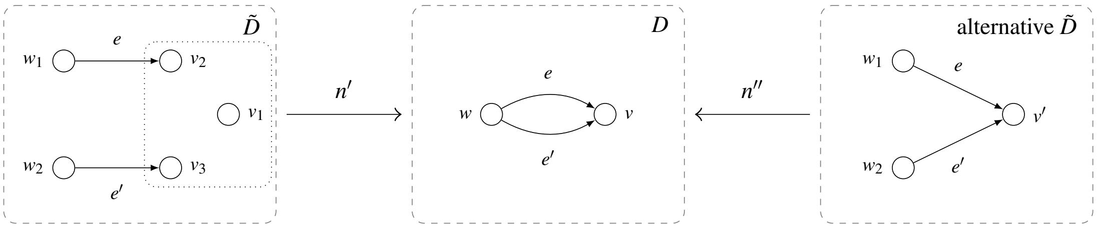
The fact that two isomorphic pushout complements give rise to the same equivalence means that the number of generated (valid) equivalences is exactly the number of different pushout complements. However, if we consider only isomorphisms on C—without requiring commutativity of the triangles consisting of morphisms j,n_1,n_2 and j,g_1,g_2 (in the terminology of Definition 5)—there will usually be fewer different pushout complements. The examples in Sections 5.1 and 5.2 are chosen in such a way that both interpretations give rise to the same number.

4 Optimizations

In the given construction there exist several possibilities for optimization. These lie in the construction of $A \oplus \tilde{D}$ and in the method used to enumerate all possible equivalences $\equiv$.

4.1 Possible Simplifications

In Step (1) of Construction 1 the graph $\tilde{D}$ is constructed by inserting all nodes and edges of D which are not in the range of m. Additionally for every edge e of D for every node connected with e a new node is inserted. This ensures that every node attached to e is also in $\tilde{D}$. However, if e is connected to a node x not in the range of m, another copy of this node has been added earlier to $\tilde{D}$. Both are equivalent with respect to $\equiv_{g'}$ but not with respect to $\equiv_f$ since they do not have a preimage under n'. Therefore these two copies have to be equivalent according to every possible equivalence $\equiv$. Hence the first copy was superfluous and it was unnecessary to create it in the first place.



The previous diagram shows an example graph $\tilde{D}$ generated by the given construction if the middle graph is D and only w is in the range of m. In the left graph v_1, v_2 and v_3 are all copies of

v in the middle graph and all have to be in the same $\equiv$-class. The construction would therefore still be correct if the right graph is generated instead of the left graph.

In general it is only necessary to add one node to $A \oplus \tilde{D}$ for every node not in the range of m and for every node in the range of m as many nodes as there are edges not in the range of m connected with the node (it depends on f whether the latter copies are really needed). This improvement can help to manage the combinatorial explosion when determining all possible equivalences $\equiv$.

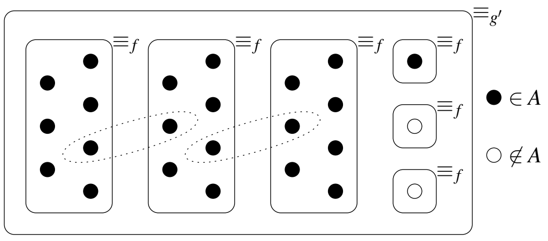
4.2 Enumerating Equivalences

A problem not addressed earlier is how to generate all permissible equivalences $\equiv$. The straightforward way would be to enumerate all possible equivalences over $A \oplus \tilde{D}$ and to store every equivalence satisfying $\overline{\equiv \cup \equiv_f} = \equiv_{g'}$. This method is however not recommended because of combinatorial explosion and in addition many of these equivalences will not satisfy the required conditions. In the following we explain how the generation of equivalences could be handled more efficiently.

If f is injective there is only one permissible equivalence $\equiv$. This is true since in this case g must necessarily also be injective and hence $\equiv$ equals $\equiv_{g'}$ (also $x \equiv_f y \Leftrightarrow x = y$ holds). It is already known that in this case the pushout complement is unique, if it exists [5].

A non-injective morphism f produces several permissible equivalences $\equiv$. In this case it is sufficient to look at each equivalence class of $\equiv_{g'}$ separately. We further distinguish between equivalence classes which contain elements of A and those which do not. In either case every equivalence class of $\equiv_f$ is entirely contained in exactly one equivalence class of $\equiv_{g'}$ due to the definition of g'.

If an equivalence class c of $\equiv_{g'}$ contains no elements of A, every equivalence class of $\equiv_f$ contained in c only contains one element. Therefore c must also be an equivalence class of $\equiv$, i.e., all elements of c must be merged.



If an equivalence class c of $\equiv_{g'}$ contains elements of A, the equivalence classes of $\equiv_f$ in c contain either only elements of A or no elements of A (see figure above). Only equivalence classes of $\equiv_f$ containing elements of A can consist of more than one element. Elements already equivalent according to $\equiv_f$ do not have to be equated via $\equiv$ because they will anyway be equivalent after the equivalence closure. It is however necessary to add relations between elements in such a way that the resulting structure connects all equivalence classes to each other, possibly indirectly. (One such possibility connecting the three leftmost equivalence classes is indicated by the dashed ovals in the figure above.) Therefore, in order to calculate all permissible equivalences $\equiv$ for all elements of c, we first enumerate all equivalences over elements contained in equivalence classes of $\equiv_f$ with at least two elements, but keep only those that induce connectivity. We then distribute the remaining elements (contained in equivalence classes of $\equiv_f$ with only one element) to the resulting equivalence classes in every possible way. The results are all equivalences $\equiv$ restricted to elements of c. If we perform these steps for all other equivalence classes of $\equiv_{g'}$, a complete equivalence $\equiv$ can be obtained by taking arbitrary combinations of such (restricted) equivalences $\equiv$ for each class c.

5 Combinatorial Interpretation

Some coefficients from combinatorics arise naturally as the number of pushout complements for a (parameterized) pair of arrows. We first present some examples, all of them for hypergraphs with unary edges only, and later give a formula for calculating the number of pushout complements arising from given graphs and morphisms.

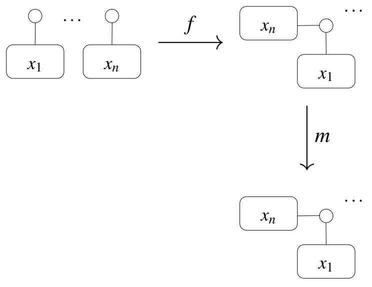
5.1 Bell Numbers

The n-th Bell number B_n is the number of equivalence relations on the set $\{1,\ldots,n\}$. The first Bell numbers (starting with B_1) are: 1, 2, 5, 15, 52, 203, 877, 4140, ... (see the On-Line Encyclopedia of Integer Sequences[4]).

Now take $\Lambda_x = \{x_1,\ldots,x_n\}$ as a label set. Assume that X_{Λ_x} is the graph with n nodes, where to each node we attach a unary hyperedge and each hyperedge has a different label. Furthermore Z_{Λ_x} is the graph with one node to which n hyperedges are attached, where each hyperedge has a different label.

We consider the unique morphism $f\colon X_{\Lambda_x} \to Z_{\Lambda_x}$ and the identity $m = id_{Z_{\Lambda_x}}\colon Z_{\Lambda_x} \to Z_{\Lambda_x}$. Then—if we apply our construction—the graph $A \oplus \tilde{D}$ will consist only of $A = X_{\Lambda_x}$ and *all* equivalences $\equiv$ on the nodes of X_{Λ_x} are admissible (for the edges each edge must be in a separate equivalence class). Hence there are B_n different pushout complements up to isomorphism.

$$\begin{tikzcd} X_{\Lambda_x} \arrow[r, "f"] & Z_{\Lambda_x} \arrow[d, "m"] \\ & Z_{\Lambda_x} \end{tikzcd}$$



5.2 Stirling Numbers of the Second Kind

The Stirling number of the second kind $S_{n,k}$ is the number of equivalence relations with k equivalence classes on the set $\{1,\ldots,n\}$. It holds that $B_n = \sum_{k=1}^{n} S_{n,k}$.

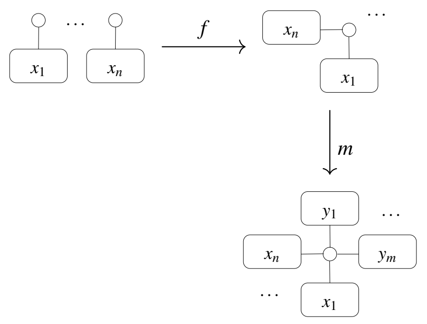
The Stirling numbers satisfy the following recursive equation: $S_{n,k} = S_{n-1,k-1} + k \cdot S_{n-1,k}$, which is based on a case distinction according to the element n: either n is in an

[4] http://www.research.att.com/~njas/sequences/

equivalence class of its own and the remaining $n-1$ elements have to be grouped in $k-1$ equivalence classes; or the remaining $n-1$ elements have to be grouped in k equivalence classes and there are k possibilities to assign n to one of these classes. Our implemented method for enumerating equivalences follows the same pattern.

Now we set $\Lambda_x = \{x_1,\dots,x_n\}$, $\Lambda_y = \{y_1,\dots,y_m\}$ and $\Lambda = \Lambda_x \cup \Lambda_y$. We take the unique morphism $f\colon X_{\Lambda_x} \to Z_{\Lambda_x}$ and the unique morphism $m\colon Z_{\Lambda_x} \to Z_\Lambda$.

$$\begin{tikzcd} X_{\Lambda_x} \arrow[r, "f"] & Z_{\Lambda_x} \arrow[d, "m"] \\ & Z_\Lambda \end{tikzcd}$$



Then $A \oplus \tilde{D}$ is the disjoint union of X_{Λ_x} and separate copies of m edges which are labelled $y_1,\dots,y_m$. Now we take all permissible equivalences on the nodes of the copy of X_{Λ_x}. Assume that we have k equivalence classes. Then there are k^m possibilities to distribute the m nodes of the separate edges over the equivalence classes. Hence the total number of pushout complements is

$$\sum_{k=1}^{n} S_{n,k} \cdot k^m$$

Note that for the special case of $m = 0$ we obtain again the Bell numbers.

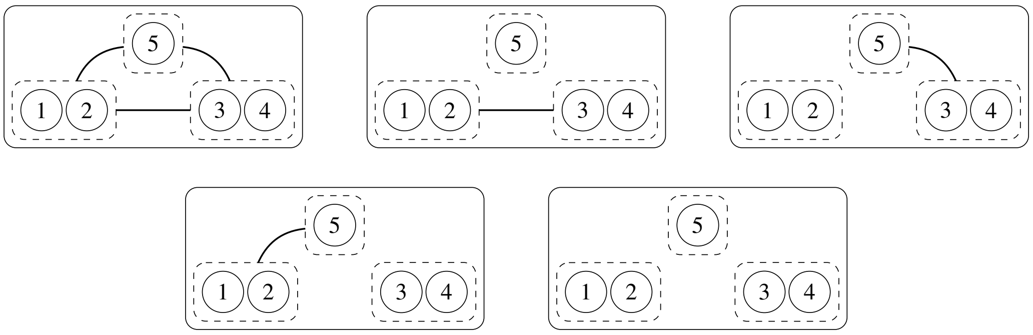
5.3 Counting pushout complements with only node fusions

With the examples above we have shown the worst cases for the number of pushout complements by looking at only one equivalence class of $\equiv_{g'}$. We now present a more accurate formula for calculating this number, still looking at the classes of $\equiv_{g'}$ individually. We will give this formula for the general case – independent of our construction – where an equivalence on an arbitrary set is given and a second, coarser, equivalence is searched for. We will in the following exploit the fact that the equivalences on a given set X coincide with the partitions on a set and we will switch between both representations.

Let $P_{all}(X) \subset \mathcal{P}(\mathcal{P}(X))$ be the set of all partitions of the given set X. We also define $Cnt(Y,Z)$ as the number of partitions on a set Z, such that the closure of the union of that partition and Y, a partition on Z, results in a partition where everything is in the same equivalence class (see also Section 4.2, the notion of closure is here straightforwardly extended to partitions). If Y contains only one element (i.e. Z), any partition on Z will result in $\{Z\}$ when forming the closure with Y, hence $Cnt(Y,Z) = B_{|Z|}$. The Bell Number $B_{|Z|}$ is thereby also a natural upper bound for $Cnt(Y,Z)$ regardless of the cardinality of Y. We can group these partitions according to which elements of Y they connect, i.e. which partition they induce on Y. For any induced partition I on Y we can calculate the number of partitions on Z inducing I by calculating the possibilities for each element of I individually and forming the product of the results. These possibilities can in turn be calculated by Cnt. For any partition Y of a set Z we obtain the following formula:

$$B_{|Z|} = \sum_{X \in P_{all}(Y)} \prod_{N \in X} Cnt\left(N, \bigcup_{M \in N} M\right)$$

Example 2 *Let $Y = \{\{1,2\},\{3,4\},\{5\}\}$ be a partition on $Z = \{1,2,3,4,5\}$. All partitions on Z will induce one of the partitions (or equivalences) on Y displayed below. For instance the top right case (where $\{3,4\}$ and $\{5\}$ are connected) represents six equivalences: 5 can either be equivalent to 3, to 4 or to both. In all three cases 1 and 2 can be equivalent or not, but can not be equivalent to 3, 4 or 5, because this would induce another equivalence. In the end, we want to count the number of equivalences inducing the top left partition by counting the rest and subtracting it from the number of all equivalences (given by the Bell numbers).*



If $Cnt(Y,Z)$ is the number of partitions to be counted, we obtain the following proposition by reorganizing the formula above to obtain a recursive formula. We obtain $Cnt(Y,Z)$ by subtracting the partitions not resulting in $\{Z\}$ when constructing the closure from the upper bound $B_{|Z|}$.

Proposition 5 *For any set Z and any partition Y on Z the number of partitions on Z which result in $\{Z\}$ when constructing the closure with Y, can be computed by the following recursive formula:*

$$Cnt(\{Z\},Z) = B_{|Z|}$$

$$Cnt(Y,Z) = B_{|Z|} - \sum_{X \in P_{all}(Y) \setminus \{\{Y\}\}} \prod_{C \in X} Cnt\left(C, \bigcup_{C' \in C} C'\right) \qquad \textit{with } |Y| \geq 2$$

The evaluation of the formula above will always terminate because every C has at most $|Y|-1$ elements.

We now extend the formula to the case where two equivalences $\equiv'_f$ and $\equiv'_{g'}$ are given. The prime indicates that these equivalences are two "normal" equivalences and not a pair of equivalences such as $\equiv_f$, which in our setting consists of $\equiv^V_f$ and $\equiv^E_f$.

We introduce the following auxiliary notation:

- We define $Part(\equiv)$ as the partition consisting of the equivalence classes of a given equivalence $\equiv$.
- Let $Eq_{coll}\left(\equiv'_f, \equiv'_{g'}\right)$ be the set of all equivalences such that the equivalence closure of that equivalence and $\equiv'_f$ results in $\equiv'_{g'}$.

  Note that the subscript *coll* stands for "collapsing", i.e., we want to collapse $\equiv'_f$ in order to obtain $\equiv'_{g'}$. The connection to the earlier notation is the following. For an arbitrary equivalence $\equiv$ on a set X we have that $|Eq_{coll}(\equiv, X \times X)| = Cnt(Part(\equiv), X)$.
- Given an equivalence $\equiv$ on a set X and $Y \subseteq X$, we define $[\equiv]_X$ as the equivalence which results when we restrict $\equiv$ to elements of X, i.e., $[\equiv]_X = \equiv \cap (X \times X)$.

Regardless of $\equiv'_f$, only elements in the same class according to $\equiv'_{g'}$ may and must be merged. Hence we split the calculation and apply the previous formula to the classes of $\equiv^V_{g'}$ individually. The total number of possibilities is the product of the individual results, resulting in the following proposition.

Proposition 6 *For any pair of equivalences $\equiv'_f$ and $\equiv'_{g'}$, where $\equiv'_f$ is a refinement of $\equiv'_{g'}$, the number of equivalences $\equiv'$ with $\overline{\equiv' \cup \equiv'_f} = \equiv'_{g'}$ can be computed as:*

$$\left|Eq_{coll}\left(\equiv'_f, \equiv'_{g'}\right)\right| = \prod_{Y \in Part\left(\equiv'_{g'}\right)} \left|P_{coll}\left(Part\left(\left[\equiv'_f\right]_Y\right)\right)\right|$$

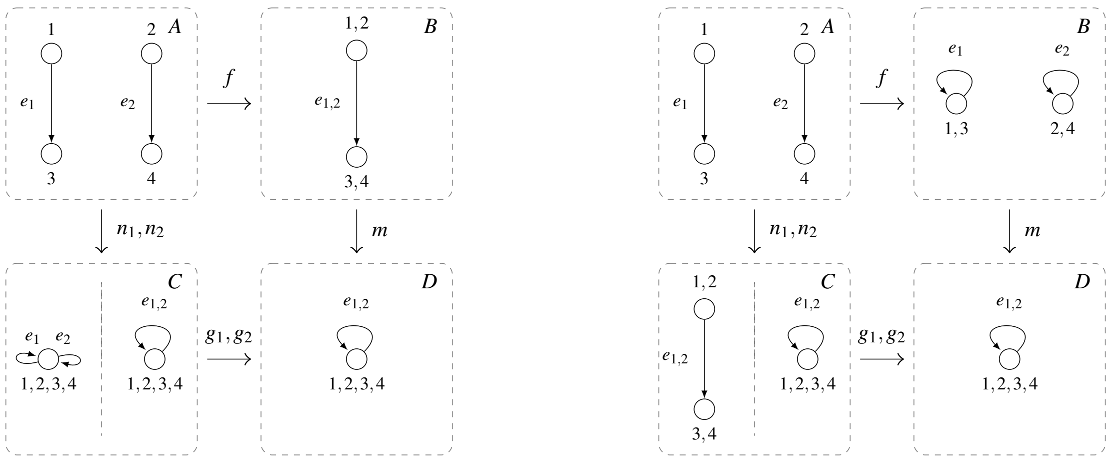
$$\textit{with } |P_{coll}(Z)| = Cnt\left(Z, \bigcup_{Z' \in Z} Z'\right)$$

Note that this formula can be used to calculate the equivalences on nodes and on edges. Using $\equiv^V_f$ and $\equiv^V_{g'}$ as initial values will result in the number of equivalences on nodes, whereas $\equiv^E_f$ and $\equiv^E_{g'}$ will result in the number of equivalences on edges.

When calculating the number of equivalences on nodes this formula does not take equivalences on edges into account, and vice versa. Fusion of edges would imply fusion of the connected nodes because of the properties of morphisms. Hence the number of equivalences on nodes is only equal to the number of equivalences on nodes and edges, and thus to the correct number of pushout complements, if no edges are merged. In the next section we will extend this formula to the general case.

5.4 Counting pushout complements with node and edge fusions

Edges can only be merged if also the connected nodes with equal indices are merged. On the one hand merging edges can increase the number of possibilities (compared to no merging of edges) if merging is optional, because every pushout complement, where these edges are not merged, results in an additional pushout complement, where the edges are merged. This is demonstrated in Figure 1a, where both graphs are correct pushout complements. On the other hand merging edges can decrease the possibilities if it is mandatory, because this reduces the possible equivalences on the nodes. In Figure 1b m merges e_1 and e_2, therefore this fusion is



(a) optional edge fusion (b) mandatory edge fusion

Figure 1: Examples of edge fusions

mandatory for any possible morphism n resulting in only two possible pushout complements. To compute the number of possibilities in the general case we first have to consider any possible equivalences on edges and then the problem can be solved using the results of Section 5.3.

Let V be the node set and E the edge set of the graph on which $\equiv_f$ and $\equiv_{g'}$ are defined. Note that both equivalences are pairs consisting of one equivalence on nodes and one on edges. Taking into account the connection between these equivalences, we define $ind(\equiv)$ as an equivalence on the node set V for an equivalence $\equiv$ on the edge set E as the induced equivalence on the nodes, i.e. two nodes are equivalent if and only if there are equivalent edges connected to the two nodes at the same index. Any equivalence we want to consider is coarser than this equivalence.

The computation of all possibilities can be divided into the sum of all possibilities assuming a valid equivalence on edges. Although each summand corresponds to a different equivalence on edges, the equivalence on nodes can be the same. For each such equivalence $\equiv^E$ we have to take into account the induced equivalence on nodes $ind\left(\equiv^E\right)$ before calculating the actual equivalences. Using $Part\left(ind\left(\equiv^E\right)\right)$ instead of the node set is equivalent to requiring that any possible equivalence must be coarser than $ind\left(\equiv^E\right)$. For this $\equiv_f^V$ has to be extended to $Part\left(ind\left(\equiv^E\right)\right)$ such that two elements of $Part\left(ind\left(\equiv^E\right)\right)$ are equivalent iff they contain some equivalent nodes. We combine these ideas in the following proposition.

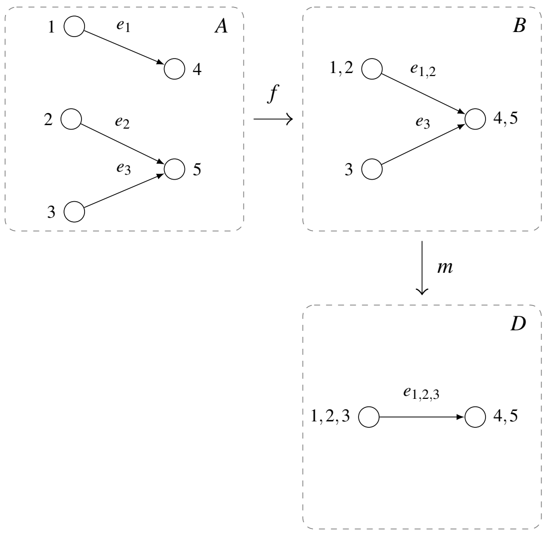
Proposition 7 *The number of equivalences on a graph such that the equivalence closure with $\equiv_f$ (on nodes and edges) results in $\equiv_{g'}$, which corresponds to the number of pushout complements for given morphisms f,m, can be computed by the following formula:*

$$\sum_{\equiv^E \in Eq_{coll}\left(\equiv_f^E, \equiv_{g'}^E\right)} \left| Eq_{coll}\left(\equiv_f^{P(\equiv^E)}, \equiv_{g'}^{P(\equiv^E)}\right) \right|$$

where $\equiv_h^{P(\equiv^E)}$ for $h \in \{f, g'\}$ is the equivalence $\equiv_h^V$ lifted to the induced partition $P(\equiv^E) = Part\left(ind\left(\equiv^E\right)\right)$ on nodes, i.e.:

$$X_1 \equiv_h^{P(\equiv^E)} X_2 \Leftrightarrow \exists Z \in Part\left(\overline{ind\left(\equiv^E\right) \cup \equiv_h^V}\right) : X_1 \subseteq Z \wedge X_2 \subseteq Z \; for \; X_1, X_2 \in Part\left(ind\left(\equiv^E\right)\right).$$

Example 3 *In the following we apply the above formula to the graphs and morphisms shown below.*



All edges are equivalent according to $\equiv_{g'}$ *and only* e_1 *and* e_2 *are equivalent according to* $\equiv_f$ *resulting in three possible equivalences on the edges. These equivalences are listed below together with their induced equivalence on the nodes and their lifting of* $\equiv_f$ *and* $\equiv_{g'}$. *We denote the set containing the nodes* 1 *and* 2 *by* $S_{1,2}$ *to indicate that these sets are no longer seen as sets but as objects (i.e. nodes) for later calculation.*

$$Part\left(\equiv_f^V\right) = \{\{1,2\},\{3\},\{4,5\}\} \qquad Part\left(\equiv_{g'}^V\right) = \{\{1,2,3\},\{4,5\}\}$$

- *Case 1:*

$$Part\left(\equiv_1^E\right) = \{\{e_1,e_2,e_3\}\} \qquad Part\left(\equiv_f^{P(\equiv_1^E)}\right) = \{\{S_{1,2,3}\},\{S_{4,5}\}\}$$

$$P(\equiv_1^E) = \{S_{1,2,3},S_{4,5}\} \qquad Part\left(\equiv_{g'}^{P(\equiv_1^E)}\right) = \{\{S_{1,2,3}\},\{S_{4,5}\}\}$$

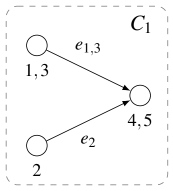
In this case every equivalence class of $\equiv_{g'}^{P(\equiv_1^E)}$ *contains only one element, hence nothing can be merged and only one equivalence is possible, because the edge fusions force the nodes to be merged as well. The resulting pushout complement is then identical to D, where* $n = m \circ f$ *and* g *is the identity.*

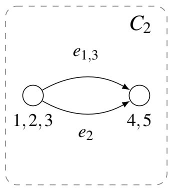
- *Case 2:*

$$Part\left(\equiv_2^E\right) = \{\{e_1,e_3\},\{e_2\}\} \qquad Part\left(\equiv_f^{P(\equiv_2^E)}\right) = \{\{S_{1,3},S_2\},\{S_{4,5}\}\}$$

$$P(\equiv_2^E) = \{S_{1,3},S_2,S_{4,5}\} \qquad Part\left(\equiv_{g'}^{P(\equiv_2^E)}\right) = \{\{S_{1,3},S_2\},\{S_{4,5}\}\}$$

Any valid equivalence can relate $S_{1,3}$ *and* S_2 *but need not because they are already related by* $\equiv_f^{P(\equiv_2^E)}$. *Nothing can be merged in the second class of* $\equiv_{g'}^{P(\equiv_2^E)}$, *hence there are two possibilities in this case resulting in the pushout complements below:*





- *Case 3:*

$$Part\left(\equiv_3^E\right) = \{\{e_1\},\{e_2,e_3\}\} \qquad Part\left(\equiv_f^{P(\equiv_3^E)}\right) = \{\{S_1,S_{2,3}\},\{S_4,S_5\}\}$$

$$P(\equiv_3^E) = \{S_1,S_{2,3},S_4,S_5\} \qquad Part\left(\equiv_{g'}^{P(\equiv_3^E)}\right) = \{\{S_1,S_{2,3}\},\{S_4,S_5\}\}$$

For $\{S_1,S_{2,3}\}$ equivalence is similar to that of case two. Additionally S_4 and S_5 can either be merged or not merged resulting in four different equivalences.

All in all we get seven possible equivalences on nodes and edges resulting in seven different pushout complements.

6 Conclusion

We have shown how to construct pushout complements when both given morphisms might be non-injective. Such a construction is necessary for performing backwards analysis and computing the set of predecessors of a given graph. We have implemented this construction (in a tool that performs backwards search in well-structured transition systems, based on [7]) and we presented the optimizations that we used in the implementation.

We gave some result on combinatorics that illustrate the potential combinatorial explosion in the number of pushout complements for an arbitrary pair f,m of morphisms. The computations needed to compute this number are quite involved and we tried to simplify the formulas as much as possible. It is unclear whether any further simplification or an easier theory is feasible.

Furthermore it is an open question whether the construction could be transferred to a more categorical setting, similar to [1].

Acknowledgements: We would like to thank Benjamin Braatz for our discussions on this topic.

Bibliography

[1] Benjamin Braatz, Ulrike Golas, and Thomas Soboll. How to delete categorically - two pushout complement constructions. *Journal of Symbolic Computation*, 46(3):246–271, March 2011.

[2] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe. Algebraic approaches to graph transformation—part I: Basic concepts and double pushout approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 1: Foundations*, chapter 3. World Scientific, 1997.

[3] H. Ehrig, R. Heckel, M. Korff, M. Löwe, L. Ribeiro, A. Wagner, and A. Corradini. Algebraic approaches to graph transformation—part II: Single pushout approach and comparison with double pushout approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations*, chapter 4. World Scientific, 1997.

[4] H. Ehrig, M. Pfender, and H. Schneider. Graph grammars: An algebraic approach. In *Proc. 14th IEEE Symp. on Switching and Automata Theory*, pages 167–180, 1973.

[5] Hartmut Ehrig. Introduction to the algebraic theory of graph grammars. In *Proc. 1st International Workshop on Graph Grammars*, pages 1–69. Springer-Verlag, 1979. LNCS 73.

[6] Annegret Habel, Jürgen Müller, and Detlef Plump. Double-pushout graph transformation revisited. *Mathematical Structures in Computer Science*, 11(5):637–688, 2001.

[7] Salil Joshi and Barbara König. Applying the graph minor theorem to the verification of graph transformation systems. In *Proc. of CAV '08*, pages 214–226. Springer, 2008. LNCS 5123.

[8] Yasuo Kawahara. Pushout-complements and basic concepts of grammars in toposes. *Theoretical Computer Science*, 77:267–289, 1990.

[9] Barry K. Rosen. Deriving graphs from graphs by applying a production. *Acta Informatica*, 4:337–357, 1975.