Relabelling in Graph Transformation

Annegret Habel[1] and Detlef Plump[2]

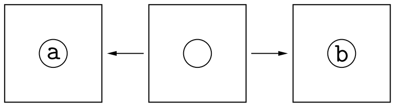
[1] Fachbereich Informatik, Universität Oldenburg
Postfach 2503, D-26111 Oldenburg, Germany
habel@informatik.uni-oldenburg.de

[2] Department of Computer Science, The University of York
York YO10 5DD, United Kingdom
det@cs.york.ac.uk

Abstract. The traditional double-pushout approach to graph transformation does not allow to change node labels in an arbitrary context. We propose a simple solution to this problem, namely to use rules with partially labelled interface graphs and to match rules injectively. In [8] we have shown that injective matching makes the double-pushout approach more expressive, and here we further generalise that approach. Besides solving the relabelling problem, our framework allows to write rules with partially labelled left-hand sides which are equivalent to (possibly infinite) sets of rules in the traditional setting. Unlike previous work on rules with partially labelled graphs, we do not need any labelling condition on matching morphisms, nor do we exclude node merging rules.

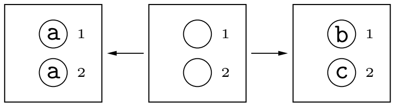
1 Introduction

The double-pushout approach to graph transformation has been studied for almost 30 years and has been applied in several areas of computer science, see the recent handbook volumes [12,4,7]. However, the traditional formulation of the double-pushout approach has the drawback that it is impossible to write a rule that changes the label of a node in an arbitrary context, that is, regardless of the number of edges incident to a node. For example, to change a label a into b one would like to write a rule like the following:



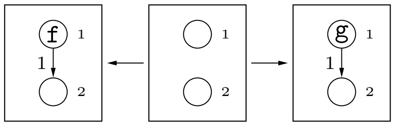
The node in the interface graph is unlabelled, but the traditional double-pushout approach is based on totally labelled graphs and label-preserving graph morphisms which exclude such a rule. (See [3,2] for introductions to the double-pushout approach.) A problem with partially labelled graphs is that the so-called gluing condition does no longer guarantee the applicability of a rule. This condition forbids to remove a node that is incident to a context edge or is the image of two distinct nodes in the rule. For example, consider the rule[1]

[1] Nodes are numbered to indicate the graph morphisms.



and a graph G consisting of a single node labelled with `a`. There is a non-injective graph morphism from the left-hand side of the rule to G, and this morphism satisfies the gluing condition since the rule preserves both nodes. But the rule is not applicable to G because, intuitively, it requires the label `a` of G's node to be changed to both `b` and `c`. This problem vanishes into thin air if the left-hand sides of rules are matched injectively.

If unlabelled nodes are used not only in interfaces of rules but also in left- and right-hand sides, single rules can specify transformations that otherwise require possibly infinite sets of rules. For example, the rule



implements the term rewriting rule `f(x)` $\to$ `g(x)` on node-labelled term graphs (see also Example 1 below). If we allowed only interface graphs to be partially labelled, this rule had to be replaced by a set of rules in which node 2 is labelled with all given function symbols and variables — of which there may be infinitely many.

In this paper we show that in the double-pushout approach on partially labelled graphs, a rule $r = \langle L \leftarrow K \rightarrow R\rangle$ is applicable to a graph G via an injective morphism $g\colon L \to G$ if and only if g satisfies the dangling condition. This condition requires that nodes to be removed must not be incident to context edges. Moreover, we show that the graph resulting from the rule application is unique up to isomorphism, and is totally labelled if G is totally labelled. For these results we require a mild condition on undefined labels in rules and that pushouts are "natural" in the sense that they are simultaneously pullbacks. Both requirements are always satisfied in the double-pushout approach with injective matching on totally labelled graphs, so our framework is a conservative extension of that approach.

The rest of this paper is organised as follows. The next section introduces partially labelled graphs, transformation rules and direct derivations in form of two natural pushouts. In Section 3 we show that the category of partially labelled graphs possesses pullbacks and give sufficient conditions for the existence of pushouts. The existence and uniqueness of natural pushout complements and direct derivations is proved in Section 4. We conclude in Section 5 by comparing our approach with previous work.

2 Graphs, Rules and Derivations

A *partially labelled graph* is a system $G = (\mathrm{V}_G, \mathrm{E}_G, \mathrm{s}_G, \mathrm{t}_G, \mathrm{l}_{G,\mathrm{V}}, \mathrm{l}_{G,\mathrm{E}})$ consisting of two finite sets V_G and E_G of *nodes* and *edges*, two source and target functions $\mathrm{s}_G, \mathrm{t}_G \colon \mathrm{E}_G \to \mathrm{V}_G$, and two partial labelling functions $\mathrm{l}_{G,\mathrm{V}} \colon \mathrm{V}_G \to \mathcal{C}_\mathrm{V}$ and $\mathrm{l}_{G,\mathrm{E}} \colon \mathrm{E}_G \to \mathcal{C}_\mathrm{E}$ [2], where $\mathcal{C}_\mathrm{V}$ and $\mathcal{C}_\mathrm{E}$ are fixed sets of node and edge labels. For simplicity, partially labelled graphs are called *graphs* in the following. A graph G is *totally labelled* if $\mathrm{l}_{G,\mathrm{V}}$ and $\mathrm{l}_{G,\mathrm{E}}$ are total functions.

A *graph morphism* $g \colon G \to H$ between two graphs G and H consists of two functions $g_\mathrm{V} \colon \mathrm{V}_G \to \mathrm{V}_H$ and $g_\mathrm{E} \colon \mathrm{E}_G \to \mathrm{E}_H$ that preserve sources, targets and labels, that is, $\mathrm{s}_H \circ g_\mathrm{E} = g_\mathrm{V} \circ \mathrm{s}_G$, $\mathrm{t}_H \circ g_\mathrm{E} = g_\mathrm{V} \circ \mathrm{t}_G$, and $\mathrm{l}_H(g(x)) = \mathrm{l}_G(x)$ for all x in $\mathrm{Dom}(\mathrm{l}_G)$ [3]. The morphism g *preserves undefinedness* if $\mathrm{l}_H(g(x)) = \bot$ for all x in $G - \mathrm{Dom}(\mathrm{l}_G)$, and it *reflects undefinedness* if $g^{-1}(x) \neq \emptyset$ for all x in $H - \mathrm{Dom}(\mathrm{l}_H)$. A morphism g is *injective* (*surjective*) if g_V and g_E are injective (surjective), and an *isomorphism* if it is injective, surjective and preserves undefinedness. In the latter case G and H are *isomorphic*, which is denoted by $G \cong H$. Furthermore, we call g an *inclusion* if $g(x) = x$ for all x in G. (Note that inclusions need not preserve undefinedness.) Partially labelled graphs and graph morphisms constitute a category, where composition of morphisms is defined componentwise as function composition.

Definition 1 (Rule). A *rule* $r = \langle L \leftarrow K \to R \rangle$ consists of two graph morphisms $K \to L$ and $b \colon K \to R$ such that $K \to L$ is an inclusion and

(1) for all $x \in L$, $\mathrm{l}_L(x) = \bot$ implies $x \in K$ and $\mathrm{l}_R(b(x)) = \bot$,
(2) for all $x \in R$, $\mathrm{l}_R(x) = \bot$ implies $\mathrm{l}_L(x') = \bot$ for exactly one $x' \in b^{-1}(x)$.

We call L the *left-hand side*, R the *right-hand side* and K the *interface* of r. The rule r is *injective* if $b \colon K \to R$ is injective. Note that conditions (1) and (2) are trivially satisfied if L and R are totally labelled[4].

Example 1. In *term graph rewriting* one implements term rewriting systems by graph transformation to improve the efficiency of computations (see [10] for a survey). Our framework allows to translate term rewriting rules into graph transformation rules operating on node-labelled term graphs. A node with a function symbol `f` of arity n has n outgoing edges, labelled with 1 to n, whose destinations represent the arguments of `f`. As an example for the translation of term rewriting rules into graph transformation rules, Figure 1 shows the rule corresponding to the term rewriting rule `f(x)` $\to$ `x`. Node 2, representing the variable `x`, has to be unlabelled to make the rule applicable independently of `f`'s argument. Upon application of the rule, nodes 1 and 2 are merged into a single node labelled with the function symbol of `f`'s argument.

[2] Given sets A and B, a partial function $f \colon A \to B$ is a function from some subset A' of A to B. The set A' is the *domain* of f and is denoted by $\mathrm{Dom}(f)$. We say that $f(x)$ is *undefined*, and write $f(x) = \bot$, if x is in $A - \mathrm{Dom}(f)$.

[3] We often do not distinguish between nodes and edges in statements that hold analogously for both sets.

[4] We also remark that in (2) the more liberal "for at most one $x' \in b^{-1}(x)$" could be used, but then Theorem 2 had to require the present condition.

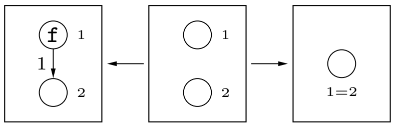


Fig. 1. A rule implementing the term rewriting rule $\mathtt{f(x)} \to \mathtt{x}$

A diagram of graph morphisms

$$\begin{tikzcd} K \arrow[r] \arrow[d] & R \arrow[d] \\ D \arrow[r] & H \end{tikzcd}$$

is a *pushout* if (i) $K \to R \to H = K \to D \to H$ and (ii) for every pair of graph morphisms $\langle R \to H', D \to H' \rangle$ with $K \to R \to H' = K \to D \to H'$, there is a unique morphism $H \to H'$ such that $R \to H' = R \to H \to H'$ and $D \to H' = D \to H \to H'$. The above diagram is a *pullback* if property (i) holds and if for every pair of graph morphisms $\langle K' \to R, K' \to D \rangle$ with $K' \to R \to H = K' \to D \to H$, there is a unique morphism $K' \to K$ such that $K' \to R = K' \to K \to R$ and $K' \to D = K' \to K \to D$. A pushout is *natural* if it is simultaneously a pullback.

Definition 2 (Direct derivation). A *direct derivation* from a graph G to a graph H via a rule $r = \langle L \leftarrow K \to R \rangle$ consists of two natural pushouts as in Figure 2, where $g\colon L \to G$ is injective. We write $G \Rightarrow_{r,g} H$ if there exists such a direct derivation.

$$\begin{tikzcd} L \arrow[d] & K \arrow[l] \arrow[r] \arrow[d] \arrow[dl, phantom, "(1)"] \arrow[dr, phantom, "(2)"] & R \arrow[d] \\ G & D \arrow[l] \arrow[r] & H \end{tikzcd}$$

Fig. 2. A direct derivation

Remark 1. Given a rule $r = \langle L \leftarrow K \to R \rangle$ and an injective morphism $L \to G$, the pushouts in Figure 2 are natural if L, K and R are totally labelled. But in general a rule may admit both a natural and a non-natural double-pushout. For example, in Figure 3 only the left double-pushout is natural. We characterise the naturalness of pushouts in Lemma 3.

Remark 2. Employing injective matching in the double-pushout approach is not a restriction since every derivation with a set of rules $\mathcal{R}$ in the traditional approach can be simulated, using injective matching, by a derivation with the set

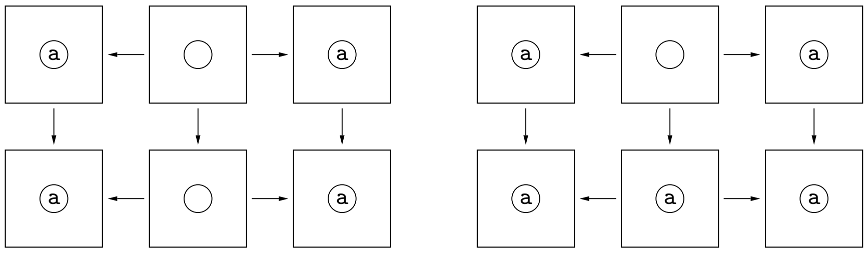


Fig. 3. A natural and a non-natural double-pushout

Q($\mathcal{R}$) of quotient rules of $\mathcal{R}$. The quotient rules of a rule are obtained, roughly speaking, by considering all possible identifications among the nodes and edges of the rule. Actually, injective matching makes the double-pushout approach more expressive because, by omitting some quotients, one gets a finer control on transformations than in the traditional framework. We refer to [8] for precise results on the expressiveness gained in this way.

3 Existence of Pullbacks and Pushouts

In this section we prove two lemmata about the existence of pullbacks and pushouts in the category of partially labelled graphs. We also characterise the naturalness of pushouts.

Lemma 1 (Existence of pullbacks). *Given graph morphisms $g\colon L \to G$ and $c\colon D \to G$, there exist a graph K and graph morphisms $b\colon K \to L$ and $d\colon K \to D$ such that diagram (1) in Figure 4 is a pullback.*

$$\begin{tikzcd}
 & & K' \arrow[dll, "b'"'] \arrow[dl, "u"'] \arrow[ddl, "d'"] \\
L \arrow[d, "g"'] \arrow[dr, phantom, "(1)"] & K \arrow[l, "b"'] \arrow[d, "d"] & \\
G & D \arrow[l, "c"] &
\end{tikzcd}$$

Fig. 4. A pullback

Proof. The node and edge sets of K are constructed as $\{\langle x, y\rangle \in L \times D \mid g(x) = c(y)\}$, the source mapping is defined by $\mathrm{s}_K(\langle x, y\rangle) = \langle \mathrm{s}_L(x), \mathrm{s}_D(y)\rangle$, the target mapping is defined analogously, and the labelling is given by

$$\mathrm{l}_K(\langle x, y\rangle) = \underline{\text{if}}\ \mathrm{l}_L(x) = \mathrm{l}_D(y) \neq \bot\ \underline{\text{then}}\ \mathrm{l}_L(x)\ \underline{\text{else}}\ \bot.$$

Let $b\colon K \to L$ and $d\colon K \to D$ be the projections from $L \times D$ to L and D, that is, $b(\langle x, y\rangle) = x$ and $d(\langle x, y\rangle) = y$, separately for nodes and edges. As in the case of totally labelled graphs, it is easy to check that b and d preserve the sources and targets of edges. To see that b and d preserve labels, consider $\langle x, y\rangle \in \mathrm{Dom}(\mathrm{l}_K)$. Then $\mathrm{l}_L(b(\langle x, y\rangle)) = \mathrm{l}_L(x) = \mathrm{l}_K(\langle x, y\rangle)$ and $\mathrm{l}_D(d(\langle x, y\rangle)) = \mathrm{l}_D(y) = \mathrm{l}_L(x) = \mathrm{l}_K(\langle x, y\rangle)$ by definition of b, d and l_K. Hence b and d are graph morphisms.

Next we show that square (1) in Figure 4 is a pullback. By definition of b and d, (1) commutes. To see that (1) satisfies the universal property, let $b'\colon K' \to L$ and $d'\colon K' \to D$ be graph morphisms with $g \circ b' = c \circ d'$. There is only one choice (implying uniqueness) for a morphism $u\colon K' \to K$ such that $b \circ u = b'$ and $d \circ u = d'$: define $u(z) = \langle b'(z), d'(z)\rangle$, for all $z \in K'$. It remains to be shown that u is a graph morphism. The proof that u preserves sources and targets of edges is the same as in the case of totally labelled graphs, see [3]. To show that u is label-preserving, let $z \in \mathrm{Dom}(\mathrm{l}_{K'})$. If $\mathrm{l}_L(b'(z)) = \mathrm{l}_D(d'(z)) \neq \perp$, we have $\mathrm{l}_K(u(z)) = \mathrm{l}_K(\langle b'(z), d'(z)\rangle) = \mathrm{l}_L(b'(z)) = \mathrm{l}_{K'}(z)$ by definition of u and l_K, and since b' is label-preserving. If $\mathrm{l}_L(b'(z)) = \perp$ or $\mathrm{l}_D(d'(z)) = \perp$, then $\mathrm{l}_K(u(z)) = \mathrm{l}_K(\langle b'(z), d'(z)\rangle) = \perp = \mathrm{l}_{K'}(z)$ by definition of u and l_K, and since b' and d' are label-preserving. This concludes the proof that diagram (1) is a pullback. □

Lemma 2 (Existence of pushouts). *Let $b\colon K \to R$ and $d\colon K \to D$ be graph morphisms such that d is injective and for all x in R, $\{\mathrm{l}_R(x)\} \cup \mathrm{l}_D(d(b^{-1}(x)))$ contains at most one element. Then there exist a graph H and graph morphisms $R \to H$ and $D \to H$ such that diagram (2) in Figure 5 is a pushout.*

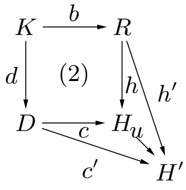

$$\begin{tikzcd}
K \arrow[r, "b"] \arrow[d, "d"'] \arrow[dr, phantom, "(2)"] & R \arrow[d, "h"] \arrow[ddr, "h'"] & \\
D \arrow[r, "c"'] \arrow[drr, "c'"'] & H \arrow[dr, "u"] & \\
& & H'
\end{tikzcd}$$

Fig. 5. A pushout

Proof. Construct the node and edge sets of H as $(D - d(K)) + R$, and define $\mathrm{s}_H(e) =$ <u>if</u> $e \in E_R$ <u>then</u> $\mathrm{s}_R(e)$ <u>else</u> $\mathrm{s}_D(e)$, $\mathrm{t}_H(e)$ analogously to $\mathrm{s}_H(e)$, and

$$\mathrm{l}_H(x) = \begin{cases} \mathrm{l}_R(x) & \text{if } x \in R \text{ and } \mathrm{l}_R(x) \neq \perp, \\ \mathrm{l}_D(d(x')) & \text{if } x \in R,\ \mathrm{l}_R(x) = \perp \text{ and } \mathrm{l}_D(d(x')) \neq \perp \\ & \text{for some } x' \in b^{-1}(x), \\ \perp & \text{if } x \in R,\ \mathrm{l}_R(x) = \perp \text{ and } \mathrm{l}_D(d(x')) = \perp \\ & \text{for all } x' \in b^{-1}(x), \\ \mathrm{l}_D(x) & \text{if } x \in (D - d(K)). \end{cases}$$

Note that l_H is well-defined because for all $x \in R$, the set $\{\mathrm{l}_R(x)\} \cup \mathrm{l}_D(d(b^{-1}(x)))$ contains at most one element. Now define $h\colon R \to H$ and $c\colon D \to H$ by $h(x) = x$

and $c(x) = \underline{\text{if}}\ x = d(x')$ for some $x' \in K$ $\underline{\text{then}}\ b(x')$ $\underline{\text{else}}\ x$, separately for nodes and edges. The proof that h and c preserve sources and targets is the same as in the case of totally labelled graphs, see [3]. To show that h and c preserve labels, let $x \in \mathrm{Dom}(\mathrm{l}_R)$. Then $\mathrm{l}_H(h(x)) = \mathrm{l}_R(x)$ by definition of h and l_H, so h is label-preserving. Consider now $x \in \mathrm{Dom}(\mathrm{l}_D)$. If $x \in (D - d(K))$, then $\mathrm{l}_H(c(x)) = \mathrm{l}_D(x)$ by definition of c and l_H. If $x \in d(K)$, then $x = d(x')$ for some $x' \in K$. There are two cases:

– $\mathrm{l}_R(b(x')) \neq \perp$. Then $\mathrm{l}_H(c(x)) = \mathrm{l}_R(b(x')) = \mathrm{l}_D(d(x')) = \mathrm{l}_D(x)$ by definition of l_H, the assumption that for all $x \in R$, $\{\mathrm{l}_R(x)\} \cup \mathrm{l}_D(d(b^{-1}(x)))$ contains at most one element, the fact that $x \in \mathrm{Dom}(\mathrm{l}_D)$, and the fact that d is label-preserving.
– $\mathrm{l}_R(b(x')) = \perp$. Then $\mathrm{l}_H(c(x)) = \mathrm{l}_D(d(x')) = \mathrm{l}_D(x)$ by the commutativity of the diagram, the definition of l_H and the fact that d is label-preserving.

Thus h and c are graph morphisms.

Next we show that diagram (2) in Figure 5 is a pushout. By definition of h and c, we have $h \circ b = c \circ d$. To show the universal property, let $h': R \to H'$ and $c': D \to H'$ be graph morphisms with $h' \circ b = c' \circ d$. There is only one choice (implying uniqueness) to define $u: H \to H'$ such that $u \circ h = h'$ and $u \circ c = c'$: let $u(x) = \underline{\text{if}}\ x \in R$ $\underline{\text{then}}\ h'(x)$ $\underline{\text{else}}\ c'(x)$, separately for nodes and edges. It remains to be shown that u is a graph morphism. That u is structure-preserving follows as in the proof of Lemma 2.8 in [3]. To show that u preserves labels, let $x \in \mathrm{Dom}(\mathrm{l}_H)$. We have three cases:

– $x \in R$ and $\mathrm{l}_R(x) \neq \perp$. Then $\mathrm{l}_{H'}(u(x)) = \mathrm{l}_{H'}(h'(x)) = \mathrm{l}_R(x) = \mathrm{l}_H(x)$ by the definition of u, the fact that $h': R \to H'$ preserves labels, and the definition of l_H.
– $x \in R$, $\mathrm{l}_R(x) = \perp$ and $\mathrm{l}_D(d(x')) \neq \perp$ for some $x' \in b^{-1}(x)$. Then $\mathrm{l}_{H'}(u(x)) = \mathrm{l}_{H'}(h'(b(x'))) = \mathrm{l}_{H'}(c'(d(x'))) = \mathrm{l}_D(x) = \mathrm{l}_H(x)$ by the definition of u, the fact that $h' \circ b = c' \circ d$, the fact that $c': D \to H'$ preserves labels, and the definition of l_H.
– $x \in (D - d(K))$. Then $\mathrm{l}_{H'}(u(x)) = \mathrm{l}_{H'}(c'(x)) = \mathrm{l}_D(x) = \mathrm{l}_H(x)$ by definition of u, the fact that $c': D \to H'$ preserves labels, and the definition of h.

This concludes the proof that square (2) in Figure 5 is a pushout. □

Next we characterise the naturalness of pushouts.

Lemma 3 (Characterisation of natural pushouts). *Given two graph morphisms $b: K \to L$ and $d: K \to D$ such that b is injective, pushout (1) in Figure 2 is natural if and only if*

(∗) for all $z \in K$, $\mathrm{l}_K(z) = \perp$ implies $\mathrm{l}_L(b(z)) = \perp$ or $\mathrm{l}_D(d(z)) = \perp$.

Proof. Let square (1) in Figure 2 be a natural pushout with graph morphisms $g: L \to G$ and $c: D \to G$. Since (1) is a pullback, an explicit construction of K up to isomorphism is given by $\{\langle x, y\rangle \in L \times D \mid g(x) = c(y)\}$, separately for nodes and edges, where $\mathrm{l}_K(\langle x,y\rangle) = \mathrm{l}_L(x)$ if and only if $\mathrm{l}_L(x) = \mathrm{l}_D(y) \neq \bot$. It follows that $\mathrm{l}_K(\langle x,y\rangle) = \bot$ if and only if $\mathrm{l}_L(x) = \bot$ or $\mathrm{l}_D(y) = \bot$. Hence condition $(*)$ is satisfied.

Conversely, let (1) be a pushout satisfying condition $(*)$. In the diagram below, let (1') be a pullback of $g: L \to G$ and $d: D \to G$. Let b' and d' be the morphisms from K' to L and D, respectively. By the universal property of (1'), there exists a unique morphism $u: K \to K'$ such that $b' \circ u = b$ and $d' \circ u = d$.

$$\begin{tikzcd}
L \arrow[dd, "g"'] & & K \arrow[ll, "b"'] \arrow[dl] \arrow[dd, "d"] \\
 & K' \arrow[ul] \arrow[dr] \arrow[u, phantom, "="] \arrow[r, phantom, "="] \arrow[dl, phantom, "(1')"] & \\
G & & D \arrow[ll, "c"]
\end{tikzcd}$$

We show that u is an isomorphism. Injectivity of u follows from injectivity of $b = b' \circ u$. To see that u is surjective, consider some $z' \in K'$. Then $g(b'(z')) = c(d'(z'))$ by commutativity of diagram (1'). Hence, by injectivity of b and the pushout characterisation of [5] (Theorem 1.2) applied to diagram (1), there is some $z \in K$ such that $b(z) = b'(z')$ and $d(z) = d'(z')$. Applying the pullback characterisation of [5] (Theorem 1.7) to (1') gives $u(z) = z'$.

Finally, u preserves undefinedness by condition $(*)$. Thus u is an isomorphism, implying that diagram (1) is a pullback and hence a natural pushout. □

4 Existence and Uniqueness of Direct Derivations

Our main result (Theorem 1) will show that, given a rule $r = \langle L \leftarrow K \to R\rangle$ and an injective graph morphism $g: L \to G$, there exists a direct derivation as in Figure 2 if and only if g satisfies the dangling condition. Moreover, in this case r and g determine D and H uniquely up to isomorphism. The proof of this result is based on the following lemma which provides a condition for the existence and uniqueness of “natural pushout complements”.

A graph morphism $g: L \to G$ satisfies the *dangling condition* with respect to an inclusion $K \to L$, if no node in $g(L) - g(K)$ is incident to an edge in $G - g(L)$.

Lemma 4 (Existence and uniqueness of natural pushout complements). *Let $g: L \to G$ be an injective graph morphism and $K \to L$ an inclusion that reflects undefinedness. Then there exist a graph D and graph morphisms $K \to D$ and $D \to G$ such that diagram (1) in Figure 2 is a natural pushout if and only if g satisfies the dangling condition. Moreover, in this case D is unique up to isomorphism.*

Proof. “Only if”: Let diagram (1) in Figure 2 be a pushout. By forgetting all labels, this diagram becomes a pushout in the category of unlabelled graphs. By the pushout characterisation of [5] (Theorem 1.2), g satisfies the dangling condition in that situation. Since the dangling condition is not concerned with labels, g in diagram (1) satisfies the dangling condition as well.

"If": Let g be an injective graph morphism satisfying the dangling condition. Construct the graph D as follows: D has node and edge sets $(G - g(L)) + g(K)$, and s_D and t_D are the restrictions of s_G and t_G to these sets. Note that s_D and t_D are well-defined because g satisfies the dangling condition. We define D's labelling by

$$\mathrm{l}_D(x) = \ \underline{\text{if}}\ x = g(x') \text{ and } \mathrm{l}_L(x') \neq \bot \text{ for some } x' \text{ in } K\ \underline{\text{then}}\ \mathrm{l}_K(x')\ \underline{\text{else}}\ \mathrm{l}_G(x),$$

separately for nodes and edges. These mappings are well-defined because $K \to L$ and $L \to G$ are injective. Next define the graph morphism $c\colon D \to G$ as the inclusion of D in G and the morphism $d\colon K \to D$ as the restriction of $K \to L \to G$ to D. In [11] (Lemma 4.4) it is shown that c and d defined in this way preserve sources and targets. We show that they preserve labels in our case. For $x \in \mathrm{Dom}(\mathrm{l}_K)$, we have $\mathrm{l}_D(d(x)) = \mathrm{l}_K(x)$ by definition of d and l_D. So d is label-preserving. To see that c is label-preserving as well, let $x \in \mathrm{Dom}(\mathrm{l}_D)$. There are two cases to consider.

- $x \in G - g(L)$. Then $\mathrm{l}_G(c(x)) = \mathrm{l}_G(x) = \mathrm{l}_D(x)$ by the definition of c and l_D.
- $x \in g(K)$. Let $x' \in K$ with $g(x') = x$. If $\mathrm{l}_L(x') \neq \bot$, then $\mathrm{l}_G(c(x)) = \mathrm{l}_G(x) = \mathrm{l}_G(g(x')) = \mathrm{l}_L(x') = \mathrm{l}_K(x') = \mathrm{l}_D(x)$ by definition of c, the fact that $K \to L$ and $L \to G$ are label-preserving, and the definition of l_D. On the other hand, if $\mathrm{l}_L(x') = \bot$, then $\mathrm{l}_G(c(x)) = \mathrm{l}_G(x) = \mathrm{l}_D(x)$ by definition of l_D.

So c is label-preserving and hence c and d are graph morphisms in the category of partially labelled graphs.

By forgetting all labels, diagram (1) becomes a diagram in the category of unlabelled graphs. In [11] (Lemma 4.4) it is shown that (1) is a pushout in that case. Denote the inclusion $K \to L$ by b. The pushout property means that $g \circ b = c \circ d$, and that for every unlabelled G' and graph morphisms $g'\colon L \to G'$ and $c'\colon D \to G'$ with $g' \circ b = c' \circ d$, there is a unique morphism $u\colon G \to G'$ such that $u \circ g = g'$ and $u \circ c = c'$. To prove that (1) remains a pushout if the graphs have labels, we have to show that u preserves labels. Let $x \in \mathrm{Dom}(\mathrm{l}_G)$. Since $G = (D - d(K)) \cup g(L)$, separately for nodes and edges, we consider the following cases:

- $x \in g(L)$ and $\mathrm{l}_L(\bar{x}) \neq \bot$ for the unique $\bar{x} \in L$ with $g(\bar{x}) = x$. Then $\mathrm{l}_{G'}(u(x)) = \mathrm{l}_{G'}(u(g(\bar{x}))) = \mathrm{l}_{G'}(g'(\bar{x})) = \mathrm{l}_L(\bar{x}) = \mathrm{l}_G(g(\bar{x})) = \mathrm{l}_G(x)$ since g' and g are label-preserving.
- $x \in g(L)$ and $\mathrm{l}_L(\bar{x}) = \bot$ for the unique $\bar{x} \in L$ with $g(\bar{x}) = x$. Since the inclusion $K \to L$ reflects undefinedness, we have $\bar{x} \in K$. By definition of l_D, $\mathrm{l}_D(d(\bar{x})) = \mathrm{l}_G(d(\bar{x})) = \mathrm{l}_G(g(\bar{x})) = \mathrm{l}_G(x)) \neq \bot$. Thus $\mathrm{l}_{G'}(u(x)) = \mathrm{l}_{G'}(u(g(\bar{x}))) = \mathrm{l}_{G'}(u(d(\bar{x}))) = \mathrm{l}_{G'}(c'(d(\bar{x}))) = \mathrm{l}_D(d(\bar{x})) = \mathrm{l}_G(x)$, since c' preserves labels.
- $x \in (D - d(K))$. Then $\mathrm{l}_{G'}(u(x)) = \mathrm{l}_{G'}(c'(x)) = \mathrm{l}_D(x) = \mathrm{l}_G(x)$ by definition of l_D and since c' is label-preserving.

So u preserves labels and hence diagram (1) is a pushout in the category of partially labelled graphs. Moreover, by definition of l_D, property $(*)$ of Lemma

3 holds: for all $x \in K$, $\mathrm{l}_K(x) = \bot$ implies $\mathrm{l}_L(x) = \bot$ or $\mathrm{l}_D(d(x)) = \bot$. Thus, by Lemma 3, diagram (1) is a natural pushout.

It remains to be shown that D is unique up to isomorphism. Again we first forget the labels of all graphs in diagram (1) to obtain a pushout in the category of unlabelled graphs. Then by Lemma 4.2 in [11], D is uniquely determined up to isomorphism. To lift this result to our setting, we have to show that D's labelling is uniquely determined. Consider any $x \in D$.

Case 1: $x \in (G - g(L))$. It can be shown that for every pushout of form (1), for each $x \in G$ there is $x' \in L$ with $g(x') = x$ and $\mathrm{l}_L(x') = \mathrm{l}_G(x)$ or there is $x' \in D$ with $c(x') = x$ and $\mathrm{l}_D(x') = \mathrm{l}_G(x)$. Since in the present case $x \notin g(L)$ and c is an inclusion, $\mathrm{l}_D(x)$ must be equal to $\mathrm{l}_G(x)$.

Case 2: $x \in g(K)$. Let x' be the unique element in K with $g(x') = x$.

Case 2.1: $\mathrm{l}_L(x') = \bot$. Then by the property of pushouts mentioned in Case 1 and since c preserves labels, again $\mathrm{l}_D(x)$ must equal $\mathrm{l}_G(x)$.

Case 2.2: $\mathrm{l}_L(x') \neq \bot$. If $\mathrm{l}_K(x') \neq \bot$, then $\mathrm{l}_D(x) = \mathrm{l}_D(d(x')) = \mathrm{l}_K(x')$ because d preserves labels. If $\mathrm{l}_K(x') = \bot$, then by the characterisation of natural pushouts in Lemma 3, $\mathrm{l}_D(x) = \mathrm{l}_D(d(x')) = \bot$. □

The following theorem is our main result.

Theorem 1 (Existence and uniqueness of direct derivations). *Given a rule $r = \langle L \leftarrow K \rightarrow R \rangle$ and an injective graph morphism $g\colon L \rightarrow G$, there exists a direct derivation as in Figure 2 if and only if g satisfies the dangling condition. Moreover, in this case D and H are unique up to isomorphism.*

Proof. "Only if": For a direct derivation as in Figure 2, g satisfies the dangling condition by Lemma 4.

"If": If g satisfies the dangling condition, then by Lemma 4 there exist a graph D and graph morphisms $d\colon K \rightarrow D$ and $D \rightarrow G$ such that diagram (1) in Figure 2 is a natural pushout. In order to obtain pushout (2) of Figure 2 by Lemma 2, we have to show that for all $x \in R$, the set $\{\mathrm{l}_R(x)\} \cup \mathrm{l}_D(d(b^{-1}(x)))$ contains at most one element.

Case 1: $\mathrm{l}_R(x) = \bot$. Then $\mathrm{l}_K(x') = \bot$ for all $x' \in b^{-1}(x)$ because b preserves labels. By the definition of a rule, there is exactly one $x' \in b^{-1}(x)$ with $\mathrm{l}_L(x') = \bot$. Hence for each other element $\bar{x} \in b^{-1}(x)$, $\mathrm{l}_L(\bar{x}) \neq \bot$ and therefore $\mathrm{l}_D(d(\bar{x})) = \bot$ by the characterisation of natural pushouts in Lemma 3. Thus $\{\mathrm{l}_R(x)\} \cup \mathrm{l}_D(d(b^{-1}(x)))$ contains at most one label.

Case 2: $\mathrm{l}_R(x) \neq \bot$. Then, by the definition of a rule, $\mathrm{l}_L(x') \neq \bot$ for all $x' \in b^{-1}(x)$. Consider any $x' \in b^{-1}(x)$.

Case 2.1: $\mathrm{l}_K(x') \neq \bot$. Then $\mathrm{l}_D(d(x')) = \mathrm{l}_K(x') = \mathrm{l}_R(b(x')) = \mathrm{l}_R(x)$ because d and b preserve labels.

Case 2.1: $\mathrm{l}_K(x') = \bot$. Then $\mathrm{l}_D(d(x')) = \bot$ by the characterisation of natural pushouts in Lemma 3.

Hence in Case 2, elements in $b^{-1}(x)$ are either labelled with $\mathrm{l}_R(x)$ or are unlabelled. Thus again $\{\mathrm{l}_R(x)\} \cup \mathrm{l}_D(d(b^{-1}(x)))$ contains at most one label.

Now Lemma 2 shows that pushout (2) in Figure 2 exists. We also have to show that (2) is natural. Consider any $x \in K$ with $\mathrm{l}_K(x) = \bot$ and $\mathrm{l}_R(b(x)) \neq \bot$. Then

$\mathrm{l}_L(x) \neq \perp$ by the definition of a rule. Hence $\mathrm{l}_D(d(x)) = \perp$ by the naturalness of pushout (1) and Lemma 3. Thus, by Lemma 3, diagram (2) is a natural pushout.

Finally, by Lemmas 4 and 2, the graphs D and H are uniquely determined up to isomorphism. This concludes the proof of Theorem 1. □

Theorem 2 (Totality of labelling). *For every direct derivation $G \Rightarrow H$, H is totally labelled if and only if G is totally labelled.*

Proof. "If": Let G be totally labelled. By the uniqueness of H up to isomorphism (Theorem 1) and the proof of Lemma 2, we can assume that H has node and edge sets $(D - d(K)) + R$, where D is constructed as in the proof of Lemma 4. Let $x \in H$. We consider three cases. If $x \in (D - d(K))$, then $\mathrm{l}_H(x) = \mathrm{l}_D(x) = \mathrm{l}_G(x) \neq \perp$ by definition of l_H and l_D, and since G is totally labelled. If $x \in R$ with $\mathrm{l}_R(x) \neq \perp$, then $\mathrm{l}_H(x) = \mathrm{l}_R(x) \neq \perp$. Finally, if $x \in R$ with $\mathrm{l}_R(x) = \perp$, then $\mathrm{l}_L(x') = \perp$ for exactly one $x' \in b^{-1}(x)$ by the definition of a rule. Hence $\mathrm{l}_H(x) = \mathrm{l}_D(d(x')) = \mathrm{l}_G(g(x')) \neq \perp$ by definition of l_H and l_D, and because G is totally labelled.

"Only if": Let $x \in G$ with $\mathrm{l}_G(x) = \perp$. We consider two cases, using the constructions of D and H referred to above. If $x \notin g(L)$, then $\mathrm{l}_H(x) = \mathrm{l}_D(x) = \mathrm{l}_G(x) = \perp$ by definition of l_H and l_D. If $x \in g(L)$, then by the definition of a rule there is some $x' \in K$ such that $g(x') = x$ and $\mathrm{l}_R(b(x')) = \perp$. Since also $\mathrm{l}_D(d(x')) = \mathrm{l}_D(x) = \perp$, we have $\mathrm{l}_H(h(x')) = \perp$ by definition of l_H in Lemma 2. □

5 Related Work

In this section we compare our approach with previous work on partially labelled graphs in the double-pushout approach.

Rosen [11] allows partially labelled graphs in rules (but requires that transformed graphs are totally labelled) and considers possibly non-injective matching morphisms. To ensure the existence of direct derivations, matching morphisms must not only satisfy the gluing condition but also an additional labelling condition. Moreover, non-injective rules as the rule in Figure 1 cannot be handled. Probably the most serious drawback of [11] is that direct derivations are not double-pushouts in the usual sense but consist of two pushouts which need not share a common morphism. That is, the morphism $K \to D$ in Figure 2 is replaced by two morphisms $K \to D_1$ and $K \to D_2$ where the transition from D_1 to D_2 corresponds to a relabelling. This feature makes the approach of [11] complicated, especially for proving properties of derivations.

In [6], Ehrig et al. extend the double-pushout approach from graphs to relational structures. Their results can be specialised to partially labelled graphs. Corollary 5.9 in [6] states under which conditions certain special derivations starting from a totally labelled graph yield a totally labelled graph. The approach of [6] (specialised to graphs) is more restrictive than the present approach in that only injective rules are allowed. Moreover, since matching morphisms are allowed to be non-injective, a labelling condition has to be satisfied in addition to the gluing condition.

Parisi-Presicce, Ehrig and Montanari [9] extend the traditional double-push-out approach by considering graphs with labels from an alphabet equipped with a partial order. Since every partially labelled graph can be seen as a totally labelled graph over a label alphabet containing a special label $\perp$ and being equipped with the flat order $\{\langle\perp, x\rangle \mid \perp \neq x\}$, their approach can be specialised to the case of partially labelled graphs. Again [9] is more restrictive than the present approach in that only injective rules are allowed. For example, our rule in Figure 1 cannot be used. Moreover, an injective rule like the second rule in the introduction does not satisfy the consistency condition of [9]. Apart from these restrictions, Proposition 3.13 in [9] states that a direct derivation exists if and only if the gluing condition is satisfied. However, a counterexample is given by the rule having a single node labelled with $\perp$ as left-hand side, an empty interface and an empty right-hand side (which is allowed in [9]). The unique graph morphism from the left-hand side to a graph consisting of a single node not labelled with $\perp$ satisfies the gluing condition, but there does not exist a pushout complement in this situation.

Finally, Berthold, Fischer and Koch [1] recently considered the double-push-out approach over partially attributed graphs. They employ injective matching morphisms but restrict themselves to injective rules. The framework is liberal with respect to attributes in rules but requires extra conditions besides the usual gluing condition for matching morphisms. [1] does not contain correctness proofs for the indicated constructions of pushouts and pushout complements.

References

1. Michael R. Berthold, Ingrid Fischer, and Manuel Koch. Attributed graph transformation with partial attribution. In *Proc. Joint APPLIGRAPH/GETGRATS Worshop on Graph Transformation Systems (GRATRA 2000)*, pages 171–178, 2000.
2. Andrea Corradini, Ugo Montanari, Francesca Rossi, Hartmut Ehrig, Reiko Heckel, and Michael 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*, volume 1, chapter 3, pages 163–245. World Scientific, 1997.
3. Hartmut Ehrig. Introduction to the algebraic theory of graph grammars. In *Proc. Graph-Grammars and Their Application to Computer Science and Biology*, volume 73 of *Lecture Notes in Computer Science*, pages 1–69. Springer-Verlag, 1979.
4. Hartmut Ehrig, Gregor Engels, Hans-Jörg Kreowski, and Grzegorz Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 2: Applications, Languages, and Tools.* World Scientific, 1999.
5. Hartmut Ehrig and Hans-Jörg Kreowski. Pushout-properties: An analysis of gluing constructions for graphs. *Mathematische Nachrichten*, 91:135–149, 1979.
6. Hartmut Ehrig, Hans-Jörg Kreowski, Andrea Maggiolo-Schettini, Barry K. Rosen, and Jozef Winkowski. Transformations of structures: an algebraic approach. *Mathematical Systems Theory*, 14:305–334, 1981.

7. Hartmut Ehrig, Hans-Jörg Kreowski, Ugo Montanari, and Grzegorz Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 3: Concurrency, Parallelism, and Distribution.* World Scientific, 1999.
8. Annegret Habel, Jürgen Müller, and Detlef Plump. Double-pushout graph transformation revisited. *Mathematical Structures in Computer Science*, 11(5):637–688, 2001.
9. Francesco Parisi-Presicce, Hartmut Ehrig, and Ugo Montanari. Graph rewriting with unification and composition. In *Proc. Graph-Grammars and Their Application to Computer Science*, volume 291 of *Lecture Notes in Computer Science*, pages 496–514. Springer-Verlag, 1987.
10. Detlef Plump. Term graph rewriting. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 2, chapter 1, pages 3–61. World Scientific, 1999.
11. Barry K. Rosen. Deriving graphs from graphs by applying a production. *Acta Informatica*, 4:337–357, 1975.
12. Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations.* World Scientific, 1997.