Attributed Graph Transformation via Rule Schemata: Church-Rosser Theorem

Ivaylo Hristakiev* and Detlef Plump

The University of York, United Kingdom

Abstract. We present an approach to attributed graph transformation which requires neither infinite graphs containing data algebras nor auxiliary edges that link graph items with their attributes. Instead, we use the double-pushout approach on partially labelled graphs and extend it with rule schemata which are instantiated to ordinary rules prior to application. This framework provides the formal basis for the graph programming language GP 2. In this paper, we abstract from the data algebra of GP 2, define parallel independence of rule schema applications, and prove the Church-Rosser Theorem for our approach. The proof relies on the Church-Rosser Theorem for partially labelled graphs and adapts the classical proof by Ehrig and Kreowski, bypassing the technicalities of adhesive categories.

1 Introduction

Traditionally, the theory of graph transformation assumed that labels in graphs do not change in derivations (see, for example, [1]). But in applications of graph transformation it is often necessary to compute with labels. For instance, finding shortest paths in a graph whose edges are labelled with distances requires to determine the shorter of two distances and to add distances.

Graphs in which data elements of some fixed algebra are attached to nodes and edges have been called attributed graphs since [11], the first formal approach to extend graph transformation with computations on labels. In that paper, graphs are encoded as algebras to treat graph structure and algebra data uniformly. With a similar intention, the papers [8,5] go the other way round and encode the data algebra in graphs. Each data element becomes a special data node and auxiliary edges connect ordinary nodes and edges with the data nodes.

The latter approach has become mainstream but has some serious drawbacks (bemoaned as the *akwardness of attributes* in [16]). Firstly, the way attributes are attached to edges leads to the situation of edges having other edges as sources. This requires non-standard graphs and makes the model unusual. Secondly, and more importantly, there is typically an infinite number of data nodes because standard data algebras (such as integers or lists) have infinite domains. This

* The first author is supported by a grant from the Engineering and Physical Sciences Research Council (EPSRC) in the UK.

means that attributed graphs are usually infinite, leading to a discrepancy between theory and practice as graphs are stored using finite representations. In the approach of [5], even rules are normally infinite because they consist of graphs containing the complete term algebra corresponding to the data algebra.

In this paper, we propose an alternative approach to attributed graph transformation which avoids both infinite graphs and auxiliary attribute edges. Instead of merging graphs with the data algebra, we keep them separate. Host graph items simply get labelled with data elements and rule graph items get labelled with terms. To make this work, rules are instantiated by replacing terms with corresponding data values and then applied as usual. Hence our rules are actually *rule schemata* whose application can be seen as a two-stage process.

In order to modify attributes, it is crucial that interface items in rules can be relabelled. We therefore use the double-pushout approach with partially labelled interface graphs as a formal basis [6]. This approach is also the foundation of the graph programming language GP 2 [14]. The fixed data algebra of GP 2 consists of integers, character strings, and heterogeneous lists of strings and integers. In this paper, we abstract from this particular algebra and consider an arbitrary data algebra (see Subsection 2.2).

In Section 3, we define parallel independence of rule schema applications and prove the so-called Church-Rosser Theorem for our setting. Roughly, this result establishes that independent rule schema applications can be interchanged and result in the same graph. Our proof nicely decomposes into the Church-Rosser Theorem for the double-pushout approach with relabelling plus a simple extension to rule schemata (see Subsection 3.2).

The Church-Rosser Theorem for the relabelling setting was obtained in [7] as a corollary of an abstract result for $\mathcal{M},\mathcal{N}$-adhesive transformation systems. However, we deliberately avoid the categorical machinery of adhesiveness, van Kampen squares, etc. which we believe is difficult to digest for an average reader. Instead, we merely adapt the classical proof of Ehrig and Kreowski [4] to partially labelled graphs, essentially by replacing properties of pushouts and pullbacks in the unlabelled case by properties of natural pushouts in the setting of partially labelled graphs. (A pushout is natural if it is also a pullback.)

The rest of this paper is organized as follows. In Section 2, we describe the general idea of our approach. In Section 3, we present the notions of parallel and sequential independence and formalize the Church-Rosser Theorem at the rule schema level. Section 4 contains the relevant proofs. A conclusion and future work are given in Section 6.

We assume the reader to be familiar with basic notions of the double-pushout approach to graph transformation (see [2]). An extended version of this paper, along with complete proofs, can be found in [9].

2 Attributed Graph Transformation via Rule Schemata

In this section, we present our approach to transforming labelled graphs by rule schemata. We begin by briefly reviewing labelled graphs and the double-pushout approach to graph transformation with relabelling (see [6] for details).

2.1 The Double-Pushout Approach with Relabelling

A *partially labelled graph* G over a label alphabet $\mathcal{L} = \langle \mathcal{L}_V, \mathcal{L}_E \rangle$ consists of finite sets V_G and E_G of *nodes* and *edges*, *source* and *target* functions $s_G, t_G \colon E_G \to V_G$, a partial node labelling function $l_{G,V} \colon V_G \to \mathcal{L}_V$, and a partial edge labelling function $l_{G,E} \colon E_G \to \mathcal{L}_E$.

Given a node or edge x, $l_G(x) = \perp$ expresses that $l_G(x)$ is undefined [1]. Graph G is *totally labelled* if $l_{G,V}$ and $l_{G,E}$ are total functions. The classes of partially and totally labelled graphs are denoted by $\mathbf{Graph}_\perp(\mathcal{L})$ and $\mathbf{Graph}(\mathcal{L})$.

A *premorphism* $g \colon G \to H$ consists of two functions $g_V \colon V_G \to V_H$ and $g_E \colon E_G \to E_H$ that preserve sources and targets. A *graph morphism* g is a premorphism that preserves labels of nodes and edges, that is $l_H\big(g(x)\big) = l_G(x)$ for all $x \in Dom(l_G)$. A morphism g *preserves undefinedness* if it maps unlabelled items of G to unlabelled items in H. Morphism g is an *inclusion* if $g(x) = x$ for all items x in G. Note that inclusions need not preserve undefinedness. 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.

Partially labelled graphs and label-preserving morphism constitute a category $\mathbf{Graph}_\perp$. This category has been shown to be $\mathcal{M},\mathcal{N}$-adhesive [7] if one picks $\mathcal{M}$ to be the injective morphisms and $\mathcal{N}$ to be the injective morphisms that preserve undefinedness. (In double-pushouts, the horizontal and vertical morphisms come from $\mathcal{M}$ and $\mathcal{N}$, respectively.) What is special about this category is that pushouts need not always exist, and not all pushouts along injective morphisms are natural. These facts can be observed in Figure 2.

A *rule* $r = \langle L \leftarrow K \rightarrow R \rangle$ over an alphabet $\mathcal{L}$ consists of two inclusions $K \to L$ and $K \to R$ such that L, R are graphs in $\mathbf{Graph}(\mathcal{L})$ and K is a graph in $\mathbf{Graph}_\perp(\mathcal{L})$.

Definition 1 (Direct derivation). A *direct derivation* between graphs G and H via a rule $r = \langle L \leftarrow K \rightarrow R \rangle$ consists of two *natural pushouts* [2] as in Figure 1, where $g \colon L \to G$ is injective.

Operationally, the application of r to G proceeds as follows: 1) Match the left-hand graph L of r with a subgraph of G by means of an injective morphism $g \colon L \to G$ satisfying the *dangling condition*: no node in $g(L) - g(K)$ is incident to an edge in $G - g(L)$; 2) obtain a subgraph D by removing all items in $g(L) - g(K)$ and making each item $g(x)$ unlabelled for which x in K is unlabelled; 3) add

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

[2] A pushout is *natural* if it is also a pullback.

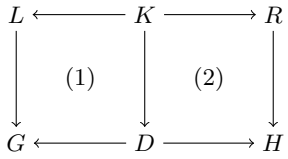


Fig. 1: A direct derivation

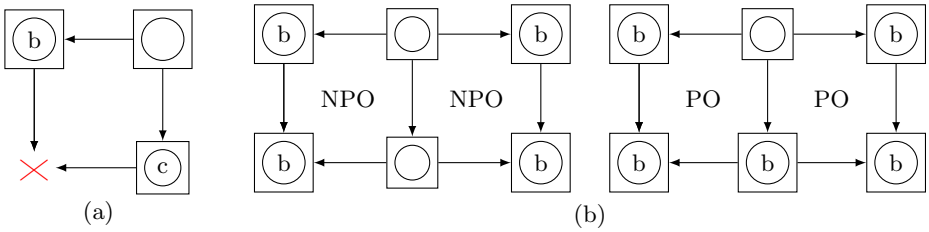


Fig. 2: (a) Pushouts need not exist. (b) A natural and a non-natural double pushout.

disjointly to D all items from $R - K$, keeping their labels, to obtain a graph H; 4) for each unlabelled item x in K, $l_H(g(x))$ becomes $l_R(x)$.

We write $G \Rightarrow_{r,g} H$ if there is a direct derivation from G to H. In [6] it is shown that if G is totally labelled, then the resulting graph H is also totally labelled. Moreover, in case the interface graph K has unlabelled items, their images in the intermediate graph D are also unlabelled by the condition that the pushouts are natural. Given a rule r and a graph G together with an injective match $g\colon L \to G$ satisfying the dangling condition, there exists a unique double natural pushout as in Figure 1 [6, Theorem 1].

2.2 Rule Schemata

Rule schemata were first introduced in the context of the graph programming language GP [15]. We review the basic notions of *signatures* and *algebras* (for details, see for example [2, Appendix B]).

Consider a *signature* Σ consisting of a set S of *sorts* and a family of *operation symbols* $OP = (OP_{w,s})_{(w,s)\in S^*\times S}$. A Σ*-algebra* A consists of a family of carrier sets $(A_s)_{s\in S}$ containing data values, and a set of functions implementing the operations of Σ. Moreover, *terms* built up from constants and variables are formally represented as a term algebra $T_\Sigma(X)$ where X is a family of variables that is disjoint from OP.

An assignment $\alpha\colon X \to A$ is a family of mappings $(\alpha_s\colon X_s \to A_s)_{s\in S}$ and gives values to each variable in a given set X. The extension $\alpha^*\colon T_\Sigma(X) \to A$

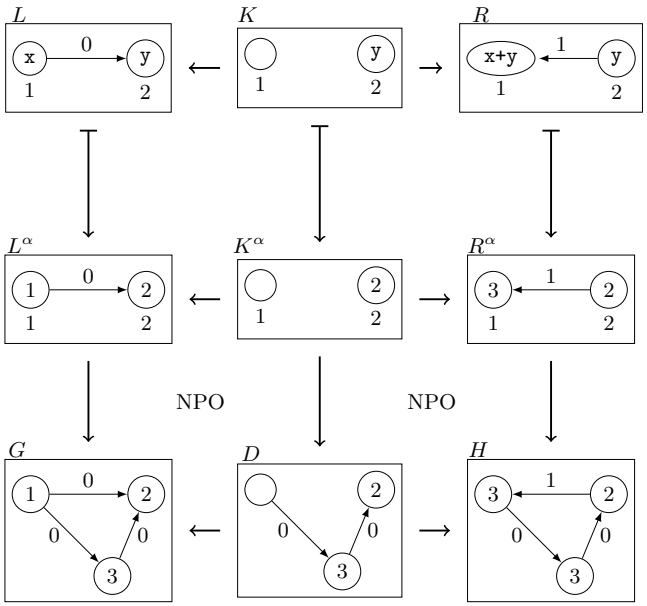


Fig. 3: Example of rule schema direct derivation

allows for the evaluation of terms in the given algebra A according to α. We assume a fixed Σ-algebra A whose elements are used as host graph labels, and a corresponding term algebra $T_\Sigma(X)$ whose terms are used as labels in rule schemata.

Definition 2 (Rule schema). A *rule schema* $r = \langle L \leftarrow K \rightarrow R \rangle$ consists of two inclusions $K \rightarrow L$ and $K \rightarrow R$ such that L, K, R are labelled over $T_\Sigma(X)$, where L and R are totally labelled.

To apply a rule schema r to a graph, the schema is first *instantiated* by evaluating its labels according to some assignment $\alpha \colon \mathcal{V}ar(r) \rightarrow A$ (where $\mathcal{V}ar(r)$ denotes the variables occurring in r).

Definition 3 (Rule schema instance and direct derivation). Given a graph G and an assignment $\alpha \colon X \rightarrow A$, the instance G^α of G is the graph obtained from G by replacing the labelling functions l_G with $\alpha^* \circ l_G$. The instance of a rule schema $r = \langle L \leftarrow K \rightarrow R \rangle$ is the rule $r^\alpha = \langle L^\alpha \leftarrow K^\alpha \rightarrow R^\alpha \rangle$.

Given a rule schema r, an injective match g and assignment $\alpha \colon \mathcal{V}ar(r) \rightarrow A$, a *rule schema direct derivation* between two graphs G and H is a direct derivation $G \Rightarrow_{r^\alpha, g} H$ via the schema instance r^α according to Definition 1

We write $G \Rightarrow_{r,g} H$ if there exists a direct derivation from G to H with rule schema r and match g. Note that we use $\Rightarrow$ for the application of both rule schemata and rules, to avoid an inflation of symbols. Figure 3 shows an example

of a rule schema direct derivation. We assume an algebra containing the integers with addition (+). The variables `x` and `y` are of sort `int` and the associated assignment α satisfies $\alpha(\mathtt{x}) = 1, \alpha(\mathtt{y}) = 2$. This allows for the relabelling of node 1 to 3. In general, a rule schema will have infinitely many instances if the algebra A has infinite carrier sets.

Given a rule schema and an injective premorphism $g\colon L \to G$ satisfying the dangling condition, it can be shown that a direct derivation $G \Rightarrow_r H$ can be constructed if an assignment α exists such that identical terms in L are mapped by g to identical labels in G. A similar proposition regarding *deterministic* rule schemata was presented in [15].

Proposition 1 (Construction of direct derivations). *Given a rule schema $r = \langle L \leftarrow K \rightarrow R \rangle$, an injective premorphism $g\colon L \to G$ satisfying the dangling condition and an assignment $\alpha\colon \mathcal{V}ar(r) \to A$ such that identical terms in L are mapped by g to identical labels in G, then the natural pushout complement D and the pushout H in Figure 3a exist and are unique up to isomorphism. Moreover, if G is totally labelled then H is also totally labelled.*

Proof. The rule schema r and assignment α uniquely determine a rule schema instance r^α according to Definition 3 where the graphs L^α and R^α are totally labelled. Since the dangling condition is not concerned with labels, the premorphism g induces a unique graph morphism $g' : L^\alpha \to G$ that satisfies the dangling condition with respect to r^α. Therefore the graph D is unique (up to isomorphism) according to [6, Lemma 4], and H is also unique since it is constructed via pushout. Totality of labelling follows directly from [6, Theorem 2]. □

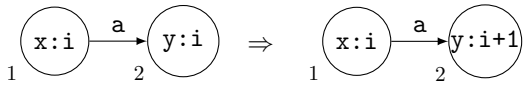
For complexity reasons, it is prohibitive for an implementation to try and enumerate all possible instances of a rule schema $r = \langle L \leftarrow K \rightarrow R \rangle$ in order to find a suitable match $g\colon L \to G$. This is because r may have an infinite set of instances - consider a premorphism that matches a node labelled with $x + y$ to a node labelled with the value 3. This induces an infinite number of assignments for the variables x, y via the equation $x + y = 3$ and the infinite domain of integers. A language such as GP 2 restricts the shape of labels appearing in rule schemata to avoid this problem. However, in this paper we are not concerned with implementation issues.

Example 1 (GP 2 rule schema). The graph programming language GP 2 [14] models structure in labels using heterogeneous lists over strings and integers. Given lists `x` and `y`, their concatenation is denoted by `x:y`. Rule schema terms in L are syntactically restricted by forbidding arithmetic operators (except unary minus). Also, there must be no repeated list variables in the same label, and $\mathcal{V}ar(R) \subseteq \mathcal{V}ar(L)$ must be satisfied, meaning that no fresh variables are created. Then for a given injective premorphism $g\colon L \to G$ there exists at most one assignment for the variables of r and thus at most one schema instance r'.

The GP 2 rule schema `inc` is presented below. It declares several typed variables which are assigned to concrete values during graph matching. Only the

left- and right-hand graphs are displayed, the common nodes are the interface nodes. The schema increments the last element in the label of node 2.

```
inc(a,x,y:list; i:int)
```



Abstracting from GP 2 and its built-in lists, other possible data types for labels include (multi)sets, stacks, queues and records.

3 Church-Rosser Theorem

In this section, we present the notion of parallel independence for direct derivations with relabelling and then extend it to applications of rule schemata.

3.1 Independence of Direct Derivations with Relabelling

Let the diagram in Figure 4 represent two direct derivations according to Definition 1.

Definition 4 (Independence). Two direct derivations $H_1 \Leftarrow_{r_1,m_1} G \Rightarrow_{r_2,m_2} H_2$ are *parallel independent* if there exist morphisms $i : L_1 \to D_2$ and $j : L_2 \to D_1$ such that $f_2 \circ i = m_1$ and $f_1 \circ j = m_2$

Two direct derivations $G \Rightarrow_{r_1,m_1} H_1 \Rightarrow_{r_2,m_2} H_2$ are *sequentially independent* if there exist morphisms $i : R_1 \to D_2$ and $j : L_2 \to D_1$ such that $f_2 \circ i = m'_1$ and $f_1 \circ j = m_2$.

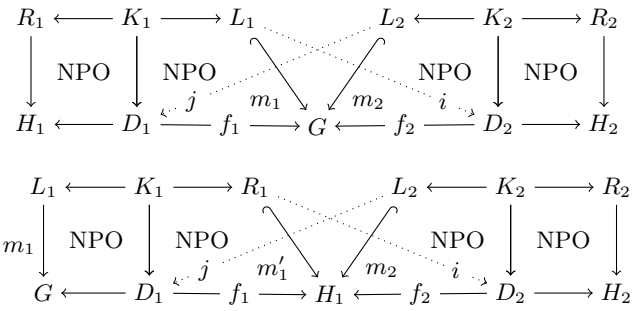


Fig. 4: Parallel (a) and Sequential (b) Independence.

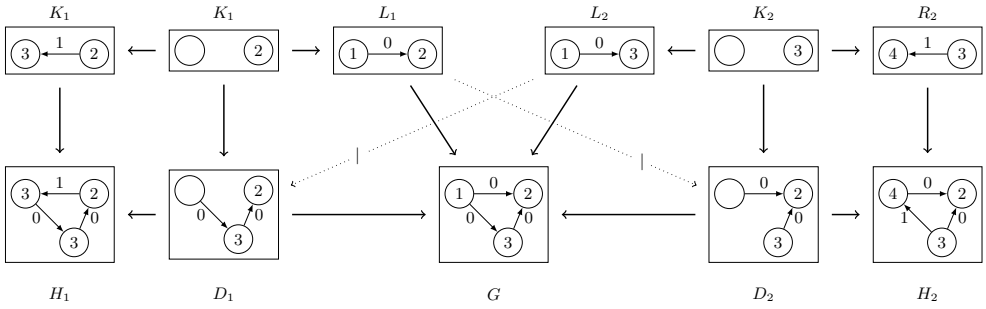


Fig. 5: Parallel Independence Example

Parallel independence is a sufficient condition for joining direct derivations starting from the same graph, while parallel dependence is a prerequiste for the definition of critical pairs [13].

Note that parallel and sequential independence are related - two derivations $H_1 \Leftarrow_{r_1,m_1} G \Rightarrow_{r_2,m_2} H_2$ are parallel independent if and only if the direct derivations $H_1 \Rightarrow_{r_1^{-1},m_1'} G \Rightarrow_{r_2,m_2} H_2$ are sequentially independent, where r_1^{-1} denotes the inverse rule of r_1 with m_1' as the comatch of m_1.

Lemma 1 (Characterization of parallel independence). *Two direct derivations $H_1 \Leftarrow_{r_1,m_1} G \Rightarrow_{r_2,m_2} H_2$ are parallel independent if for all items $x_1 \in L_1$ and $x_2 \in L_2$ such that $m_1(x_1) = m_2(x_2)$*

- *$x_1 \in K_1$ and $x_2 \in K_2$, and*
- *$l_{K_1}(x_1) \neq \bot$ and $l_{K_2}(x_2) \neq \bot$*

The first condition states that every common item is an interface item. The second condition states that no common item is relabelled by either derivation.

Example 2 (Parallel Independence). Figure 5 presents two direct derivations $H_1 \Leftarrow G \Rightarrow H_2$ that are instances of the rule schema shown in Figure 3b but applied in different ways to G. The derivations are not parallelly independent: there are no morphisms $L_1 \to D_2$ and $L_2 \to D_1$ with the wanted properties. The issue is that node 1 gets relabelled, breaking the second independence condition.

Our main result will show that, given two rule schema direct derivations that are parallel independent, there exists a joining derivation to a common result graph. First, we state the Church-Rosser Theorem for 'plain' rules in the sense of Definition 1. This has been shown in [7] as a corollary using the Church-Rosser Theorem for $\mathcal{M},\mathcal{N}$-Adhesive Transformation Systems. However, we obtain the result directly without using the notions of adhesiveness and van Kampen squares. Later, we obtain a Church-Rosser Theorem for *rule schema* derivations.

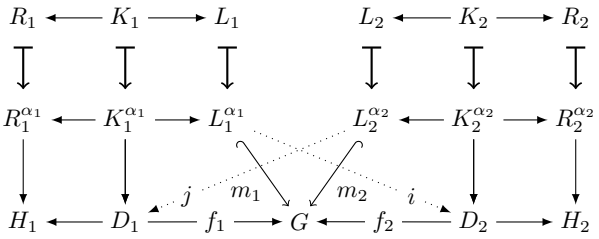
The proof follows the original Church-Rosser Theorem proof of [4]. At specific points it will be necessary to show that the results for NPO decomposition apply to the given setting. See Section 4 for the complete proof.

Theorem 1 (Church-Rosser Theorem). *Given two parallel independent direct derivations $G \Rightarrow_{r_1,m_1} H_1$ and $G \Rightarrow_{r_2,m_2} H_2$, there is a graph $\tilde{H}$ and direct derivations $H_1 \Rightarrow_{r_2,m_2'} \tilde{H}$ and $H_2 \Rightarrow_{r_1,m_1'} \tilde{H}$. Moreover $G \Rightarrow_{r_1,m_1} H_1 \Rightarrow_{r_2,m_2'} \tilde{H}$ as well as $G \Rightarrow_{r_2,m_2} H_2 \Rightarrow_{r_1,m_1'} \tilde{H}$ are sequentially independent.*

3.2 Church-Rosser Theorem for Rule Schema Derivations

This part lifts the previous independence result to *rule schema* applications. The main idea is to add instantiation on top of plain direct derivations.

Definition 5 (Parallel Independence of Rule Schema Derivations). Two rule schema direct derivations $G \Rightarrow_{r_1,m_1,\alpha_1} H_1$ and $G \Rightarrow_{r_2,m_2,\alpha_2} H_2$ are *parallel independent* if the plain derivations with relabelling $G \Rightarrow_{r_1^{\alpha_1},m_1} H_1$ and $G \Rightarrow_{r_2^{\alpha_2},m_2} H_2$ are parallel independent according to Definition 4.



The above notion can be lifted from the level of schema derivations to the level of schemata in the usual way by requiring that all instances between two schemata are parallel independent. However, as already noted, this may not always be possible to check as rule schemata may give rise to infinitely many instances if the underlying algebra has infinite carrier sets (and thus an infinite number of possible assignments $\alpha : \mathcal{V}ar(r) \to A$).

Theorem 2 (Church-Rosser Theorem for Rule Schemata). *Given two parallel independent rule schema direct derivations $G \Rightarrow_{r_1,m_1} H_1$ and $G \Rightarrow_{r_2,m_2} H_2$, there is a graph $\tilde{H}$ and rule schema direct derivations $H_1 \Rightarrow_{r_2,m_2'} \tilde{H}$ and $H_2 \Rightarrow_{r_1,m_1'} \tilde{H}$. Moreover $G \Rightarrow_{r_1,m_1} H_1 \Rightarrow_{r_2,m_2'} \tilde{H}$ as well as $G \Rightarrow_{r_2,m_2} H_2 \Rightarrow_{r_1,m_1'} \tilde{H}$ are sequentially independent.*

Proof. From Theorem 1, we know that independence of the plain derivations with relabelling $G \Rightarrow_{r_1^{\alpha_1},m_1} H_1$ and $G \Rightarrow_{r_2^{\alpha_2},m_2} H_2$ implies the existence of a graph $\tilde{H}$ and direct derivations $H_1 \Rightarrow_{r_2^{\alpha_2},m_2} \tilde{H}$ and $H_2 \Rightarrow_{r_1^{\alpha_1},m_1} \tilde{H}$.

This is illustrated in Figure 6.

We have that $G \Rightarrow_{r_1^{\alpha_1},m_1} H_1$ and $G \Rightarrow_{r_2^{\alpha_2},m_2} H_2$ are instances of the rule schemata r_1 and r_2, and therefore there are rule schema direct derivations $H_1 \Rightarrow_{r_2,m_2'} \tilde{H}$ and $H_2 \Rightarrow_{r_1,m_1'} \tilde{H}$. Again from Theorem 1 it follows the sequences of derivations are sequentially independent.

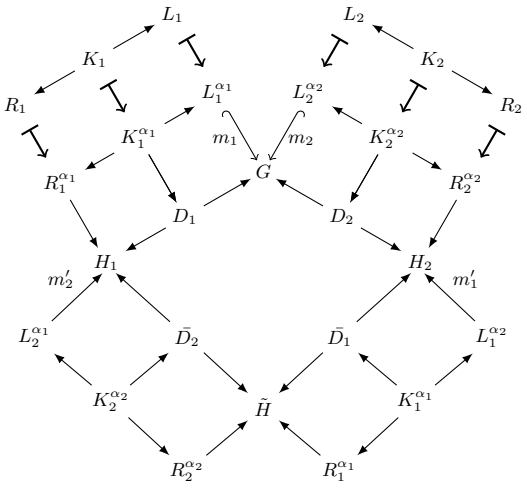
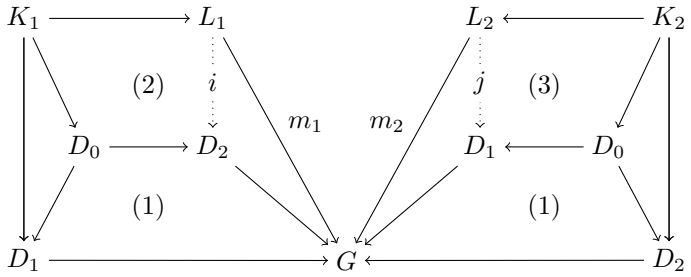


Fig. 6: Church-Rosser Theorem for rule schemata

4 Proof of Theorem 1

The proof follows the original Church-Rosser Theorem proof of [4]. At specific points it will be necessary to show that the results for NPO decomposition apply to the given setting. This is because for partially labelled graphs ($\mathbf{Graph}_\perp$), pushouts need not always exist, and not all pushouts along injective morphisms are natural. These facts have been observed in Figure 2.

Using the definition of parallel independence (Definition 4), we start by decomposing the derivations in the following diagram.



The graph D_0 is obtained as a pullback of $(D_1 \to G \leftarrow D_2)$. The universal property of pullbacks gives us that $K_1 \to D_1$ and $K_2 \to D_2$ decompose into $K_1 \to D_0 \to D_1$ and $K_2 \to D_0 \to D_2$ respectively. We also have that $(1+2)$ and $(1+3)$ are NPOs because they are left-hand sides of derivations. Furthermore, $D_1 \to G$ and $D_2 \to G$ are injective and jointly surjective which makes (1) a NPO ([9, Lemma 4]).

$D_1 \to G$ and $D_2 \to G$ injective imply that $D_0 \to D_2$ and $D_0 \to D_1$ are also injective. The subsequent parts of the proof contain four claims which are proven afterwards.

Claim 1. The squares (2) and (3) are NPOs.

Next, the pushouts $\overline{D_1}$ of $(D_0 \leftarrow K_1 \rightarrow R_1)$ (5) and $\overline{D_2}$ of $(D_0 \leftarrow K_2 \rightarrow R_2)$ (6) are constructed. These exist by the following claim:

Claim 2. In Figure 7, the pushouts $\overline{D_1}$ of $(D_0 \leftarrow K_1 \rightarrow R_1)$ (5) and $\overline{D_2}$ of $(D_0 \leftarrow K_2 \rightarrow R_2)$ (6) exist.

Again using uniqueness, the morphisms $R_1 \to H_1$ and $R_2 \to H_2$ decompose into $R_1 \to \overline{D_2} \to H_1$ and $R_2 \to \overline{D_1} \to H_2$. We also have that $(5+7)$ and $(6+8)$ are NPOs because they are right-hand sides of derivations.

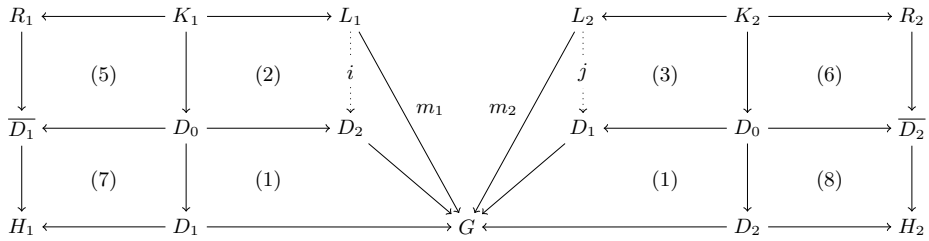


Fig. 7: Church-Rosser decomposition diagram.

Also, (7) and (8) become NPOs by the NPO Decomposition Lemma [9, Lemma 5.3].

Claim 3. In Figure 7, the squares (7) and (8) are NPOs.

The graph $\tilde{H}$ is constructed as a pushout of $(\overline{D_1} \leftarrow D_0 \rightarrow \overline{D_2})$ (4). (See square (4) in Figure 8.)

Claim 4. The pushout of $(\overline{D_1} \leftarrow D_0 \rightarrow \overline{D_2})$ exists.

This pushout becomes NPO by [9, Lemma 3] and the arguments in the proof of Claim 4. Furthermore, the graph $\tilde{H}$ is totally labelled due to the way D_0, $\overline{D_1}$ and $\overline{D_2}$ are constructed - $\overline{D_1}$ can contain unlabelled items only from $D_0 - K_1$ which are labelled in $\overline{D_2}$, and vice versa.

The pushouts can be rearranged (Figure 8) to show that $G \Rightarrow_{r_1,m_1} H_1 \Rightarrow_{r_2,m_2'} \tilde{H}$ as well as $G \Rightarrow_{r_2,m_2} H_2 \Rightarrow_{r_1,m_1'} \tilde{H}$ are sequentially independent. Note that the graph $\tilde{H}$ is totally labelled.

This concludes the proof of the Church-Rosser Theorem. □

Next, we present the proofs of the above claims.

Proof of Claim 1 We need to show that the conditions of the NPO Decomposition Lemma [9, Lemma 5.2] hold for the following diagrams.

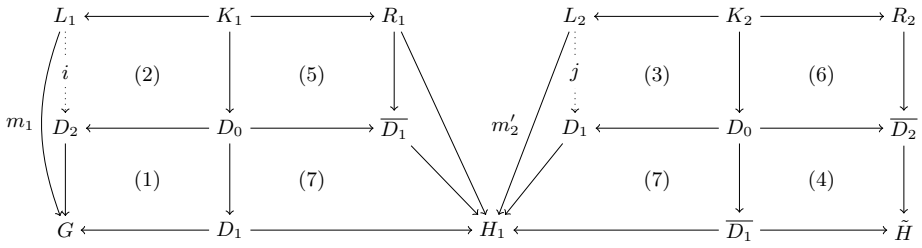
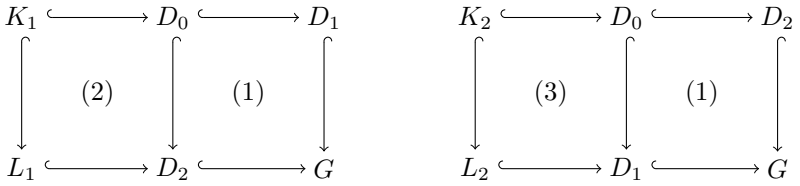


Fig. 8: Rearranged pushouts.



$D_0 \to D_1$ is injective because $D_2 \to G$ is injective by definition and (1) is PB. (1) has already been proven to be NPO (at the start of this section). Pushout exists over $(L_1 \leftarrow K_1 \to D_0)$ as L_1 is totally labelled, both morphisms are injective and $K_1 \to D_0$ preserves undefinedness, all by the definition of direct derivation with relabelling. The square $K_1 L_1 D_0 D_2$ commutes because $(1 + 2)$ and (1) are NPOs. Therefore, all conditions of the NPO Decomposition Lemma [9, Lemma 5.2] hold.

The proof for the second diagram is analogous.

This concludes the proof that the squares (2) and (3) are NPOs. □

Proof of Claim 2 As in the previous proof, R_1 and R_2 are totally labelled, all morphisms are injective and both $K_1 \to D_0$ and $K_2 \to D_0$ preserve undefinedness, all by the definition of direct derivation with relabelling. Therefore, the pushouts (5) and (6) exist by [9, Lemma 2.2]. □

Proof of Claim 3 In the context of Figure 9, we need to show that the conditions of the NPO Decomposition Lemma [9, Lemma 5.3] hold.

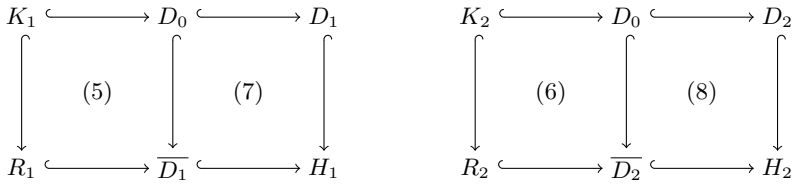
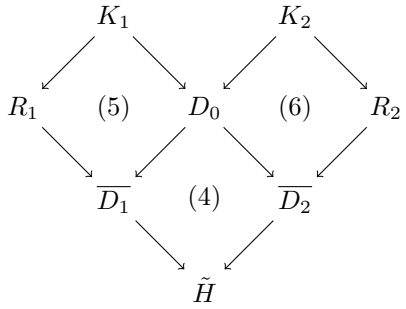


Fig. 9: Pushouts (5), (6), (7) and (8).

$D_0 \to D_1$ has already been established as injective. We need that there exists unique NPO complement of $K_1 \to R_1 \to \overline{D_1}$. We have that $K_1 \to R_1$ is injective by definition. $R_1 \to \overline{D_1}$ is injective because $K_1 \to D_0$ and $L_1 \to D_2$ are injective. The pushout (5+7) is a right-hand side of a derivation and $R_1 \to H_1 = R_1 \to \overline{D_1} \to H_1$. Consequently, $R_1 \to \overline{D_1}$ satisfies the dangling condition w.r.t.

$K_1 \to R_1$, thus the existence of a unique NPO complement is given by [9, Lemma 2.3]. The proof for the second diagram is analogous.

This concludes the proof that the prerequisites for the NPO Decomposition Lemma [9, Lemma 5.3] hold. Hence, squares (7) and (8) become NPOs. □



Proof of Claim 4 For a pushout to exist, $\overline{D_1}$ and $\overline{D_2}$ have to agree on the labels of the unlabelled nodes of D_0 ([9, Lemma 2.2]).

$\overline{D_1}$ is constructed as the pushout of $(R_1 \leftarrow K_1 \to D_0)$ with R_1 being totally labelled. Its node and edge sets and labelling function is as defined in [9, Lemma 2.2]. Moreover, $R_1 \to \overline{D_1}$ and $D_0 \to \overline{D_1}$ are injective and jointly surjective.

There are 3 main cases for an item x to be labelled in $\overline{D_1}$:

- the item is created by the first derivation $x \in R_1 - K_1$. This means it does not exist in D_1, D_2 or G. Consequently, this item does not have a preimage in D_0 by pullback construction ([9, Lemma 2.1]). Therefore, it cannot be a source of conflict for pushout existence.
- the item is relabelled by the first derivation, meaning its preimage in D_1 (and D_0) is unlabelled $x \in K_1$ and $l_{K_1}(x) = l_{D_1}(x) = l_{D_0}(x) = \perp$. By the definition of parallel independence, no common items are relabelled making the item not have a preimage in R_2. Therefore it is unlabelled in $\overline{D_2}$ (by definition of pushout), making it a non-conflict w.r.t. pushout existence.
- the item is in $\overline{D_1} - R_1$, i.e. a labelled item of $D_0 - K_1$. We have that $D_0 \to \overline{D_1}$ and $D_0 \to \overline{D_2}$ are label preserving, so the label of x in $\overline{D_2}$ is the same as in $\overline{D_1}$.

In all cases, the labels of $\overline{D_1}$ are preserved by the second derivation. The argument for the labelled items of $\overline{D_2}$ is analogous.

This concludes the proof that the pushout (4) over $(\overline{D_1} \leftarrow D_0 \to \overline{D_2})$ exists. □

5 Related Work

In this paper we have adapted the classical Church-Rosser proof of Ehrig and Kreowski [4] to partially labelled graphs and extended the result to rule schemata, essentially by replacing properties of pushouts and pullbacks in the unlabelled case by properties of natural pushouts in the setting of partially labelled graphs.

In [5], the theory of attributed graph transformation is developed in the framework of so-called adhesive HLR categories. Among other results, the Church-Rosser Theorem is proved in this setting. The approach is further studied in [3] by adding nested application conditions and proving the previous results for this more expressive approach. Both are a generalized version of the Church-Rosser Theorem of [8].

So-called symbolic graphs are attributed graphs in which all data nodes are variables, combined with a first-order logic formula over these variables. In [12] it is shown that this approach can specify and transform classes of ordinary attributed graphs that satisfy the given formula. The underlying graph structure is the same as in [5], hence the approach shares the issues described in the introduction.

Recently, a generalised Church-Rosser Theorem for attributed graph transformation has been proved in [10] by using symbolic graphs. A notion of parallel independence is used that takes into account the semantics of attribute operations, in order to reduce the number of "false positives" in conflict checking.

6 Conclusion

In this paper, we have presented an approach to attributed graph transformation based on partially labelled graphs and rule schemata which are instantiated to ordinary rules prior to application. We have defined parallel independence of rule schema applications and proved the Church-Rosser Theorem for our approach. The proof relies on the Church-Rosser Theorem for partially labelled graphs and adapts the classical proof by Ehrig and Kreowski, bypassing the technicalities of adhesive categories.

Future work includes establishing other classical graph transformation results in our setting, such as embedding and restriction theorems. Furthermore, we aim at studying critical pairs and confluence both for the particular case of GP 2 and for attribute algebras in general. In particular, we plan to give a construction of critical pairs (labelled with expressions rather than concrete values) that guarantees the set of critical pairs is both finite and complete as opposed to being based on schema instances, the set of which is usually infinite. Completeness of critical pairs would mean all possible conflicts are an embedding of a critical pair computed by our algorithm, and thus it would be possible to automate critical pair analysis as part of a confluence checker.

References

1. Ehrig, H.: Introduction to the algebraic theory of graph grammars. In: Proc. Graph-Grammars and Their Application to Computer Science and Biology. Lecture Notes in Computer Science, vol. 73, pp. 1–69. Springer (1979)
2. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science, Springer (2006)

3. Ehrig, H., Golas, U., Habel, A., Lambers, L., Orejas, F.: $\mathcal{M}$-adhesive transformation systems with nested application conditions. Part 1: Parallelism, concurrency and amalgamation. Mathematical Structures in Computer Science 24(4) (2014)
4. Ehrig, H., Kreowski, H.J.: Parallelism of manipulations in multidimensional information structures. In: Proc. Mathematical Foundations of Computer Science. Lecture Notes in Computer Science, vol. 45, pp. 284–293. Springer (1976)
5. Ehrig, H., Prange, U., Taentzer, G.: Fundamental theory for typed attributed graph transformation. In: Proc. Graph Transformations (ICGT 2004). Lecture Notes in Computer Science, vol. 3256, pp. 161–177. Springer (2004)
6. Habel, A., Plump, D.: Relabelling in graph transformation. In: Proc. International Conference on Graph Transformation (ICGT 2002). Lecture Notes in Computer Science, vol. 2505, pp. 135–147. Springer (2002)
7. Habel, A., Plump, D.: $\mathcal{M},\mathcal{N}$-adhesive transformation systems. In: Proc. International Conference on Graph Transformation (ICGT 2012). Lecture Notes in Computer Science, vol. 7562, pp. 218–233. Springer (2012)
8. Heckel, R., Küster, J.M., Taentzer, G.: Confluence of typed attributed graph transformation systems. In: Proc. International Conference on Graph Transformation (ICGT 2002). LNCS, vol. 2505, pp. 161–176. Springer (2002)
9. Hristakiev, I., Plump, D.: Attributed graph transformation via rule schemata: Church-Rosser theorem (long version) (2016), `http://www.cs.york.ac.uk/plasma/publications/pdf/HristakievPlump16.Full.pdf`
10. Kulcsár, G., Deckwerth, F., Lochau, M., Varró, G., Schürr, A.: Improved conflict detection for graph transformation with attributes. In: Proc. Graphs as Models (GaM 2015). Electronic Proceedings in Theoretical Computer Science, vol. 181, pp. 97–112 (2015)
11. Löwe, M., Korff, M., Wagner, A.: An algebraic framework for the transformation of attributed graphs. In: Term Graph Rewriting: Theory and Practice, pp. 185–199. John Wiley (1993)
12. Orejas, F., Lambers, L.: Symbolic attributed graphs for attributed graph transformation. In: Graph and Model Transformation. Electronic Communications of the EASST, vol. 30 (2010)
13. Plump, D.: Confluence of graph transformation revisited. In: Processes, Terms and Cycles: Steps on the Road to Infinity: Essays Dedicated to Jan Willem Klop on the Occasion of His 60th Birthday, Lecture Notes in Computer Science, vol. 3838, pp. 280–308. Springer (2005)
14. Plump, D.: The design of GP 2. In: Proc. Workshop on Reduction Strategies in Rewriting and Programming (WRS 2011). Electronic Proceedings in Theoretical Computer Science, vol. 82, pp. 1–16 (2012)
15. Plump, D., Steinert, S.: Towards graph programs for graph algorithms. In: Proc. International Conference on Graph Transformation (ICGT 2004). Lecture Notes in Computer Science, vol. 3256, pp. 128–143. Springer (2004)
16. Rensink, A.: The edge of graph transformation - graphs for behavioural specification. In: Graph Transformations and Model-Driven Engineering - Essays Dedicated to Manfred Nagl on the Occasion of His 65th Birthday, Lecture Notes in Computer Science, vol. 5765, pp. 6–32. Springer (2010)