

Confluence of Graph Transformation Revisited

Detlef Plump

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

Abstract. It is shown that it is undecidable in general whether a terminating graph rewriting system is confluent or not—in contrast to the situation for term and string rewriting systems. Critical pairs are introduced to hypergraph rewriting, a generalisation of graph rewriting, where it turns out that the mere existence of common reducts for all critical pairs of a graph rewriting system does not imply local confluence. A Critical Pair Lemma for hypergraph rewriting is then established which guarantees local confluence if each critical pair of a system has joining derivations that are compatible in that they map certain nodes to the same nodes in the common reduct.

1 Introduction

To compute efficiently with graph transformation rules requires some way of cutting down the nondeterminism in the derivation spaces of graphs. A common solution to this problem in rule-based formalisms is to rely on *confluent* sets of rules so that all terminating derivations from an initial state will yield the same result, making backtracking unnecessary. For example, confluence properties are important for efficiently recognizing graph classes and executing graph algorithms by graph reduction [2, 5, 7], for parsing graph languages by using the inverse rules of graph grammars [16, 31], and for verifying the deterministic behaviour of programs in graph rewriting languages such as PROGRES [34], AGG [15] and GP [29].

In the setting of term rewriting systems, Knuth and Bendix [22] showed that confluence is decidable for terminating sets of rules. It suffices to compute all *critical pairs* $t \leftarrow s \rightarrow u$ of rewrite steps in which s is the superposition of the left-hand sides of two rules, and to check whether t and u reduce to a common term. This procedure is justified by the Critical Pair Lemma [20]—stating that a term rewriting system is locally confluent if and only if all its critical pairs have common reducts—and by Newman’s Lemma which asserts the equivalence of confluence and local confluence in the presence of termination.

In contrast to the situation for term and string rewriting, Theorem 5 below will show that confluence is undecidable for terminating graph rewriting systems. Roughly, the reason is that the embedding of derivations into context is more complicated for graphs than for terms and strings, meaning that the existence of common reducts for all critical pairs need not imply local confluence of the system.

The second major result of this paper is a Critical Pair Lemma for hypergraph rewriting which provides a sufficient condition for local confluence and hence for confluence of terminating systems (Theorem 7). The result requires each critical pair $T \leftarrow S \Rightarrow U$ to be joinable by two derivations $T \Rightarrow^* W_1 \cong W_2 \leftarrow^* U$ such that an isomorphism $W_1 \rightarrow W_2$ is compatible with the joining derivations, in that each node in S that is preserved by both $S \Rightarrow T$ and $S \Rightarrow U$ is mapped to the same node in W_2 by $S \Rightarrow^* W_2$ and $S \Rightarrow^* W_1$ followed by $W_1 \rightarrow W_2$.

The next section recalls some properties of binary relations and defines labelled and directed hypergraphs. Section 3 reviews the double-pushout approach to graph transformation, adapted to the setting of hypergraphs. Subsections 3.2 and 3.3 provide results about the restriction, extension and independence of derivations which will be needed to prove the Critical Pair Lemma. Section 4 starts by arguing, in Subsection 4.1, that confluence modulo isomorphism rather than confluence is the right notion to consider in graph transformation. Subsection 4.2 then presents a reduction of the Post Correspondence Problem showing that confluence is undecidable for terminating graph rewriting systems. Subsection 4.3 introduces critical pairs to hypergraph rewriting and proves the Critical Pair Lemma. Section 5 concludes by mentioning related work and topics for future work. Finally, the Appendix summarises some properties of hypergraph pushouts that are needed in proofs.

The confluence results of this paper were established in [27], but the undecidability of confluence was only shown for hypergraph rewriting; the current result also covers graph rewriting. In addition, the undecidability proof has been simplified so that the number of rule schemata in the reduction of the Post Correspondence Problem decreased from 21 to 12. Moreover, this paper is rigorous with respect to the role of confluence modulo isomorphism.

2 Preliminaries

This section fixes some terminology for binary relations (see also [3, 4]) and introduces hypergraphs and their morphisms.

2.1 Relations

Let \rightarrow be a binary relation on a set A . The inverse relation of \rightarrow is denoted by \leftarrow . The identity on A is the relation $\rightarrow^0 = \{ \langle a, a \rangle \mid a \in A \}$. The reflexive closure of \rightarrow is $\rightarrow^= = \rightarrow \cup \rightarrow^0$. The composition of two binary relations \rightarrow_1 and \rightarrow_2 on A is $\rightarrow_1 \circ \rightarrow_2 = \{ \langle a, c \rangle \mid a \rightarrow_1 b \text{ and } b \rightarrow_2 c \text{ for some } b \}$. For every $n > 0$, the n -fold composition of \rightarrow is $\rightarrow^n = \rightarrow \circ \rightarrow^{n-1}$. The transitive closure of \rightarrow is $\rightarrow^+ = \cup_{n>0} \rightarrow^n$, and the transitive-reflexive closure of \rightarrow is $\rightarrow^* = \rightarrow^+ \cup \rightarrow^0$. Two elements a and b have a *common reduct* if $a \rightarrow^* c \leftarrow^* b$ for some c . If $a \rightarrow^* c$ and there is no d such that $c \rightarrow d$, then d is a *normal form* of a .

Definition 1 (Termination and confluence). The relation \rightarrow is

- (1) *terminating* if there is no infinite sequence of the form $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \dots$,
- (2) *confluent* if for all a, b and c with $b \leftarrow^* a \rightarrow^* c$, elements b and c have a common reduct (see Figure 1(a)),

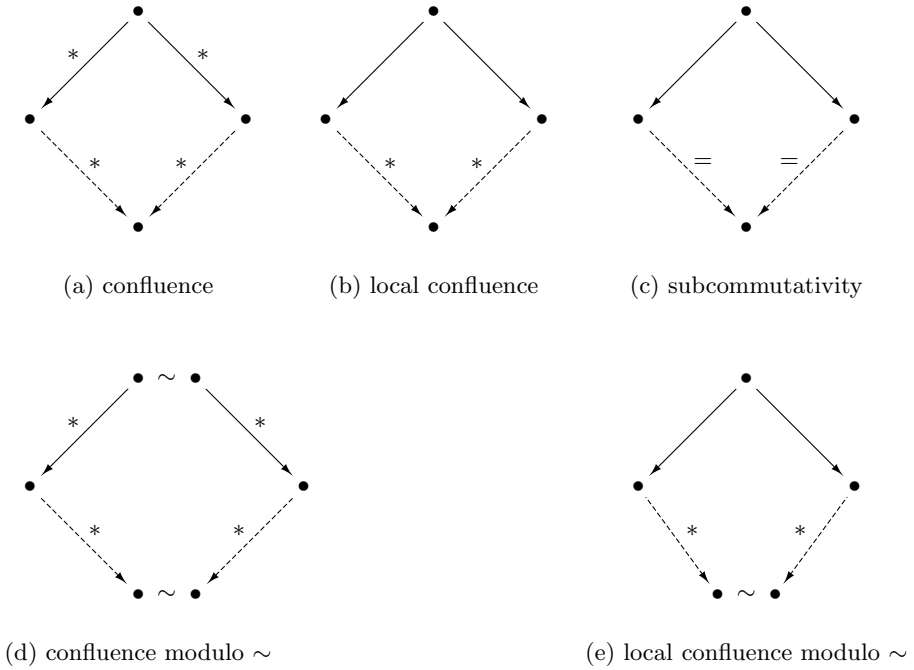


Fig. 1. Confluence properties

- (3) *locally confluent* if for all a, b and c with $b \leftarrow a \rightarrow c$, elements b and c have a common reduct (see Figure 1(b)),
- (4) *subcommutative* if for all a, b and c with $b \leftarrow a \rightarrow c$ there is some d such that $b \rightarrow^= d \leftarrow^= c$ (see Figure 1(c)),
- (5) *confluent modulo \sim* , where \sim is an equivalence relation on A , if for all a, a', b and c with $b \leftarrow^* a \sim a' \rightarrow^* c$ there are d and d' such that $b \rightarrow^* d \sim d' \leftarrow^* c$ (see Figure 1(d)),
- (6) *locally confluent modulo \sim* if for all a, b and c with $b \leftarrow a \rightarrow c$ there are d and d' such that $b \rightarrow^* d \sim d' \leftarrow^* c$ (see Figure 1(e)).

By the following well-known result [25], local confluence and confluence are equivalent in the presence of termination.

Lemma 1 (Newman’s Lemma). *A terminating relation is confluent if and only if it is locally confluent.*

2.2 Hypergraphs

This paper deals with directed hypergraphs in which nodes and hyperedges carry labels and where the label of a hyperedge can restrict both the number of incident nodes and their possible labels. A *signature* $\Sigma = \langle \Sigma_V, \Sigma_E \rangle$ is a pair of sets of *node labels* and *hyperedge labels* such that each $l \in \Sigma_E$ comes with a set $\text{Type}(l) \subseteq \Sigma_V^*$.

(Note the similarity to signatures of many-sorted algebras [17]: sorts correspond to node labels and operation symbols correspond to hyperedge labels.)

Definition 2 (Hypergraph). A *hypergraph* over a signature Σ is a system $G = \langle V_G, E_G, \text{mark}_G, \text{lab}_G, \text{att}_G \rangle$ consisting of two finite sets V_G and E_G of *nodes* (or *vertices*) and *hyperedges*, two labelling functions $\text{mark}_G: V_G \rightarrow \Sigma_V$ and $\text{lab}_G: E_G \rightarrow \Sigma_E$, and an attachment function $\text{att}_G: E_G \rightarrow V_G^*$ such that $\text{mark}_G^*(\text{att}_G(e)) \in \text{Type}(\text{lab}_G(e))$ for each hyperedge e .¹

Hyperedges are said to be *incident* to their attachment nodes. In pictures, nodes and hyperedges are drawn as circles and boxes, respectively, with labels inside. Lines represent the attachment of hyperedges to nodes. If a hyperedge is attached to more than two nodes, the lines are numbered according to the left-to-right order in the attachment string. For hyperedges with just two attachment nodes—ordinary *edges*—, an arrowhead points to the second node; in this case the box may be omitted and the label written next to the arrow. As an example, Figure 2 shows a hypergraph borrowed from [17]. Here $\text{Type}(\text{PUSH})$, for instance, contains the string `data stack stack`.

A hypergraph G is a *graph* if each hyperedge e is an ordinary edge, that is, if the attachment sequence $\text{att}_G(e)$ has length two.

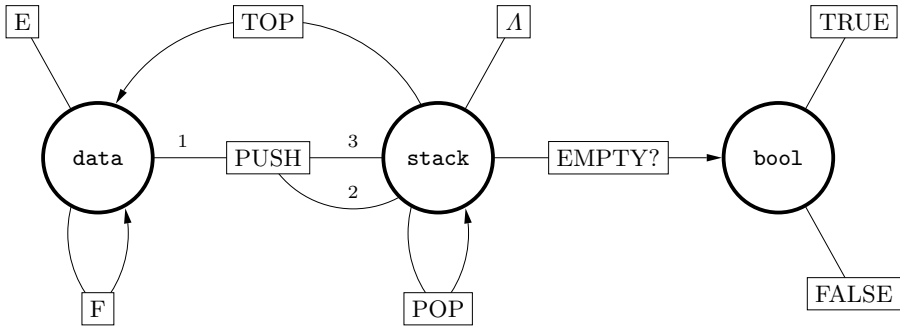


Fig. 2. A hypergraph

Definition 3 (Hypergraph morphism). Given two hypergraphs G and H over Σ , a *hypergraph morphism* $f: G \rightarrow H$ consists of two functions $f_V: V_G \rightarrow V_H$ and $f_E: E_G \rightarrow E_H$ that preserve labels and attachment to nodes, that is, $\text{mark}_H \circ f_V = \text{mark}_G$, $\text{lab}_H \circ f_E = \text{lab}_G$ and $\text{att}_H \circ f_E = f_V^* \circ \text{att}_G$.

A hypergraph morphism $\text{incl}: G \rightarrow H$ is an *inclusion* if $\text{incl}_V(v) = v$ and $\text{incl}_E(e) = e$ for all $v \in V_G$ and $e \in E_G$. In this case G is a *subhypergraph* of H which is denoted by $G \subseteq H$. Every hypergraph morphism $f: G \rightarrow H$

¹ The extension $f^*: A^* \rightarrow B^*$ of a function $f: A \rightarrow B$ maps the empty string to itself and $a_1 \dots a_n$ to $f(a_1) \dots f(a_n)$.

induces a subhypergraph of H , denoted by $f(G)$, which has nodes $f_V(V_G)$ and hyperedges $f_E(E_G)$. The *composition* of $f: G \rightarrow H$ with a morphism $g: H \rightarrow M$ is the hypergraph morphism $g \circ f: G \rightarrow M$ consisting of the composed functions $g_V \circ f_V$ and $g_E \circ f_E$. The composition is also written as $G \rightarrow H \rightarrow M$ if f and g are clear from the context.

The morphism f is *injective* (*surjective*) if f_V and f_E are injective (surjective). Injectivity of f may be indicated by writing $f: G \hookrightarrow H$. If f is both injective and surjective, then it is an *isomorphism*. In this case G and H are *isomorphic*, which is denoted by $G \cong H$.

3 Graph Transformation

This section reviews the *double-pushout approach* to graph transformation, where the approach presented in the overviews [9, 6] is generalized in three ways: hypergraphs rather than graphs are considered and rules are matched injectively and can have non-injective right-hand morphisms. Both injective matching and non-injective right-hand morphisms add expressiveness to the double-pushout approach [18].

3.1 Rules and Derivations

From now on an arbitrary but fixed signature is assumed over which all hypergraphs are labelled, unless signatures are explicitly mentioned.

Definition 4 (Rule). A rule $r: \langle L \hookleftarrow K \rightarrow^b R \rangle$ consists of three hypergraphs and two hypergraph morphisms, where $K \hookrightarrow L$ is an inclusion. The hypergraphs L and R are the *left-* and *right-hand side* of r , and K is the *interface*. The rule r is *injective* if $b: K \rightarrow R$ is injective.

Figure 3 shows a rule which removes two hyperedges and a node, merges two nodes, and creates a hyperedge. The letters x,y,z are node names; they are used to represent the hypergraph morphisms between the interface and the left- and right-hand side. The name $x=y$ indicates that the right-hand morphism identifies node x with node y . The lower half of Figure 3 shows the same rule in a shorthand notation. In this format, only the left- and right-hand sides are depicted while the interface is implicitly given by all the named nodes of the left-hand side.

Definition 5 (Direct derivation). Let G and H be hypergraphs, $r: \langle L \hookleftarrow K \rightarrow^b R \rangle$ a rule and $f: L \hookrightarrow G$ an injective hypergraph morphism. Then G *directly derives* H by r and f , denoted by $G \Rightarrow_{r,f} H$, if there exist two pushouts² of the following form:

² See the Appendix for the definition and construction of hypergraph pushouts.

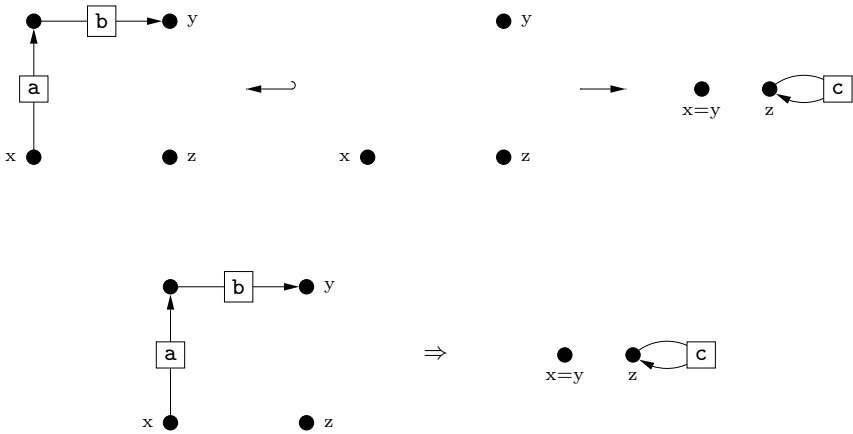


Fig. 3. A rule and its shorthand notation

$$\begin{array}{ccccc}
 L & \longleftarrow & K & \xrightarrow{b} & R \\
 f \downarrow & & \downarrow & & \downarrow \\
 G & \longleftarrow & D & \longrightarrow & H
 \end{array} \tag{1}$$

Intuitively, the left pushout corresponds to the replacement of L with K (equivalently, to the removal of L up to K) and the right to the replacement of K with R . As an example, Figure 4 shows an application of the rule of Figure 3.

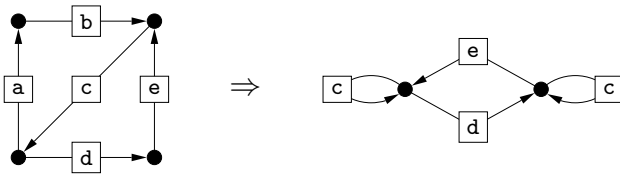


Fig. 4. An application of the rule of Figure 3

A double-pushout as in diagram (1) is called a *direct derivation* from G to H and may be denoted (by abuse of notation) by $G \Rightarrow_{r,f} H$ or just by $G \Rightarrow_r H$ or $G \Rightarrow H$. A *derivation* from G to H is a sequence of direct derivations $G = G_0 \Rightarrow \dots \Rightarrow G_n = H$ for some $n \geq 0$ and may be denoted by $G \Rightarrow^* H$.

Definition 6 (Dangling condition). Given a rule $r: \langle L \leftarrow K \rightarrow R \rangle$ and a hypergraph G , a hypergraph morphism $f: L \rightarrow G$ satisfies the *dangling condition* if no hyperedge in $E_G - f_E(E_L)$ is incident to a node in $f_V(V_L) - f_V(V_K)$.

In other words, the dangling condition guarantees that each attachment node of a hyperedge in $E_G - f_E(E_L)$ belongs to either $V_G - f_V(V_L)$ or $f_V(V_K)$.

With this condition the existence of a direct derivation can be characterized operationally.

Theorem 1 (Constructing a direct derivation [28]). *Let G and H be hypergraphs, $r: \langle L \hookrightarrow K \rightarrow^b R \rangle$ a rule and $f: L \hookrightarrow G$ an injective hypergraph morphism. Then $G \Rightarrow_{r,f} H$ if and only if f satisfies the dangling condition and H is isomorphic to the hypergraph M constructed as follows:*

- $V_M = (V_G - f_V(V_L)) + V_R$ and $E_M = (E_G - f_E(E_L)) + E_R$,³
- $\text{mark}_M(v) = \underline{\text{if}} \ v \in V_R \ \underline{\text{then}} \ \text{mark}_R(v) \ \underline{\text{else}} \ \text{mark}_G(v)$,
- $\text{lab}_M(e) = \underline{\text{if}} \ e \in E_R \ \underline{\text{then}} \ \text{lab}_R(e) \ \underline{\text{else}} \ \text{lab}_G(e)$, and
- $\text{att}_M(e) = \underline{\text{if}} \ e \in E_R \ \underline{\text{then}} \ \text{att}_R(e) \ \underline{\text{else}} \ t^*(\text{att}_G(e))$,

where the auxiliary function $t: (V_G - f_V(V_L)) \cup f_V(V_K) \rightarrow V_M$ is defined by $t(v) = \underline{\text{if}} \ v = f_V(v') \ \underline{\text{for some}} \ v' \in V_K \ \underline{\text{then}} \ b_V(v') \ \underline{\text{else}} \ v$.

With every derivation $\Delta: G_0 \Rightarrow^* G_n$ a partial hypergraph morphism⁴ can be associated that “follows” the items of G_0 through the derivation: this morphism is undefined for all items in G_0 that are removed by Δ , and maps all other items to the corresponding items in G_n .

Definition 7 (Track morphism). Given a direct derivation $G \Rightarrow H$ as in diagram (1), the *track morphism* $\text{tr}_{G \Rightarrow H}: G \rightarrow H$ is the partial hypergraph morphism defined by

$$\text{tr}_{G \Rightarrow H}(x) = \begin{cases} c'(c^{-1}(x)) & \text{if } x \in c(D), \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Here $c: D \hookrightarrow G$ and $c': D \rightarrow H$ are the morphisms in the lower row of (1) and $c^{-1}: c(D) \hookrightarrow D$ maps each item $c(x)$ to x .

The track morphism of a derivation $\Delta: G_0 \Rightarrow^* G_n$ is defined by $\text{tr}_\Delta = \text{id}_{G_0}$ if $n = 0$ and $\text{tr}_\Delta = \text{tr}_{G_1 \Rightarrow^* G_n} \circ \text{tr}_{G_0 \Rightarrow G_1}$ otherwise, where id_{G_0} is the identity morphism on G_0 .

Definition 8 (Hypergraph rewriting system). A *hypergraph rewriting system* $\langle \Sigma, \mathcal{R} \rangle$ consists of a signature Σ and a set \mathcal{R} of rules over Σ . Such a system is *finite* if Σ_V, Σ_E and \mathcal{R} are finite, and it is *injective* if all rules in \mathcal{R} are injective.

The system $\langle \Sigma, \mathcal{R} \rangle$ is denoted by \mathcal{R} if Σ is irrelevant or clear from the context. Given hypergraphs G and H over Σ such that there is a direct derivation $G \Rightarrow_r H$ with $r \in \mathcal{R}$, this is written $G \Rightarrow_{\mathcal{R}} H$. The system $\langle \Sigma, \mathcal{R} \rangle$ is a *graph rewriting system* if for each label l in Σ_E , $\text{Type}(l)$ contains only strings of length two.

Example 1. Figure 5 shows a hypergraph rewriting system defining a class of control-flow graphs. A hypergraph belongs to the class if and only if the rules can

³ “+” denotes the disjoint union of sets.

⁴ A *partial hypergraph morphism* $f: G \rightarrow H$ is a hypergraph morphism from a subhypergraph of G to H .

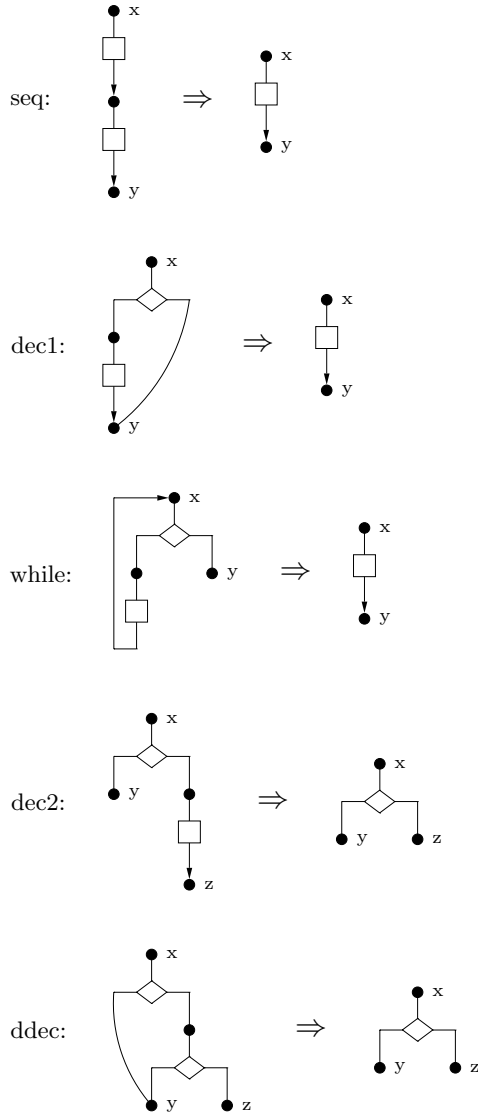


Fig. 5. Hypergraph rewriting system for flow-graph reduction

reduce it to the smallest flow graph, which is the hypergraph on the right-hand side of the rule seq. The underlying signature contains a single node label and two hyperedge labels which are graphically represented by hyperedges formed as squares and rhombs, where the order among the links of a rhomb is “top-left-right”. The flow graphs defined in this way correspond to a subset of the so-called semi-structured flow graphs of Farrow, Kennedy and Zucconi [16]. This example will be continued as Example 3 where it is shown that the rewriting system is confluent. □

Since rules are matched injectively, one sometimes wants to include in a hypergraph rewriting system some or all of the homomorphic images of a rule.

Definition 9 (Quotient rule). A rule $r': \langle L' \leftarrow K' \rightarrow R' \rangle$ is a *quotient* of a rule $r: \langle L \leftarrow K \rightarrow R \rangle$ if there are two pushouts of the form

$$\begin{array}{ccccc}
 L & \longleftarrow & K & \longrightarrow & R \\
 \downarrow & & \downarrow & & \downarrow \\
 L' & \longleftarrow & K' & \longrightarrow & R'
 \end{array} \tag{2}$$

where the vertical hypergraph morphisms are surjective.

For example, Figure 6 shows a rule and its only proper quotient. A rule has—up to isomorphism—only finitely many quotients and hence the double-pushout approach with unrestricted matching morphisms can be simulated in the present setting by replacing each rule with its quotients. However, injective matching allows to select a subset of the quotients and this feature provides additional expressiveness [18].



Fig. 6. A rule and its quotient

3.2 Clipping and Embedding

For proving confluence of graph transformation via critical pairs in Section 4, it will be necessary both to restrict derivations by clipping off context and to extend derivations with context. The technical machinery for these operations is presented next.

Definition 10 (Instance of a derivation). Let the derivation $\Delta: G_0 \Rightarrow^* G_n$ be given by the pushouts $(1), (1'), \dots, (n), (n')$ of Figure 7 and suppose there are pushouts $(\underline{1}), (\underline{1}'), \dots, (\underline{n}), (\underline{n}')$ whose vertical morphisms are injective. Then the derivation $\Delta': G'_0 \Rightarrow^* G'_n$ consisting of the composed pushouts $(1) + (\underline{1}), \dots, (n') + (\underline{n}')$ ⁵ is an *instance* of Δ based on the morphism $G_0 \hookrightarrow G'_0$. If moreover $G_0 \hookrightarrow G'_0$ is an isomorphism, then Δ and Δ' are *isomorphic* derivations.⁶

The Clipping Theorem below will show that given a derivation $\Delta': G'_0 \Rightarrow^* G'_n$ and an injective morphism $G_0 \hookrightarrow G'_0$, Δ' can be restricted to a derivation $\Delta: G_0 \Rightarrow^* G_n$ if all items in G'_0 that at some stage are used by a rule application in Δ' , belong to the image of G_0 in G'_0 .

In what follows, given a direct derivation $G \Rightarrow H$ as in diagram (1), the subhypergraph $f(L)$ of G is denoted by $\text{Match}(G \Rightarrow H)$.

⁵ See Lemma 14 in the Appendix for the composition of pushouts.

⁶ In this case all the morphisms $G_i \hookrightarrow G'_i$ and $D_i \hookrightarrow D'_i$ are isomorphisms.

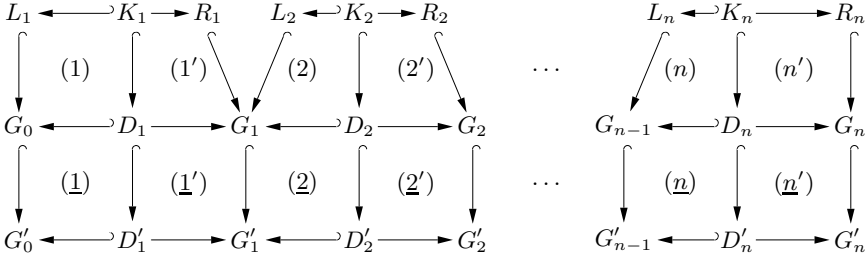


Fig. 7. Derivations $\Delta: G_0 \Rightarrow^* G_n$ and $\Delta': G'_0 \Rightarrow^* G'_n$

Definition 11 (Use_Δ). Given a derivation $\Delta: G_0 \Rightarrow^* G_n$, the subhypergraph Use_Δ of G_0 consists of all items x such that there is some $i \geq 0$ with $\text{tr}_{G_0 \Rightarrow^* G_i}(x) \in \text{Match}(G_i \Rightarrow G_{i+1})$.

Thus, Use_Δ contains those items of G_0 that at some point will occur in the image of the left-hand side of a rule. It is easily seen that these items constitute a subhypergraph of G_0 . The following theorem was first proved in [23], in the setting of injective graph rewriting systems with unrestricted matching.

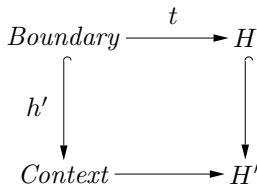
Theorem 2 (Clipping [28]). *Given a derivation $\Delta': G' \Rightarrow^* H'$ and an injective hypergraph morphism $h: G \hookrightarrow G'$ such that $\text{Use}_{\Delta'} \subseteq h(G)$, there exists a derivation $\Delta: G \Rightarrow^* H$ such that Δ' is an instance of Δ based on h .*

Next an embedding operation for derivations is considered which is inverse to the clipping operation. The associated theorem will refer to the “persistent” part of the start hypergraph of a derivation.

Definition 12 (Persist_Δ). Given a derivation $\Delta: G \Rightarrow^* H$, Persist_Δ is the subhypergraph of G consisting of all items x such that $\text{tr}_{G \Rightarrow^* H}(x)$ is defined.

In other words, Persist_Δ is the domain of definition of $\text{tr}_{G \Rightarrow^* H}$. The Embedding Theorem given next allows to extend a derivation with context provided that context edges are not attached to non-persistent nodes. The result was originally established in [8, 23], for injective graph rewriting systems with unrestricted matching.

Theorem 3 (Embedding [28]). *Let $\Delta: G \Rightarrow^* H$ be a derivation, $h: G \hookrightarrow G'$ an injective hypergraph morphism and Boundary be the discrete subhypergraph of G consisting of all nodes x such that $h(x)$ is incident to a hyperedge in $G' - h(G)$. If $\text{Boundary} \subseteq \text{Persist}_\Delta$, then there exists a derivation $\Delta': G' \Rightarrow^* H'$ such that Δ' is an instance of Δ based on h . Moreover, there exists a pushout*



where *Context* is the subhypergraph $(G' - h(G)) \cup h(\text{Boundary})$ of G' , h' is the restriction of h to *Boundary* and *Context*, and t is the restriction of $\text{tr}_{G \Rightarrow^* H}$ to *Boundary*.

3.3 Independence

Next a basic commutation result is presented showing that independent steps $H_1 \leftarrow_{r_1} G \Rightarrow_{r_2} H_2$ give rise to steps of the form $H_1 \Rightarrow_{r_2} H \leftarrow_{r_1} H_2$. Roughly speaking, the given steps are independent if the intersection of the left-hand sides of r_1 and r_2 in G consists of common interface items. If one of the rules is not injective, however, an additional injectivity condition is needed. In the following let r_i denote a rule $\langle L_i \leftarrow K_i \rightarrow R_i \rangle$, for $i = 1, 2$.

Definition 13 (Independence). Direct derivations $H_1 \leftarrow_{r_1} G \Rightarrow_{r_2} H_2$ as in Figure 8 are *independent* if there are hypergraph morphisms $L_1 \rightarrow D_2$ and $L_2 \rightarrow D_1$ such that the following holds:

- Commutativity: $L_1 \rightarrow D_2 \hookrightarrow G = L_1 \hookrightarrow G$ and $L_2 \rightarrow D_1 \hookrightarrow G = L_2 \hookrightarrow G$.
- Injectivity: $L_1 \rightarrow D_2 \rightarrow H_2$ and $L_2 \rightarrow D_1 \rightarrow H_1$ are injective.

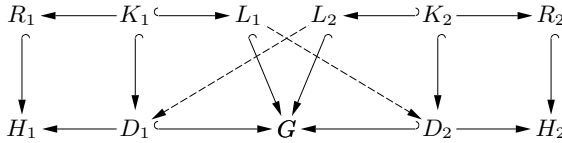


Fig. 8. Independence

If r_1 and r_2 are injective, the direct derivations of Figure 8 are independent if and only if the intersection of the two left-hand sides coincides with the intersection of the two interfaces.

Lemma 2 (Independence for injective rules). *Let r_1 and r_2 be injective rules. Then direct derivations $H_1 \leftarrow_{r_1, g_1} G \Rightarrow_{r_2, g_2} H_2$ are independent if and only if $g_1(L_1) \cap g_2(L_2) \subseteq g_1(K_1) \cap g_2(K_2)$.*

Lemma 2 does not hold for non-injective rules as the injectivity condition of Definition 13 may be violated [18]. The following result was first proved in [11], for graph rewriting systems with unrestricted matching.

Theorem 4 (Commutativity [28]). *If $H_1 \leftarrow_{r_1} G \Rightarrow_{r_2} H_2$ are independent direct derivations, then there exists an H such that $H_1 \Rightarrow_{r_2} H \leftarrow_{r_1} H_2$.*

4 Confluence

4.1 Rewriting Modulo Isomorphism

In graph transformation, the structure and labelling of (hyper-)graphs matters rather than the identities of nodes and edges. Hence isomorphic hypergraphs are

usually considered as equal. This, however, causes a subtle problem as confluence is defined via the reflexive-transitive closure \Rightarrow^* which need not contain isomorphism. For example, consider the rule

$$r: \quad \textcircled{a} \quad \Rightarrow \quad \textcircled{b}$$

and two rewrite steps $R \Leftarrow_r L \Rightarrow_r R'$, where L is the left-hand side of r . Then R and R' are isomorphic but not necessarily equal, so the relation \Rightarrow_r is not confluent in the sense of Definition 1(2). On the other hand, \Rightarrow_r becomes confluent—in fact subcommutative—if it is considered as a relation on isomorphism classes of hypergraphs. A rigorous treatment of confluence requires either to consider the transformation of isomorphism classes of hypergraphs or—equivalently—to replace confluence with confluence modulo isomorphism. In what follows, $[G]$ denotes the isomorphism class $\{G' \mid G' \cong G\}$ of a hypergraph G .

Definition 14 (Rewriting modulo isomorphism). Given a hypergraph rewriting system $\langle \Sigma, \mathcal{R} \rangle$, the relation $\Rightarrow_{\mathcal{R}, \cong}$ on isomorphism classes of hypergraphs over Σ is defined by: $[G] \Rightarrow_{\mathcal{R}, \cong} [H]$ if there are hypergraphs G' and H' such that $G \cong G' \Rightarrow_{\mathcal{R}} H' \cong H$. The relation $\Rightarrow_{\mathcal{R}, \cong}$ is referred to as *hypergraph rewriting modulo isomorphism*.

By the uniqueness of pushouts up to isomorphism (Lemma 13.3), it is clear that $[G] \Rightarrow_{\mathcal{R}, \cong} [H]$ if and only if $G \Rightarrow_{\mathcal{R}} H$. But $[G] \Rightarrow_{\mathcal{R}, \cong}^* [H]$ need not imply $G \Rightarrow_{\mathcal{R}}^* H$ since G and H may be distinct in the case $[G] = [H]$.

Definition 15 (Confluence of hypergraph rewriting systems). A hypergraph rewriting system $\langle \Sigma, \mathcal{R} \rangle$ is *confluent* (*locally confluent*, *subcommutative*) if the relation $\Rightarrow_{\mathcal{R}, \cong}$ is confluent (locally confluent, subcommutative).

A drawback of hypergraph rewriting modulo isomorphism is that one loses access to nodes and hyperedges. Fortunately, confluence of $\Rightarrow_{\mathcal{R}, \cong}$ can be characterized as confluence of $\Rightarrow_{\mathcal{R}}$ modulo isomorphism. (See Definition 1 for (local) confluence modulo isomorphism.)

Lemma 3. *Let $\langle \Sigma, \mathcal{R} \rangle$ be a hypergraph rewriting system.*

- (1) *The following are equivalent:*
 - $\langle \Sigma, \mathcal{R} \rangle$ is confluent.
 - The relation $\Rightarrow_{\mathcal{R}}$ is confluent modulo isomorphism.
 - For all hypergraphs G, G_1 and G_2 over Σ , $G_1 \Leftarrow_{\mathcal{R}}^* G \Rightarrow_{\mathcal{R}}^* G_2$ implies that there are hypergraphs H_1 and H_2 such that $G_1 \Rightarrow_{\mathcal{R}}^* H_1 \cong H_2 \Leftarrow_{\mathcal{R}}^* G_2$.
- (2) $\langle \Sigma, \mathcal{R} \rangle$ is locally confluent if and only if $\Rightarrow_{\mathcal{R}}$ is locally confluent modulo isomorphism.

Proof. By Lemma 13.3, $G \cong G' \Rightarrow_{\mathcal{R}} H' \cong H$ implies $G \Rightarrow_{\mathcal{R}} H$. The above characterizations are easy consequences of this fact. □

Note that—different from confluence—the relation $\Rightarrow_{\mathcal{R}, \cong}$ is terminating if and only if $\Rightarrow_{\mathcal{R}}$ is terminating, as $[G] \Rightarrow_{\mathcal{R}, \cong} [H]$ if and only if $G \Rightarrow_{\mathcal{R}} H$. A hypergraph rewriting system $\langle \Sigma, \mathcal{R} \rangle$ is said to be *terminating* if the relation $\Rightarrow_{\mathcal{R}, \cong}$ (equivalently, $\Rightarrow_{\mathcal{R}}$) is terminating. The following lemma follows directly from Newman’s Lemma (Lemma 1).

Lemma 4. *A terminating hypergraph rewriting system is confluent if and only if it is locally confluent.*

4.2 The Decision Problem

This section presents a reduction of the Post Correspondence Problem showing that confluence is undecidable for terminating graph rewriting systems. The precise result is as follows.

Theorem 5. *The following problem is undecidable in general:*

Instance: *A finite, injective and terminating graph rewriting system $\langle \Sigma, \mathcal{R} \rangle$ where Σ_V is a singleton.*

Question: *Is $\langle \Sigma, \mathcal{R} \rangle$ confluent?*

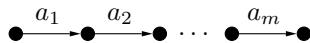
The rest of this section is used to prove Theorem 5, with a summary of the proof given at the end of the section. The plan is to encode every instance \mathcal{I} of the Post Correspondence Problem (PCP) as a (finite, injective and) terminating graph rewriting system $\langle \Sigma(\mathcal{I}), \mathcal{R}(\mathcal{I}) \rangle$ that is confluent if and only if \mathcal{I} does not have a solution.

Recall that the PCP is the problem to decide, given a nonempty list

$$\mathcal{I} = \langle (\alpha_1, \beta_1), \dots, (\alpha_n, \beta_n) \rangle$$

of pairs of words over some finite alphabet Γ , whether there exists a nonempty sequence i_1, \dots, i_k of indices such that $\alpha_{i_1} \dots \alpha_{i_k} = \beta_{i_1} \dots \beta_{i_k}$. The list \mathcal{I} is an *instance* of the PCP, and a sequence i_1, \dots, i_k with the above property is a *solution* of this instance. It is well-known that it is undecidable in general whether an instance of the PCP has a solution, see for example [33].

The following encoding of the PCP represents strings $a_1 \dots a_m$ as graphs consisting of m consecutive edges labelled with a_1, \dots, a_m , depicted as



which includes the case $m = 0$ where the graph consists of a single node.

Consider now an arbitrary instance $\mathcal{I} = \langle (\alpha_1, \beta_1), \dots, (\alpha_n, \beta_n) \rangle$. Let $\Sigma(\mathcal{I})_V = \{\bullet\}$ and $\Sigma(\mathcal{I})_E = \Gamma \cup \{1, \dots, n\} \cup \{\bowtie, A, B\}$ where it is assumed, without loss of generality, that the three latter sets are pairwise disjoint. The rule set $\mathcal{R}(\mathcal{I})$ is partitioned into subsets $\mathcal{R}_1(\mathcal{I})$ to $\mathcal{R}_4(\mathcal{I})$ which are presented by rule schemata in Figure 9 to Figure 12.

The rule schemata s_1 and s_2 of \mathcal{R}_1 enable divergent steps $T \leftarrow_{s_1} S \Rightarrow_{s_2} U$ which represent the choice to create a loop labelled with \bowtie or to check a possible

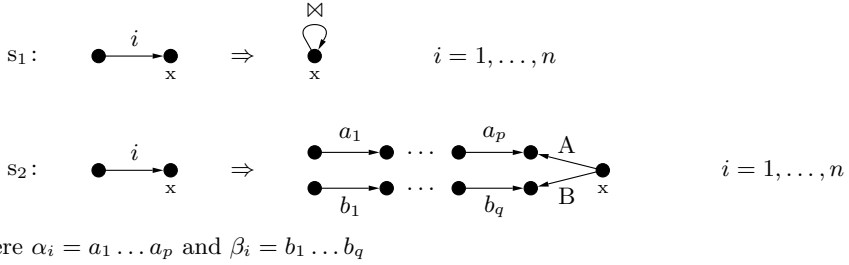


Fig. 9. $\mathcal{R}_1(\mathcal{I})$

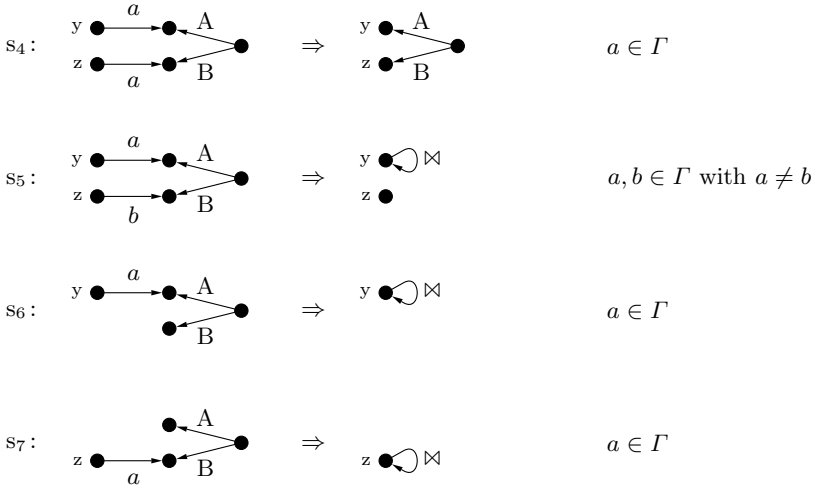
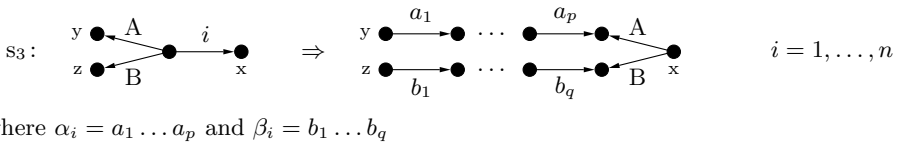


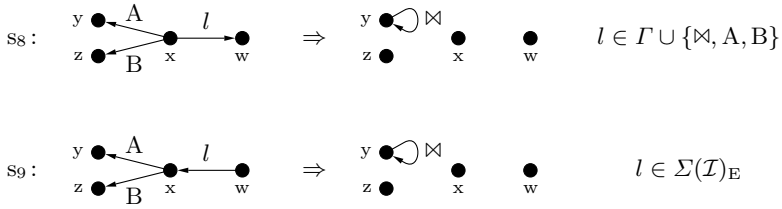
Fig. 10. $\mathcal{R}_2(\mathcal{I})$

solution of \mathcal{I} . The rules of \mathcal{R}_2 check whether a sequence of indices is a solution of \mathcal{I} , \mathcal{R}_3 detects ill-shaped graphs, and \mathcal{R}_4 performs “garbage collection” in the presence of a \curvearrowright -labelled loop.

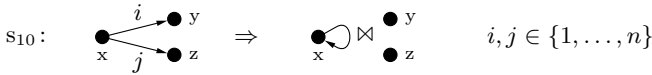
We proceed by showing that $\langle \Sigma(\mathcal{I}), \mathcal{R}(\mathcal{I}) \rangle$ is terminating, and that it is confluent if and only if \mathcal{I} does not have a solution.

Lemma 5. *The system $\langle \Sigma(\mathcal{I}), \mathcal{R}(\mathcal{I}) \rangle$ is terminating.*

Proof. Suppose that $\mathcal{R}(\mathcal{I})$ admits an infinite derivation $G_1 \Rightarrow G_2 \Rightarrow \dots$. Since no application of a rule in $\mathcal{R}(\mathcal{I})$ increases the number of edges with label in $\{1, \dots, n\}$, there is some $t \geq 1$ such that the number of these edges is the same

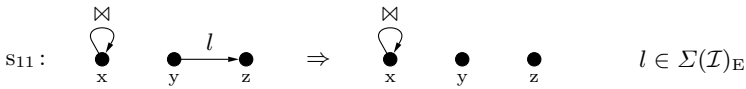


including the quotient obtained by merging x and w



including the quotient obtained by merging y and z

Fig. 11. $\mathcal{R}_3(\mathcal{I})$



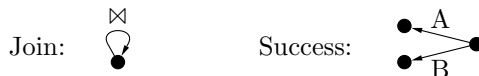
including all quotients



Fig. 12. $\mathcal{R}_4(\mathcal{I})$

in all G_i with $i \geq t$. It follows that the rule schemata s_1 to s_3 are not applied in $G_t \Rightarrow G_{t+1} \Rightarrow \dots$. But all other rule schemata in $\mathcal{R}(\mathcal{I})$ decrease the sum of the numbers of nodes and edges, hence $G_t \Rightarrow G_{t+1} \Rightarrow \dots$ cannot be infinite. \square

The next four lemmata will show that the instance \mathcal{I} has a solution if and only if $\langle \Sigma(\mathcal{I}), \mathcal{R}(\mathcal{I}) \rangle$ is not confluent. The “only if”-direction follows from the observation that every solution of \mathcal{I} —represented as a chain of edges—can be reduced to two non-isomorphic normal forms. Define graphs Join and Success as follows:

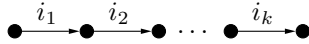


Lemma 6. Every graph over $\Sigma(\mathcal{I})$ containing a \bowtie -labelled loop reduces to Join.

Proof. Apply the rules in $\mathcal{R}_4(\mathcal{I})$ as long as possible. \square

Lemma 7. If \mathcal{I} has a solution, then $\langle \Sigma(\mathcal{I}), \mathcal{R}(\mathcal{I}) \rangle$ is not confluent.

Proof. Let i_1, \dots, i_k be a solution of \mathcal{I} . Then the graph

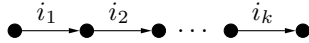


can be reduced to Join by first applying rule schema s_1 and then as long as possible the rules in $\mathcal{R}_4(\mathcal{I})$. On the other hand, by first applying rule schema s_2 and then as long as possible the rule schemata s_3 and s_4 , the graph will reduce to Success. Since Join and Success are normal forms, $\langle \Sigma(\mathcal{I}), \mathcal{R}(\mathcal{I}) \rangle$ is not confluent. □

To complete the proof of Theorem 5, it remains to be shown that $\langle \Sigma(\mathcal{I}), \mathcal{R}(\mathcal{I}) \rangle$ is confluent if \mathcal{I} does not have a solution. The next lemma provides a crucial argument for this proof.

Lemma 8. *If \mathcal{I} does not have a solution, then for every graph G over $\Sigma(\mathcal{I})$, $G \Rightarrow_{s_2} H$ implies $H \Rightarrow_{\mathcal{R}(\mathcal{I})}^* \text{Join}$.*

Proof. Let $G \Rightarrow_{s_2} H$. Call a subgraph C of H an *index chain* of length k , $k \geq 1$, if C has the form



with $i_1, \dots, i_k \in \{1, \dots, n\}$, where only the rightmost node may be incident to edges not belonging to C .

Let now C be the longest index chain in G such that the leftmost edge of C is replaced by the step $G \Rightarrow_{s_2} H$. Let e_1, \dots, e_k be the edges of C in left-to-right order and $\text{lab}_C(e_j) = i_j$ for $j = 1, \dots, k$. Then $H \Rightarrow_{s_3}^{k-1} H'$ for some graph H' such that the j^{th} step in $G \Rightarrow_{s_2} H \Rightarrow_{s_3}^{k-1} H'$ replaces e_j by two sequences of edges representing the strings α_{i_j} and β_{i_j} . Let v be the destination of edge e_k in C . In H' , v is the source of two edges labelled with A and B which were created in the k^{th} step. If v is incident to any other edges, then $H' \Rightarrow_{\mathcal{R}_3(\mathcal{I})} H''$ for some H'' and hence $H'' \Rightarrow^* \text{Join}$ by Lemma 6. Assume that v is incident to no other edges than the two edges labelled with A and B. The generated strings $\alpha_{i_1} \dots \alpha_{i_k}$ and $\beta_{i_1} \dots \beta_{i_k}$ are distinct as otherwise i_1, \dots, i_k would be a solution of \mathcal{I} . Let $H' \Rightarrow^* H''$ be a derivation in which the rule schemata s_4, s_5, s_6 and s_7 are applied as long as possible. This derivation must include an application of s_5, s_6 or s_7 because otherwise $\alpha_{i_1} \dots \alpha_{i_k} = \beta_{i_1} \dots \beta_{i_k}$. Thus H'' contains a \bowtie -edge and, by Lemma 6, reduces to Join. □

By Lemma 3 and Lemma 4, showing that $\langle \Sigma(\mathcal{I}), \mathcal{R}(\mathcal{I}) \rangle$ is confluent if \mathcal{I} has no solution amounts to proving that $\Rightarrow_{\mathcal{R}(\mathcal{I})}$ is locally confluent modulo isomorphism. It will turn out that in every situation $H_1 \leftarrow_{\mathcal{R}(\mathcal{I})} G \Rightarrow_{\mathcal{R}(\mathcal{I})} H_2$ where H_1 and H_2 are not isomorphic, either the steps are independent and hence can be commuted or H_1 and H_2 reduce to the graph Join.

Lemma 9. *If \mathcal{I} does not have a solution, then $\mathcal{R}(\mathcal{I})$ is confluent.*

Proof. By Lemma 3 and Lemma 4, it suffices to show that $\Rightarrow_{\mathcal{R}(\mathcal{I})}$ is locally confluent modulo isomorphism since $\mathcal{R}(\mathcal{I})$ is terminating. Consider two direct derivations $H_1 \leftarrow_{r_1, g_1} G \Rightarrow_{r_2, g_2} H_2$ by rules $r_i: \langle L_i \leftrightarrow K_i \rightarrow R_i \rangle \in \mathcal{R}(\mathcal{I})$, for $i = 1, 2$, such that $H_1 \not\cong H_2$. Moreover, assume that the two steps are not independent as otherwise they can be commuted by Theorem 4. By the injectivity of r_1 and r_2 and Lemma 2,

$$g_1(L_1) \cap g_2(L_2) \not\subseteq g_1(K_1) \cap g_2(K_2). \tag{3}$$

There are four cases.

Case 1: $r_1, r_2 \in \mathcal{R}_3(\mathcal{I}) \cup \mathcal{R}_4(\mathcal{I})$. Then both H_1 and H_2 contain a \bowtie -loop and hence, by Lemma 6, reduce to the graph Join.

Case 2: $r_1 \in \mathcal{R}_1(\mathcal{I}) \cup \mathcal{R}_2(\mathcal{I})$, $r_2 \in \mathcal{R}_3(\mathcal{I}) \cup \mathcal{R}_4(\mathcal{I})$. Then H_2 reduces to Join. If r_1 is an instance of rule schema s_1, s_5, s_6 or s_7 , then H_1 contains a \bowtie -loop and reduces to Join, too. Let therefore r_1 be an instance of rule schema s_2, s_3 or s_4 . By (3) and the dangling condition (Definition 6), four subcases remain. In each of these cases there will be a step $H_1 \Rightarrow_{\mathcal{R}_3(\mathcal{I})} H'_1$ and hence a derivation $H_1 \Rightarrow_{\mathcal{R}_3(\mathcal{I})} H'_1 \Rightarrow_{\mathcal{R}_4(\mathcal{I})}^* \text{Join}$.

Case 2.1: r_1 is an instance of s_2 and r_2 is an instance of s_9 . Then H_1 contains a node with two outgoing A-edges and two outgoing B-edges, so s_8 is applicable to H_1 .

Case 2.2: r_1 is an instance of s_2 and r_2 is an instance of s_{11} . Then H_1 contains a \bowtie -loop and hence s_{11} is applicable to H_1 .

Case 2.3: r_1 is an instance of s_3 and r_2 is an instance of s_9 . Again H_1 contains a node with two outgoing A-edges and two outgoing B-edges, so s_8 is applicable to H_1 .

Case 2.4: r_1 is an instance of s_3 and r_2 is an instance of s_{11} . Then H_1 contains a \bowtie -loop so that s_{11} is applicable to H_1 .

Case 3: $r_1 \in \mathcal{R}_3(\mathcal{I}) \cup \mathcal{R}_4(\mathcal{I})$, $r_2 \in \mathcal{R}_1(\mathcal{I}) \cup \mathcal{R}_2(\mathcal{I})$. This case is symmetric to Case 2.

Case 4: $r_1, r_2 \in \mathcal{R}_1(\mathcal{I}) \cup \mathcal{R}_2(\mathcal{I})$. Then one of the rules, say r_1 , is an instance of s_1 and r_2 is an instance of s_2 . By Lemma 6 and Lemma 8, both H_1 and H_2 reduce to Join. □

Proof of Theorem 5. For every instance \mathcal{I} of the PCP, the system $\langle \Sigma(\mathcal{I}), \mathcal{R}(\mathcal{I}) \rangle$ is finite and injective and contains only one node label. Moreover, $\mathcal{R}(\mathcal{I})$ is terminating by Lemma 5. By Lemma 7 and Lemma 9, $\langle \Sigma(\mathcal{I}), \mathcal{R}(\mathcal{I}) \rangle$ is confluent if and only if \mathcal{I} does not have a solution. This concludes the proof of Theorem 5 as the PCP is known to be undecidable [33]. □

4.3 Critical Pairs

The concept of a critical pair was introduced by Knuth and Bendix [22] who showed that confluence of a terminating term rewriting system can be tested by checking for each critical pair whether both terms have a common reduct. This subsection explores to what extent this idea can be adopted for the setting

of hypergraph rewriting. The motivation is to ensure that arbitrary divergent steps $H_1 \leftarrow_{r_1} G \Rightarrow_{r_2} H_2$ have a common reduct if this is the case for those steps where G represents a “critical overlap” of the left-hand sides of r_1 and r_2 . By Theorem 4 (commutativity), such an overlap is critical only if the two steps are not independent. This suggests the following definition of a critical pair.

Definition 16 (Critical pair). Let $r_i: \langle L_i \leftarrow K_i \rightarrow R_i \rangle$ be rules, for $i = 1, 2$. A pair of direct derivations $U_1 \leftarrow_{r_1, g_1} S \Rightarrow_{r_2, g_2} U_2$ is a *critical pair* if

- (1) $S = g_1(L_1) \cup g_2(L_2)$ and
- (2) the steps are not independent.

Moreover, $g_1 \neq g_2$ has to hold if $r_1 = r_2$.

Two critical pairs $U_1 \leftarrow S \Rightarrow U_2$ and $U'_1 \leftarrow S' \Rightarrow U'_2$ are *isomorphic* if there is an isomorphism $f: S \rightarrow S'$ such that for $i = 1, 2$, $S' \Rightarrow U'_i$ is an instance of $S \Rightarrow U_i$ based on f . In the sequel, isomorphic critical pairs will be equated so that condition (1) guarantees that a finite set of rules has only a finite number of critical pairs.

By Theorem 4, hypergraph rewriting systems without critical pairs enjoy a strong commutation property which implies subcommutativity and hence confluence. This is fundamentally different from the situation for term rewriting systems where the absence of critical pairs guarantees only local confluence [20].

Theorem 6. *If $\langle \Sigma, \mathcal{R} \rangle$ is a hypergraph rewriting system without critical pairs, then $H_1 \leftarrow_{\mathcal{R}} G \Rightarrow_{\mathcal{R}} H_2$ implies $H_1 \cong H_2$ or that there is a hypergraph H such that $H_1 \Rightarrow_{\mathcal{R}} H \leftarrow_{\mathcal{R}} H_2$.*

Proof. Let $H_1 \leftarrow_{r_1, g_1} G \Rightarrow_{r_2, g_2} H_2$. If the two steps are independent, then by Theorem 4 there are two steps of the form $H_1 \Rightarrow_{r_2} H \leftarrow_{r_1} H_2$. Assume therefore that the given steps are not independent. By Theorem 2 (clipping), there are two restricted steps of the form

$$U_1 \leftarrow_{r_1} g_1(L_1) \cup g_2(L_2) \Rightarrow_{r_2} U_2$$

such that the given steps are instances of the restricted steps. It is not difficult to show that the latter steps are not independent either. Hence, as there are no critical pairs, $r_1 = r_2$ and $g_1 = g_2$ must hold. It follows $H_1 \cong H_2$ since the result of a rewrite step is determined uniquely up to isomorphism. \square

By the proof of the corollary below it follows that in the absence of critical pairs, $[H_1] \leftarrow_{\mathcal{R}, \cong} [G] \Rightarrow_{\mathcal{R}, \cong} [H_2]$ with $[H_1] \neq [H_2]$ implies that there is a hypergraph H with $[H_1] \Rightarrow_{\mathcal{R}, \cong} [H] \leftarrow_{\mathcal{R}, \cong} [H_2]$. In [4] this property is denoted by CR^1 (for arbitrary binary relations).

Corollary 1. *Hypergraph rewriting systems without critical pairs are subcommutative.*

One could try to overcome this problem by adding the rule

$$r_3: \quad \text{b} \begin{array}{c} \curvearrowright \\ \bullet \end{array} \bullet \Rightarrow \emptyset$$

which reduces the outer hypergraphs of the critical pair to the empty hypergraph. Adding r_3 does not create new critical pairs, so the only critical pair is “joinable by derivations with identical track morphisms”. Still, this is not sufficient for local confluence modulo isomorphism: r_3 cannot be applied to the outer hypergraphs of the latter derivation pair because of the dangling condition for direct derivations. In other words, the joining derivations cannot be embedded into context as r_3 removes the nodes to which the context edge is attached. \square

This example suggests that the joining derivations of a critical pair need to preserve certain nodes and possess track morphisms that are compatible with the isomorphism between the resulting hypergraphs.

Definition 18 (Strong joinability). Given a critical pair $\Gamma: U_1 \leftarrow S \Rightarrow U_2$, let $\text{Persist}_\Gamma = \text{Persist}_{S \Rightarrow U_1} \cap \text{Persist}_{S \Rightarrow U_2}$. Then Γ is *strongly joinable* if there are derivations $U_1 \Rightarrow^* W_1$, $U_2 \Rightarrow^* W_2$ and an isomorphism $f: W_1 \rightarrow W_2$ such that for each node v in Persist_Γ ,

- (1) $\text{tr}_{S \Rightarrow U_1 \Rightarrow^* W_1}(v)$ and $\text{tr}_{S \Rightarrow U_2 \Rightarrow^* W_2}(v)$ are defined and
- (2) $f_V(\text{tr}_{S \Rightarrow U_1 \Rightarrow^* W_1}(v)) = \text{tr}_{S \Rightarrow U_2 \Rightarrow^* W_2}(v)$.

So each node that is preserved by both $S \Rightarrow U_1$ and $S \Rightarrow U_2$ has to be preserved by $U_1 \Rightarrow^* W_1$ and $U_2 \Rightarrow^* W_2$ as well, and its descendants in W_1 and W_2 have to be related by the isomorphism f .

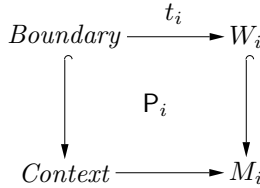
Lemma 10 (Critical Pair Lemma). *A hypergraph rewriting system is locally confluent if all its critical pairs are strongly joinable.*

Proof. Let $\langle \Sigma, \mathcal{R} \rangle$ be a hypergraph rewriting system such that all its critical pairs are strongly joinable. By Lemma 3 it suffices to show that $\Rightarrow_{\mathcal{R}}$ is locally confluent modulo isomorphism. Consider two steps $H_1 \leftarrow_{r_1, g_1} G \Rightarrow_{r_2, g_2} H_2$ by rules $r_i: \langle L_i \leftarrow K_i \rightarrow R_i \rangle$, for $i = 1, 2$. If the two steps are independent, then Theorem 4 guarantees that there are two steps of the form $H_1 \Rightarrow_{r_2} H \leftarrow_{r_1} H_2$. Assume therefore that the given steps are not independent. Assume further that $r_1 \neq r_2$ or $g_1 \neq g_2$ as otherwise $H_1 \cong H_2$. Let now $S = g_1(L_1) \cup g_2(L_2)$. Then, for $i = 1, 2$, $\text{Use}_{G \Rightarrow H_i} \subseteq S$ and hence, by Theorem 2 (clipping), there are direct derivations $U_i \leftarrow_{r_i, g'_i} S \Rightarrow_{r_2, g'_2} U_2$ such that for $i = 1, 2$, $G \Rightarrow H_i$ is an instance of $S \Rightarrow U_i$ based on the inclusion $S \hookrightarrow G$. It is not difficult to check that since the steps $H_1 \leftarrow G \Rightarrow H_2$ are not independent, $U_1 \leftarrow S \Rightarrow U_2$ are not independent either and hence constitute a critical pair Γ . Thus, by assumption, there are derivations $U_1 \Rightarrow^* W_1$, $U_2 \Rightarrow^* W_2$ and an isomorphism $f: W_1 \rightarrow W_2$ such that for each node v in Persist_Γ , $f_V(\text{tr}_{S \Rightarrow U_1 \Rightarrow^* W_1}(v))$ and $\text{tr}_{S \Rightarrow U_2 \Rightarrow^* W_2}(v)$ are defined and equal.

Let *Boundary* be the subhypergraph of *S* consisting of all nodes that are incident to a hyperedge in $G - S$. Then $Boundary \subseteq \text{Persist}_{G \Rightarrow H_1} \cap \text{Persist}_{G \Rightarrow H_2}$ since $G \Rightarrow H_1$ and $G \Rightarrow H_2$ satisfy the dangling condition. Hence

$$Boundary \subseteq (\text{Persist}_{G \Rightarrow H_1} \cap \text{Persist}_{G \Rightarrow H_2}) \cap S = \text{Persist}_\Gamma.$$

Thus $\text{tr}_{S \Rightarrow U_1 \Rightarrow^* W_1}$ and $\text{tr}_{S \Rightarrow U_2 \Rightarrow^* W_2}$ are defined for all nodes in *Boundary*. So $Boundary \subseteq \text{Persist}_{S \Rightarrow U_i \Rightarrow^* W_i}$ for $i = 1, 2$ and hence, by Theorem 3 (embedding), there are derivations $G \Rightarrow_{r_1, g_1''} H'_1 \Rightarrow^* M_1$ and $G \Rightarrow_{r_2, g_2''} H'_2 \Rightarrow^* M_2$ that are instances of $S \Rightarrow_{r_1, g_1'} U_1 \Rightarrow^* W_1$ and $S \Rightarrow_{r_2, g_2'} U_2 \Rightarrow^* W_2$, respectively. Since both instances are based on the inclusion $S \hookrightarrow G$, g_i'' is the extension of g_i' to G and hence $g_i'' = g_i$, for $i = 1, 2$. It follows $H'_1 \cong H_1$ and $H'_2 \cong H_2$; thus for $i = 1, 2$, $H_i \Rightarrow^* M_i$ or $H_i \cong M_i$. So it remains to show $M_1 \cong M_2$ for $\Rightarrow_{\mathcal{R}}$ to be locally confluent modulo isomorphism. By Theorem 3, for $i = 1, 2$ there is a pushout



where $Context = (G - S) \cup Boundary$, $Boundary \hookrightarrow Context$ is the inclusion of *Boundary* in *Context*, and t_i is the restriction of $\text{tr}_{S \Rightarrow U_i \Rightarrow^* W_i}$ to *Boundary*. By assumption, $f \circ t_1 = t_2$. So $Boundary \hookrightarrow Context \rightarrow M_2 = Boundary \xrightarrow{t_2} W_2 \hookrightarrow M_2 = Boundary \xrightarrow{t_1} W_1 \xrightarrow{f} W_2 \hookrightarrow M_2$. Hence there is a unique morphism $M_1 \rightarrow M_2$ such that the diagram in Figure 13 commutes, where the

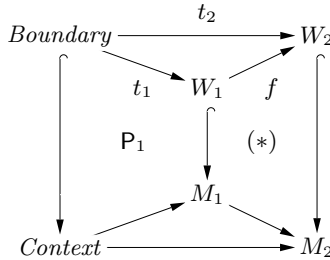


Fig. 13. Decomposing pushout P_2

outer diagram is the pushout P_2 . So $(*)$ is a pushout by Lemma 14.2. Since f is an isomorphism, Lemma 13.2 guarantees that $M_1 \rightarrow M_2$ is an isomorphism as well. This concludes the proof of Lemma 10. \square

Combining the Critical Pair Lemma with Newman’s Lemma (Lemma 1) yields a sufficient condition for the confluence of terminating hypergraph rewriting systems.

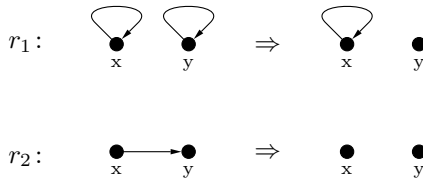
Theorem 7. *A terminating hypergraph rewriting system is confluent if all its critical pairs are strongly joinable.*

Example 3. This example continues Example 1 by applying Theorem 7 to the hypergraph rewriting system of Figure 5. That system is terminating since each of the rules reduces the size of a hypergraph it is applied to. Figure 14 shows that all critical pairs of the system are strongly joinable (where track morphisms are indicated by node names). Note that for each critical pair $\Gamma: U_1 \leftarrow S \Rightarrow U_2$, Persist_Γ is a proper subset of V_S . For instance, the persistent nodes of the topmost pair are w and z ; hence the isomorphism between the outer hypergraphs of this pair is compatible with the track morphisms in the way required by Definition 18.

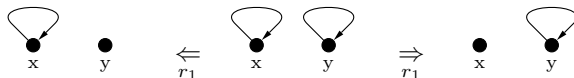
Thus, by Theorem 7, the hypergraph rewriting system of Example 1 is confluent. As a consequence, membership in the class of flow graphs defined by the system can be checked by a backtracking-free reduction algorithm: an input hypergraph G is reduced to its unique normal form $N(G)$ by applying the rules in any order; G is a flow graph if and only if $N(G)$ is the flow graph on the right-hand side of rule seq. \square

The converse of Theorem 7 does not hold because if all critical pairs of terminating and confluent systems were strongly joinable, confluence of terminating systems could be checked by testing critical pairs for strong joinability—contradicting Theorem 5. The next example gives two terminating and confluent systems having critical pairs that are not strongly joinable.

Example 4. Let the graph rewriting system $\langle \Sigma, \mathcal{R} \rangle$ consist of singletons Σ_V and Σ_E , and the following rules:



This system is terminating because every rule application decreases the number of edges by one. To see that it is confluent, consider two derivations $H_1 \xleftarrow{*}_{\mathcal{R}} G \xrightarrow{*}_{\mathcal{R}} H_2$. Then G , H_1 and H_2 have the same number of nodes. Hence $H_1 \xrightarrow{*}_{\mathcal{R}} H'_1 \cong H'_2 \xleftarrow{*}_{\mathcal{R}} H_2$, where H'_1 and H'_2 consist of $|V_G|$ nodes and either no edges (if G is loop-free) or one loop and no other edges. So the system is confluent. But the following critical pair⁷ is not strongly joinable:



⁷ The track morphisms are indicated by node names.

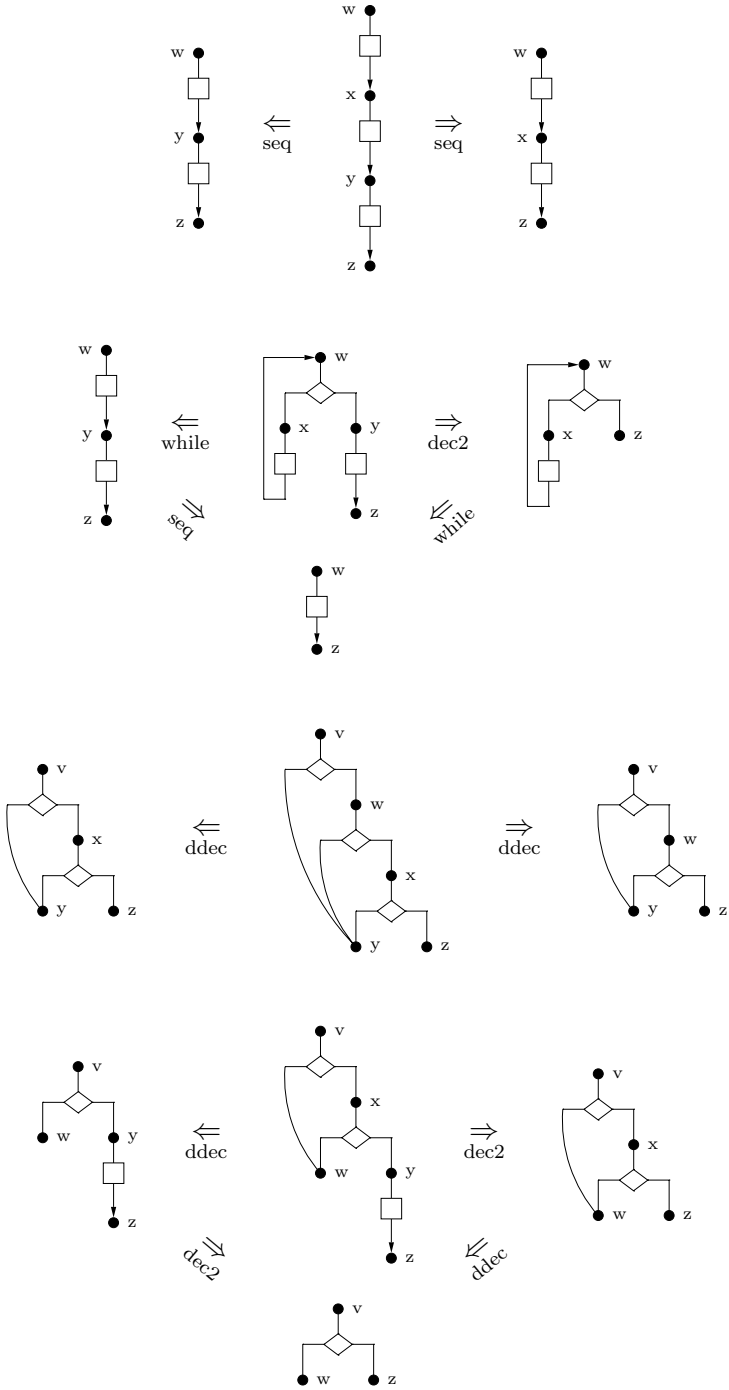
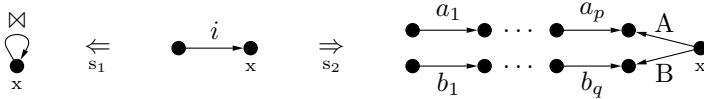


Fig. 14. The critical pairs of the rules of Figure 5

This is because the outer graphs are normal forms and the isomorphism between them is not compatible with the track morphisms of the rewrite steps.

A similar situation can arise in the system $\langle \Sigma(\mathcal{I}), \mathcal{R}(\mathcal{I}) \rangle$ of the previous subsection which encodes the Post Correspondence Problem. That system is confluent if the instance \mathcal{I} of the PCP has no solution. The rule schemata s_1 and s_2 give rise to the critical pair



which is joinable if \mathcal{I} has no solution, as then the graph on the right reduces to the graph on the left (see Lemma 9). However, this critical pair is not strongly joinable because the derivation from the right graph to the left graph deletes the node x which is preserved by both steps of the critical pair. \square

5 Related Work and Conclusion

This section mentions some related work and a couple of topics for future work.

The proof idea for showing that confluence of terminating systems is undecidable (Theorem 5) was inspired by Kapur’s, Narendran’s and Otto’s proof that ground-confluence is undecidable for terminating term rewriting systems [21].

The phenomenon that the joinability of all critical pairs need not imply local confluence of the rewrite relation refutes the critical pair lemma in [30]. In [26] the problem is avoided by imposing the strong restriction that distinct nodes in a graph must not have the same label.

The critical pair lemma of [27] was adopted to the so-called single-pushout approach to graph transformation in [24]. In [19] a critical pair lemma for a certain kind of attributed graph transformation (in the double-pushout approach) was presented. An abstract critical pair lemma in the setting of so-called adhesive high-level replacement systems was given in [10] and specialised in [14] to a form of attributed graph transformation (different from the aforementioned).

Future research should establish sufficient conditions under which all critical pairs of a confluent (hyper-)graph rewriting system are strongly joinable. For a finite and terminating system satisfying such a condition, confluence can be decided by checking all critical pairs for strong joinability.

Another application of the Critical Pair Lemma could be the completion of non-confluent systems, in analogy to the well-known completion procedure for term rewriting systems invented by Knuth and Bendix [22]. Such a procedure would add rules to a (hyper-)graph rewriting system until all critical pairs are strongly joinable. The procedure should preserve both the equivalence generated by the rewrite relation and termination.

Dedication. This paper is dedicated to Jan Willem Klop on the occasion of his 60th birthday. Since 1997, I have been a regular visitor of Jan Willem at CWI, the University of Nijmegen, and the Free University of Amsterdam. I am

grateful for Jan Willem’s hospitality and collaboration during all this time. The present paper is not directly concerned with our common topic of interest—term graph rewriting—but confluence of various forms of rewriting plays a prominent role in Jan Willem’s scientific work.

Appendix: Pushouts

This appendix presents the definition and construction of hypergraph pushouts and summarises some facts that are used in the proofs of Section 4.

Definition 19 (Pushout). Given two hypergraph morphisms $A \rightarrow B$ and $A \rightarrow C$, a hypergraph D together with two hypergraph morphisms $B \rightarrow D$ and $C \rightarrow D$ is a *pushout* of $A \rightarrow B$ and $A \rightarrow C$ if the following conditions are satisfied:

Commutativity: $A \rightarrow B \rightarrow D = A \rightarrow C \rightarrow D$.

Universal property: For all hypergraphs D' and hypergraph morphisms $B \rightarrow D'$ and $C \rightarrow D'$ such that $A \rightarrow B \rightarrow D' = A \rightarrow C \rightarrow D'$, there is a unique morphism $D \rightarrow D'$ such that $B \rightarrow D \rightarrow D' = B \rightarrow D'$ and $C \rightarrow D \rightarrow D' = C \rightarrow D'$. (See the right part of Figure 15.)

In this case the diagram on the left of Figure 15 is also called a pushout.

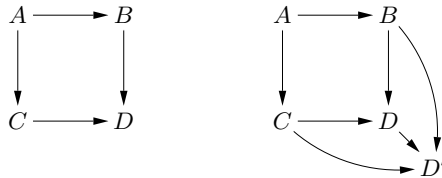
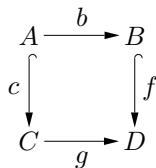


Fig. 15. A pushout diagram (on the left)

Intuitively, hypergraph D is obtained by gluing together B and C in a common part A . In particular, if $A \rightarrow C$ is injective, then D can be constructed from C by replacing the image of A with B . The following pushout construction assumes that one of the given morphisms is injective, which is the case for the two pushouts of a direct derivation as defined in Definition 5.

Lemma 11 (Pushout construction). Let $b: A \rightarrow B$ and $c: A \hookrightarrow C$ be hypergraph morphisms such that c is injective. Then a pushout



can be constructed as follows:

- $V_D = (V_C - c_V(V_A)) + V_B$ and $E_D = (E_C - c_E(E_A)) + E_B$;
 $\text{mark}_D(v) = \text{if } v \in V_B \text{ then } \text{mark}_B(v) \text{ else } \text{mark}_C(v)$;
 $\text{lab}_D(e) = \text{if } e \in E_B \text{ then } \text{lab}_B(e) \text{ else } \text{lab}_C(e)$;
 $\text{att}_D(e) = \text{if } e \in E_B \text{ then } \text{att}_B(e) \text{ else } g_V^*(\text{att}_C(e))$, where $g_V: V_C \rightarrow V_D$ is defined below.
- $f(x) = x$, separately for nodes and hyperedges.
- $g(x) = \text{if } x = c(x') \text{ for some } x' \text{ in } A \text{ then } b(x') \text{ else } x$, separately for nodes and hyperedges.

Proof. Analogous to the corresponding proof for graphs, see [9]. □

Definition 20 (Pushout complement). Given two hypergraph morphisms $A \rightarrow B$ and $B \rightarrow D$, a hypergraph C together with two morphisms $A \rightarrow C$ and $C \rightarrow D$ is a *pushout complement* of $A \rightarrow B$ and $B \rightarrow D$ if diagram (1) in Figure 16 is a pushout.

The following lemma gives a sufficient and necessary condition for the existence of the pushout complement in case $B \rightarrow D$ is injective (see [13] for the general case), viz. the dangling condition of Definition 6. A condition for the uniqueness of pushout complements is given in Lemma 13.4.

Lemma 12 (Pushout complement construction). Let $b: A \rightarrow B$ and $f: B \hookrightarrow D$ be hypergraph morphisms such that f is injective. Then b and f possess a pushout complement if and only if no hyperedge in $E_D - f_E(E_B)$ is incident to a node in $f_V(V_B) - f_V(b_V(V_A))$. In this case a pushout complement $A \rightarrow C \hookrightarrow D$ can be constructed as follows:

- C is the subhypergraph of D with nodes $(V_D - f_V(V_B)) \cup f_V(b_V(V_A))$ and edges $(E_D - f_E(E_B)) \cup f_E(b_E(E_A))$.
- $C \hookrightarrow D$ is the inclusion of C in D .
- $A \rightarrow C$ is the restriction of $A \rightarrow B \hookrightarrow D$ to C .

Proof. Analogous to the corresponding proof for relational structures in [13]. □

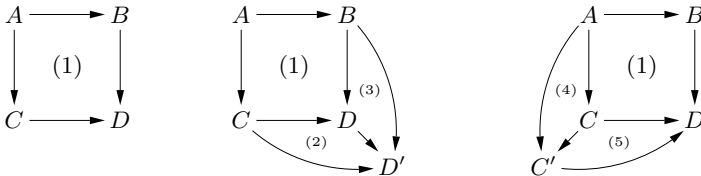


Fig. 16. Uniqueness of pushouts and pushout complements

Lemma 13 (Pushout properties). Let diagram (1) in Figure 16 be a hypergraph pushout. Then the following holds:

1. Joint surjectivity. *Each item in D has a preimage in B or C .*
2. Injectivity and surjectivity. *If $A \rightarrow B$ is injective (surjective), then $C \rightarrow D$ is injective (surjective) as well.*
3. Uniqueness of pushouts. *A hypergraph D' together with morphisms $B \rightarrow D'$ and $C \rightarrow D'$ is a pushout of $A \rightarrow B$ and $A \rightarrow C$ if and only if there is an isomorphism $D \rightarrow D'$ such that the triangles (2) and (3) in Figure 16 commute.*
4. Uniqueness of pushout complements. *If C' together with $A \rightarrow C'$ and $C' \rightarrow D$ is a pushout complement of $A \rightarrow B$ and $B \rightarrow D$, and $A \rightarrow B$ is injective, then there is an isomorphism $C \rightarrow C'$ such that the triangles (4) and (5) in Figure 16 commute.*

Proof. The first and the fourth property are shown (for graphs) in [12] and [32], respectively. The second property holds for set pushouts [1] and hence also for hypergraph pushouts. The third property holds in every category [1]. □

Lemma 14 (Composition and decomposition of pushouts). *Let the diagrams in Figure 17 consist of hypergraph morphisms. Then the following holds:*

1. *If (1) and (2) are pushouts, then (1)+(2) is a pushout.*
2. *If (1)+(2) and (1) are pushouts and (2) commutes, then (2) is a pushout.*
3. *If (1)+(2) and (2) are pushouts, (1) commutes and $B \rightarrow E$ is injective, then (1) is a pushout.*

Proof. The first two properties hold in every category, the third is proved (for graphs) in [12]. □



Fig. 17. Pushout composition and decomposition

References

1. Jiří Adámek, Horst Herrlich, and George Strecker. *Abstract and Concrete Categories*. Wiley, 1990.
2. Stefan Arnborg, Bruno Courcelle, Andrzej Proskurowski, and Detlef Seese. An algebraic theory of graph reduction. *Journal of the ACM*, 40(5):1134–1164, 1993.
3. Franz Baader and Tobias Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
4. Marc Bezem, Jan Willem Klop, and Roel de Vrijer, editors. *Term Rewriting Systems*. Cambridge University Press, 2003.
5. Hans L. Bodlaender and Babette de Fluiter. Reduction algorithms for constructing solutions in graphs with small treewidth. In *Proc. Computing and Combinatorics*, volume 1090 of *Lecture Notes in Computer Science*, pages 199–208, 1996.

6. 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.
7. Babette de Fluiter. *Algorithms for Graphs of Small Treewidth*. Dissertation, Universiteit Utrecht, 1997.
8. Hartmut Ehrig. Embedding theorems in the algebraic theory of graph-grammars. In *Proc. Fundamentals of Computation Theory*, volume 56 of *Lecture Notes in Computer Science*, pages 245–255. Springer-Verlag, 1977.
9. 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.
10. Hartmut Ehrig, Annegret Habel, Julia Padberg, and Ulrike Prange. Adhesive high-level replacement categories and systems. In *Proc. International Conference on Graph Transformation (ICGT 2004)*, volume 3256 of *Lecture Notes in Computer Science*, pages 144–160. Springer-Verlag, 2004.
11. Hartmut Ehrig and Hans-Jörg Kreowski. Parallelism of manipulations in multidimensional information structures. In *Proc. Mathematical Foundations of Computer Science*, volume 45 of *Lecture Notes in Computer Science*, pages 284–293. Springer-Verlag, 1976.
12. Hartmut Ehrig and Hans-Jörg Kreowski. Pushout-properties: An analysis of gluing constructions for graphs. *Mathematische Nachrichten*, 91:135–149, 1979.
13. 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.
14. Hartmut Ehrig, Ulrike Prange, and Gabriele Taentzer. Fundamental theory for typed attributed graph transformation. In *Proc. International Conference on Graph Transformation (ICGT 2004)*, volume 3256 of *Lecture Notes in Computer Science*, pages 161–177. Springer-Verlag, 2004.
15. Claudia Ermel, Michael Rudolf, and Gabi Taentzer. The AGG approach: Language and environment. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 2, chapter 14, pages 551–603. World Scientific, 1999.
16. Rodney Farrow, Ken Kennedy, and Linda Zucconi. Graph grammars and global program data flow analysis. In *Proc. 17th Annual Symposium on Foundations of Computer Science*, pages 42–56. IEEE, 1976.
17. Joseph A. Goguen, Jim W. Thatcher, and Eric G. Wagner. An initial algebra approach to the specification, correctness, and implementation of abstract data types. In R. T. Yeh, editor, *Current Trends in Programming Methodology, Volume 4: Data Structuring*, pages 80–149. Prentice Hall, 1978.
18. Annegret Habel, Jürgen Müller, and Detlef Plump. Double-pushout graph transformation revisited. *Mathematical Structures in Computer Science*, 11(5):637–688, 2001.
19. Reiko Heckel, Jochen Malte Küster, and Gabi Taentzer. Confluence of typed attributed graph transformation systems. In *Proc. International Conference on Graph Transformation (ICGT 2002)*, volume 2505 of *Lecture Notes in Computer Science*, pages 161–176. Springer-Verlag, 2002.
20. Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27(4):797–821, 1980.

21. Deepak Kapur, Paliath Narendran, and Friedrich Otto. On ground-confluence of term rewriting systems. *Information and Computation*, 86:14–31, 1990.
22. Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebras*, pages 263–297. Pergamon Press, 1970.
23. Hans-Jörg Kreowski. Manipulationen von Graphmanipulationen. Doctoral dissertation, Technische Universität Berlin, 1977. In German.
24. Michael Löwe and Jürgen Müller. Critical pair analysis in single-pushout graph rewriting. In Gabriel Valiente Feruglio and Francesc Roselló Llompart, editors, *Proc. Colloquium on Graph Transformation and its Application in Computer Science*, pages 71–77. Technical report B-19, Universitat de les Illes Balears, 1995.
25. M.H.A. Newman. On theories with a combinatorial definition of “equivalence”. *Annals of Mathematics*, 43(2):223–243, 1942.
26. Yasuyoshi Okada and Masahiro Hayashi. Graph rewriting systems and their application to network reliability analysis. In *Proc. Graph-Theoretic Concepts in Computer Science*, volume 570 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.
27. Detlef Plump. Hypergraph rewriting: Critical pairs and undecidability of confluence. In Ronan Sleep, Rinus Plasmeijer, and Marko van Eekelen, editors, *Term Graph Rewriting: Theory and Practice*, chapter 15, pages 201–213. John Wiley, 1993.
28. Detlef Plump. *Computing by Graph Rewriting*. Habilitation thesis, Universität Bremen, Fachbereich Mathematik und Informatik, 1999.
29. Detlef Plump and Sandra Steinert. Towards graph programs for graph algorithms. In *Proc. Int. Conference on Graph Transformation (ICGT 2004)*, volume 3256 of *Lecture Notes in Computer Science*, pages 128–143. Springer-Verlag, 2004.
30. Jean-Claude Raoult. On graph rewritings. *Theoretical Computer Science*, 32:1–24, 1984.
31. Jan Rekers and Andy Schürr. Defining and parsing visual languages with layered graph grammars. *Journal of Visual Languages and Computing*, 8(1):27–55, 1997.
32. Barry K. Rosen. Deriving graphs from graphs by applying a production. *Acta Informatica*, 4:337–357, 1975.
33. Grzegorz Rozenberg and Arto Salomaa. *Cornerstones of Undecidability*. Prentice Hall, 1994.
34. Andy Schürr, Andreas Winter, and Albert Zündorf. The PROGRES approach: Language and environment. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 2, chapter 13, pages 487–550. World Scientific, 1999.