Reversible Sesqui-Pushout Rewriting

Vincent Danos[1], Tobias Heindel[1], Ricardo Honorato-Zimmer[2], and Sandro Stucki[3]

[1] School of Informatics, University of Edinburgh.
[2] Informatics Life-Sciences Institute, University of Edinburgh.
[3] Programming Methods Laboratory (LAMP), EPFL.

Abstract. The paper proposes a variant of sesqui-pushout rewriting (SqPO) that allows one to develop the theory of nested application conditions (NACs) for arbitrary rule spans; this is a considerable generalisation compared with existing results for NACs, which only hold for linear rules (w.r.t. a suitable class of monos). Besides this main contribution, namely an adapted shifting construction for NACs, the paper presents a uniform commutativity result for a revised notion of independence that applies to arbitrary rules; these theorems hold in any category with (enough) stable pushouts and a class of monos rendering it weak adhesive HLR. To illustrate results and concepts, we use simple graphs, i.e. the category of binary endorelations and relation preserving functions, as it is a paradigmatic example of a category with stable pushouts; moreover, using regular monos to give semantics to NACs, we can shift NACs over arbitrary rule spans.

Introduction

Nested application conditions (NACs) for rules of graph transformation systems (GTSs) are a popular and intuitive means to increase the versatility of graph transformation. Tools such as AGG[4] and GROOVE[5] support a weakened form of NACs, namely negative application conditions. So far, the theory of NACs is fully developed only for double pushout (DPO) rewriting with so-called linear rules, which means that transformation operations are restricted to deletion and addition of nodes and edges; for the ubiquitous example of simple graphs, linearity is even more restrictive, namely, it is not allowed to add or delete edges between pairs of unchanged nodes.

We show that none of these restriction are necessary if we use a suitable combination of DPO and sesqui-pushout (SqPO) rewriting, which coincides with DPO for the case of linear rules (in adhesive categories [16]). More precisely, we shall extend the theory of NACs, notably the Translation Theorem [13, Theorem 6], to arbitrary spans as rules, which means that we accommodate not only for the deletion of edges between preserved nodes in simple graphs but we can also handle the operations of merging and cloning of nodes – at least, if rule applications are free of *side-effects*. Absence of side-effects will be made formal by

[4] `http://user.cs.tu-berlin.de/~gragra/agg/`
[5] `http://groove.sourceforge.net/groove-index.html`

the definition of *reversible* SqPO (SqPO^{r}) rewriting, which is the new approach that we shall propose in this paper; its definition is quite natural: it merely amounts to restricting to those SqPO-diagrams that are also SqPO-diagrams "backwards" for the reversed rule. In the end, we obtain a variation of DPO rewriting, avoiding complications involving uniqueness of pushout complements (by use of final pullback complements [8]) and thus we do not need any restriction on rules or matches any more – having the best of both worlds.

Besides the extension of the theory of NACs to arbitrary rules, we provide a suitable notion of independence for SqPO^{r} rewriting and give the corresponding commutativity result, which specialises to the existing theory for the DPO approach (with linear rules). The only categorical requirements are pullbacks and (enough) stable pushouts since we do not rely on uniqueness of pushout complements any more, which allows to drop the restriction to (left-)linear rules. Roughly, reversible SqPO-rewriting combines the controlled rewriting mechanism of DPO rewriting with the expressive power of SqPO-rewriting while being in line with the usual notions and results about independence of adjacent rule applications.

We plan to put to use our stronger version of NACs in the context of formal modeling languages for systems biology, in particular the *Kappa* language [14]. The rewrite semantics of Kappa has been formalized as a GTS over a particular category of structured graphs [6,14]. However, it seems natural to reformulate these semantics using an adhesive category [16] and NACs: some of the extra structure present in the patterns of Kappa rules intuitively specifies (positive) application conditions; moreover, matches are required to preserve so-called *free sites*, which amounts to a simple family of NACs. Kappa also supports a quantitative analysis that approximates the evolution of the expected number of occurrences of a given set of observable graphs over time using a system of differential equations [5]; if we want to formalise observables as graphs with a NAC, we also need to keep track of the change in their occurrence counts for this quantitative analysis. This is the point where the shifting constructions for NACs as formulated in the literature [13,10] are the tool of choice. One might also consider extending this type of quantitative analysis to process calculi (via graphical encodings), in which case merging rules, as supported by the SqPO approach, become relevant, e.g. for encoding substitution rules. However, as we will see in Example 6, NAC translation may break down when using the SqPO approach; this is why we consider the SqPO^{r} approach.

The final contribution of this paper, is a first tentative solution to the failure of NAC translation for the SqPO approach (see Example 6): we first construct for each SqPO-diagram the "best approximation" by a SqPO^{r}-diagram using a suitable "minimally extended" rule instance, which intuitively just contains enough additional context to make the side-effects of SqPO-rewriting (notably deletion of dangling edges) explicit; we then translate NACs to all possible SqPO-rule-instances, for which we finally can use SqPO^{r}-rewriting. Even though this solution is not effective, we conjecture that it will be viable for graph transformation systems with bounded node degree.

We illustrate the new rewriting approach and our results through examples in the category of simple graphs; in fact, that our results apply to the category of simple graphs is interesting in itself. In summary, it should become clear that the proposal of the SqPO^{r} approach to rewriting is not merely triggered by the recent interest in reversible computation, but that it is of interest for the core theory of graph transformation and contributes to versatility in applications.

Structure of the Paper We first recall the notion of (final) pullback complements [8] and sesqui-pushout (SqPO) rewriting [4] in Section 1, where we also state the relevant composition and decomposition results for the corresponding pullback squares, and recall related results on stable pushouts, which all together will be the technical backbone of our main theorems. We define reversible sesqui-pushout rewriting (SqPO^{r}) in Section 2 together with a notion of independence, for which we derive a uniform commutativity result (Theorem 1), assuming that (enough) stable pushouts exist. Then we recall the syntax and semantics of nested application conditions (NACs) in Section 3 and present our main result about NACs in Theorem 2, after describing the required categorical assumptions. In Section 4, we discuss how this result might be applied even to SqPO-rewriting. Related and future work is discussed in Section 5 where we also quickly discuss suitable categorical frameworks, before we conclude with a summary of our results in Section 6.

1 Preliminaries

A secondary theme of the present paper is the use of an algebraic approach to perform rewriting on simple graphs, which are ubiquitous in computer science and beyond, but are not as well-behaved w.r.t. algebraic graph rewriting as for example (multi-)hypergraphs; we use the following definition.

Definition 1 (Category of Simple Graphs). *A* simple graph *is a pair of sets $G = (V_G, E_G)$ where $E_G \subseteq V_G \times V_G$ is an endorelation over V_G; the elements of V_G are* nodes *or* vertices *and E_G contains all* edges *of the graph G. A* graph morphism *$f\colon G \to H$ is a function $f\colon V_G \to V_H$ such that*

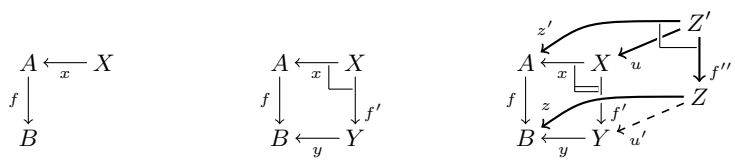
$$E_H \supseteq (f \times f)(E_G) = \left\{ \big(f(e), f(e')\big) \mid (e, e') \in E_G \right\}.$$

The category of simple graphs*, denoted by $\mathbb{G}$, has simple graphs as objects and graph morphisms as morphisms; composition and identities are given by $(f \circ g)(v) = f(g(v))$ and $\mathrm{id}_K(v) = v$ for all morphisms $f\colon G \to H$, $g\colon K \to G$ and nodes $v \in V_K$.*

The category of simple graphs will serve as running example to illustrate the concepts and results of the paper; we have chosen to keep the list of preliminary categorical concepts as short as possible. A short discussion of suitable categories of graph-like structures is given later in Section 5.

1.1 Final Pullback Complements and Stable Pushouts

The crucial concept of sesqui-pushout rewriting that goes beyond standard textbooks on category theory are *(final) pullback complements* [8]; we use the original definition in terms of the universal property illustrated on the right in Figure 1.



(a) Composable morphisms

(b) Pullback complement

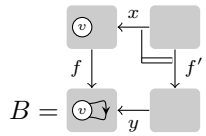
(c) Final pullback complement $y \mathrel{\dot{\cong}} \Pi_f(x)$

Fig. 1: Pullback complements for composable morphisms and finality

Definition 2 (Final Pullback Complement (FPBC)). *Let $B \leftarrow_f - A \leftarrow_x - X$ be a pair of composable morphisms (cf. Figure 1(a)). A* pullback complement *for $B \leftarrow_f - A \leftarrow_x - X$ is a pair of composable morphisms $B \leftarrow_y - Y \leftarrow_{f'} - X$ such that $A \leftarrow_x - X -_{f'}\rightarrow Y$ is a pullback of $A -_f\rightarrow B \leftarrow_y - Y$ (cf. Figure 1(b)); it is* final, *or an* FPBC *for short, if for any morphism $B \leftarrow_z - Z$, pullback $A \leftarrow_{z'} - Z' -_{f''}\rightarrow Z$ of the co-span $A -_f\rightarrow B \leftarrow_z - Z$, and morphism $u\colon Z' \rightarrow X$ satisfying $z' = x \circ u$, there exists a unique morphism $u'\colon Z \rightarrow Y$ such that $y \circ u' = z$ and $f' \circ u = u' \circ f''$ (cf. Figure 1(c), where the universally quantified morphisms are rendered as thick arrows and the dashed one denotes the unique morphism making the diagram commute). If $B \leftarrow_y - Y \leftarrow_{f'} - X$ is an* FPBC *for $B \leftarrow_f - A \leftarrow_x - X$, we write $y \mathrel{\dot{\cong}} \Pi_f(x)$.*

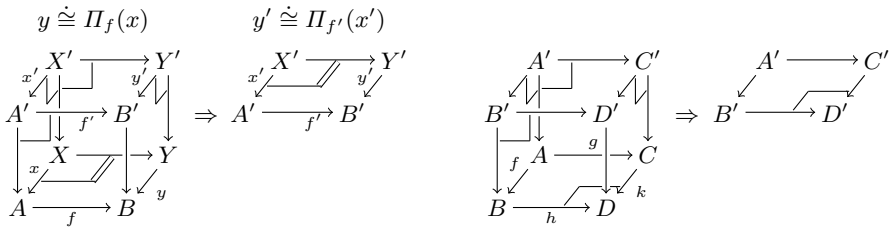
By the universal property, FPBCs are unique up to canonical isomorphism. If the composable pair $B \leftarrow_y - Y \leftarrow_{f'} - X$ is an FPBC of $B \leftarrow_f - A \leftarrow_x - X$, we mark this by a modified Freyd corner in the arising square as in Figure 1(c), i.e. we double the line that goes from the apex to the arrow f'; on several occasions, we shall refer to such squares as FPBC *squares*. The following example justifies the use of asymmetric notation.

Example 1 (Implicit Deletion as FPBC). Consider the FPBC square on the right; note that $y \mathrel{\dot{\cong}} \Pi_f(x)$ while $f \mathrel{\dot{\ncong}} \Pi_y(f') \mathrel{\dot{\cong}} \mathrm{id}_B$.



A crucial property of final pullback complements is stability w.r.t. pullbacks[6]; we shall make extensive use of it in this paper and it also essential for the good behaviour of grammar morphisms (see [1]).

[6] This corresponds to the Beck-Chevalley condition in locally cartesian closed categories.



(a) Stability of FPBCs (b) Stable Pushout

Fig. 2: Stability under pullback

Lemma 1 (Stability of FPBC). *In every cube as in Figure 2(a) that has pullback squares on all faces, if the morphism $B \leftarrow y - Y$ (on the bottom face) is the second morphism of an* FPBC *for $B \leftarrow f - A \leftarrow x - X$, then the morphism $B' \leftarrow y' - Y'$ (on top) is the second morphism of an* FPBC *for $B' \leftarrow f' - A' \leftarrow x' - X'$.*

This lemma implies that, in categories with pullbacks, any FPBC square can be pulled back along a morphism with the "tip" of the square as codomain. Before we state the consequences that we shall use in the remainder of the paper, we define pullback stability for pushouts (as it follows the same pattern of diagrams).

Definition 3 (Stable Pushouts). *Let $B \xrightarrow{h} D \leftarrow k - C$ be a pushout of the span $B \leftarrow f - A \xrightarrow{g} C$ in a category $\mathbb{C}$; the pushout is* stable *if for every commutative cube as in Figure 2(b) on the left, the top square is a pushout square if all lateral faces are pullback squares.*

Even though there are categories of graph-like structures in which some pushouts are not stable (under pullback), we generally assume that all pushouts that we operate with are stable. Our running example $\mathbb{G}$ has all pushouts and these are stable.

1.2 Splitting and Composing Pushout and FPBC Squares

We now state the relevant lemmata that allow to compose and decompose pushout and FPBC squares where composition and decomposition are also known as *pasting* and *splitting*, respectively. The reader might want to skip forward to Section 2 and come back to the remainder of the present section to look up the details, especially at a first reading. The proofs do only use diagram chasing and the defining universal properties of pushouts, pullbacks, and FPBCs (and are rather unenlightening).

The first useful fact is composition of FPBC squares, similar to pasting of pushouts and pullbacks.

Lemma 2 (FPBC Composition). *We have* (vertical) composition *of* FPBC*s as follows:*

$$\begin{tikzcd} Z \arrow[d, "z"'] & Y \arrow[l, "g'"'] \arrow[d, "y"] \arrow[dl, phantom, "\llcorner"] & X \arrow[l, "f'"'] \arrow[d, "x"] \arrow[dl, phantom, "\llcorner"] \\ C & B \arrow[l, "g"] & A \arrow[l, "f"] \end{tikzcd} \Rightarrow \begin{tikzcd} Z \arrow[d, "z"'] & Y \arrow[l, "g'"'] & X \arrow[l, "f'"'] \arrow[d, "x"] \arrow[dl, phantom, "\llcorner"] \\ C & B \arrow[l, "g"] & A \arrow[l, "f"] \end{tikzcd}$$

This means, given morphisms $f\colon A \to B$, $g\colon B \to C$, $x\colon X \to A$, *and* FPBC*s* $X \xrightarrow{f'} Y \xrightarrow{y} B$ *and* $Y \xrightarrow{g'} Z \xrightarrow{z} C$ *of* $X \xrightarrow{x} A \xrightarrow{f} B$ *and* $Y \xrightarrow{y} B \xrightarrow{g} C$, *respectively, we have* $X \xrightarrow{g' \circ f'} Z \xrightarrow{z} C$ *as* FPBC *of* $X \xrightarrow{x} A \xrightarrow{g \circ f} C$.

Besides pasting of pullback, pushout, and FPBC squares, the splitting of these squares using pullbacks is a common construction in the concurrency theory of graph transformation to derive theorems of sequential and parallel commutativity [12]. The technical tools of our commutativity result in Section 2 are the following two lemmata.

Lemma 3 (FPBC Splitting). *Let the leftmost diagram below be a pair of*

$$\begin{tikzcd} Z \arrow[d, "z"'] & Y \arrow[l, "g'"'] \arrow[d, "y"] \arrow[dl, phantom, "\llcorner"] & X \arrow[l, "f'"'] \arrow[d, "x"] \arrow[dl, phantom, "\llcorner"] \\ C & B \arrow[l, "g"] & A \arrow[l, "f"] \end{tikzcd} \;\&\; \begin{tikzcd} Z \arrow[d, "z"'] & Y \arrow[l, "g'"'] & X \arrow[l, "f'"'] \arrow[d, "x"] \arrow[dl, phantom, "\llcorner"] \\ C & B \arrow[l, "g"] & A \arrow[l, "f"] \end{tikzcd} \;\&\; \begin{tikzcd} B \arrow[d, "g"'] & A \arrow[l, "f"'] \arrow[d, "\mathrm{id}"] \arrow[dl, phantom, "\llcorner"] \\ C & A \arrow[l, "g \circ f"] \end{tikzcd} \Rightarrow \begin{tikzcd} Z \arrow[d, "z"'] & Y \arrow[l, "g'"'] \arrow[d, "y"] \arrow[dl, phantom, "\llcorner"] & X \arrow[l, "f'"'] \arrow[d, "x"] \arrow[dl, phantom, "\llcorner"] \\ C & B \arrow[l, "g"] & A \arrow[l, "f"] \end{tikzcd}$$

pullback squares such that $C \xleftarrow{z} Z \xleftarrow{g' \circ f'} X$ *is an* FPBC *of* $C \xleftarrow{g \circ f} A \xleftarrow{x} X$ *and the span* $B \xleftarrow{f} A \xrightarrow{\mathrm{id}} A$ *is a pullback of* $B \xrightarrow{g} C \xleftarrow{g \circ f} A$*; then we have* FPBC *squares as in the rightmost diagram above, i.e.* $C \xleftarrow{z} Z \xleftarrow{g'} Y$ *is an* FPBC *of* $C \xleftarrow{g} B \xleftarrow{y} Y$ *and* $B \xleftarrow{y} Y \xleftarrow{f'} X$ *is an* FPBC *of* $B \xleftarrow{f} A \xleftarrow{x} X$.

If pushouts are stable, we have a similar result for pushouts that are also pullbacks.

Lemma 4 (Pushout splitting). *Let the leftmost diagram below be a pair of*

$$\begin{tikzcd} Z \arrow[d, "z"'] & Y \arrow[l, "g'"'] \arrow[d, "y"] \arrow[dl, phantom, "\llcorner"] & X \arrow[l, "f'"'] \arrow[d, "x"] \arrow[dl, phantom, "\llcorner"] \\ C & B \arrow[l, "g"] & A \arrow[l, "f"] \end{tikzcd} \;\&\; \begin{tikzcd} Z \arrow[d, "z"'] & Y \arrow[l, "g'"'] & X \arrow[l, "f'"'] \arrow[d, "x"] \\ C & B \arrow[l, "g"] \arrow[ul, phantom, "\urcorner"] & A \arrow[l, "f"] \end{tikzcd} \;\&\; \begin{tikzcd} B \arrow[d, "g"'] & A \arrow[l, "f"'] \arrow[d, "\mathrm{id}"] \arrow[dl, phantom, "\llcorner"] \\ C & A \arrow[l, "g \circ f"] \end{tikzcd} \Rightarrow \begin{tikzcd} Z \arrow[d, "z"'] & Y \arrow[l, "g'"'] \arrow[d, "y"] & X \arrow[l, "f'"'] \arrow[d, "x"] \\ C & B \arrow[l, "g"] \arrow[ul, phantom, "\urcorner"] & A \arrow[l, "f"] \arrow[ul, phantom, "\urcorner"] \end{tikzcd}$$

pullback squares such that $Z \xrightarrow{z} C \xleftarrow{g \circ f} A$ *is a pushout of* $Z \xleftarrow{g' \circ f'} X \xrightarrow{x} A$ *that is pullback stable and* $B \xleftarrow{f} A \xrightarrow{\mathrm{id}} A$ *is a pullback of* $B \xrightarrow{g} C \xleftarrow{g \circ f} A$*; then the first two pullback squares are also pushout squares, i.e.* $Z \xrightarrow{z} C \xleftarrow{g} B$ *is a pushout of* $Z \xleftarrow{g'} Y \xrightarrow{y} B$ *and* $Y \xrightarrow{y} B \xleftarrow{f} A$ *is a pushout of* $Y \xleftarrow{f'} X \xrightarrow{x} A$.

Remark 1. If the morphism g in Lemma 3 (resp. Lemma 4) is a mono, the third assumption is trivially true (cf. [16, Lemma 4.6], for pushout splitting).

The generality of Lemmata 3 and 4 is tailored to fit exactly our new examples of independence in rewriting in Section 2.2, which feature both merging and cloning of nodes.

1.3 Finitely Powered Objects

In virtually all applications, objects of rewriting and rules are suitably finite. More precisely, in Section 4, we shall restrict to objects that are *finitely powered*, i.e. objects shall have only finitely many different subobjects, where a *subobject* of an object $A \in \mathbb{C}$ is an isomorphism class $[m]$ in the slice category $\mathbb{C}{\downarrow}A$, of some mono $m\colon M \rightarrowtail A$.

2 Reversible Sesqui-Pushout Rewriting

The central definition of the present paper is the reversible variant of sesqui-pushout rewriting (SqPO) [4]. It generalises the very controlled rewriting mechanism of DPO rewriting [3] to arbitrary rules as used in SqPO rewriting, i.e. any span can be used as a rule. As a result, we can perform DPO rewriting with duplication of entities as in SqPO rewriting; moreover, we also can lift the theory of application conditions and constraints [13], as we shall do in Section 3.

2.1 Definition and First Examples

The definition of reversible sesqui-pushout rewriting is trivial: we just require a double square diagram that is a sesqui-pushout diagram forwards and backwards. The benefits of this approach over the original one, besides being in line with the recent trend of reversible computation, will become clear when we discuss the theory of nested application conditions and its limitations in the SqPO approach.

Definition 4 (Reversible Sesqui-Pushout Rewriting). *A* rule *is any span of morphisms $L \leftarrow\alpha- K -\beta\rightarrow R$, i.e. any pair of morphisms sharing their domain; the* reversal *of a given rule $\rho = L \leftarrow\alpha- K -\beta\rightarrow R$, written ρ^{r}, is the rule $R \leftarrow\beta- K -\alpha\rightarrow L$. Let $\rho = L \leftarrow\alpha- K -\beta\rightarrow R$ be a rule, and let $m\colon L \rightarrow A$ be a morphism; an* SqPO-diagram *for ρ at m is a diagram as in* (1) *on the left*

$$\begin{tikzcd} L \arrow[d, "m"'] & K \arrow[l, "\alpha"'] \arrow[r, "\beta"] \arrow[d, "o"] \arrow[dl, phantom, "\llcorner"] & R \arrow[d, "n"] \\ A & D \arrow[l, "\gamma"] \arrow[r, "\delta"'] & B \arrow[ul, phantom, "\ulcorner"] \end{tikzcd} \qquad \begin{tikzcd} L \arrow[d, "m"'] & K \arrow[l, "\alpha"'] \arrow[r, "\beta"] \arrow[d, "o"] \arrow[dl, phantom, "\llcorner"] \arrow[dl, phantom, "\llcorner"] & R \arrow[d, "n"] \\ A & D \arrow[l, "\gamma"] \arrow[r, "\delta"'] & B \arrow[ul, phantom, "\ulcorner"] \end{tikzcd} \tag{1}$$

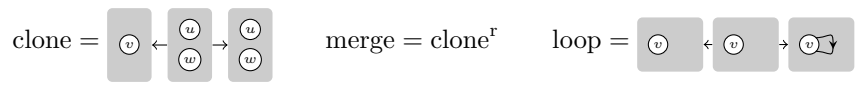
such that $A \leftarrow\gamma- D \leftarrow o- K$ is an FPBC *of $A \leftarrow m- L \leftarrow\alpha- K$ and $D -\delta\rightarrow B \leftarrow n- R$ a pushout of $D \leftarrow o- K -\beta\rightarrow R$; in such a diagram, the morphism m is called the* match *for ρ, and n is the* co-match. *Diagram* (1) *is* reversible *or an* SqPO^{r}-diagram *if it is also an* SqPO*-diagram for ρ^{r} with match n and co-match m, i.e. we have a diagram as in* (1) *on the right; if we have such a* SqPO^{r}*-diagram, the match m is called* side-effect-free *for ρ or just* reversible. *We write $A \models\langle\rho,m\rangle\Rightarrow B$ or simply $A \models\rho\Rightarrow B$ if there exists a* SqPO^{r}*-diagram as in* (1) *on the right (where $\rho = L \leftarrow\alpha- K -\beta\rightarrow R$ is the rule and $m\colon L \rightarrow A$ is the match).*

If we have $A \models\langle\rho,m\rangle\Rightarrow B$, we also speak of a *rule application* (of ρ at m), or say that rule ρ rewrites A to B at match m.

Remark 2. In adhesive categories [16], the SqPO approach coincides with the double pushout approach [4] for linear rules, i.e. for rules consisting of pairs of monos.

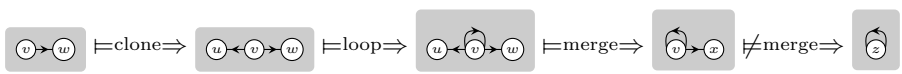
As a consequence of this observation, new examples of SqPO rewriting either have non-linear rules or take place in a category that is not adhesive. This is another reason why we have chosen simple graphs as running example; it is the paradigmatic example of a category that is neither adhesive nor rm-adhesive [11].

Example 2 (Cloning and Merging in Graph Rewriting). Consider the rules described in (2) where all graph morphisms are uniquely determined except for the right morphism of the clone rule, which we take to be the identity.



(2)

We have the following applications of these rules.



In each case, the match is uniquely determined by the effect on the graphs: first, we clone node w, obtaining its *clone* u with the same local connectivity, then we add a loop at node v, and finally we merge u with w. At the end of the above display, the merge-rule cannot be applied to the last graph as this would use an irreversible match: applying the reverse rule (cloning node z) would result in the completely connected graph on two nodes rather than the original graph.

The rule merge is not merely the reversal of clone, it may in fact serve as its inverse; more precisely, the composite rule clerge := ($v \leftarrow vu \rightarrow v$) has no effect if applied using SqPO^{r} rewriting, due to symmetry of SqPO^{r}-diagrams. Note that the category of simple graphs is somewhat peculiar as the clerge-rule can always be applied.[7]

2.2 Independence and Commutativity

The generality of arbitrary spans as rules necessitates an adaptation of the usual notion of independence of rewriting diagrams to obtain the expected commutativity result that allows to "switch" adjacent diagrams if they are *independent* (see Theorem 1 below).

Definition 5 (Independence). *Let* $\rho_i = L_i \leftarrow_{\alpha_i} K_i \rightarrow_{\beta_i} R_i$ *(*$i = 1, 2$*) be spans*

$$\begin{tikzcd}
L_1 \arrow[d, "m_1"'] & K_1 \arrow[l, "\alpha_1"'] \arrow[r, "\beta_1"] \arrow[d, "o_1"] & R_1 \arrow[d, "n_1"] \\
A_1 & D_1 \arrow[l, "\gamma_1"] \arrow[r, "\delta_1"'] & B_1
\end{tikzcd}
\quad
\begin{tikzcd}
L_2 \arrow[d, "m_2"'] & K_2 \arrow[l, "\alpha_2"'] \arrow[r, "\beta_2"] \arrow[d, "o_2"] & R_2 \arrow[d, "n_2"] \\
A_2 & D_2 \arrow[l, "\gamma_2"] \arrow[r, "\delta_2"'] & B_2
\end{tikzcd}
\quad
\begin{tikzcd}
L_2 \arrow[d, "\overline{m}_2 := \gamma_1 \circ b"'] & L_2 \arrow[l, "\mathrm{id}"'] \arrow[r, "\mathrm{id}"] \arrow[d, "b"] \arrow[ul, phantom, "(\dagger)"] \arrow[ur, phantom, "(\ddagger)"] & L_2 \arrow[d, "m_2"] \\
A_1 & D_1 \arrow[l, "\gamma_1"] \arrow[r, "\delta_1"'] & B_1
\end{tikzcd}
\tag{3}$$

[7] In contrast, in the category of multi-graphs, clerge can only be applied to isolated nodes.

and let the diagram on the left in (3) *be a pair of* SqPO^{r}*-diagrams such that* $B_1 = A_2$*. A* left witness *for these diagrams is a morphism* $D_1 \xleftarrow{b} L_2$ *such that*

- $\overline{m}_2 := \gamma_1 \circ b$ *is a reversible match for* ρ_2*, and*
- $b \cong \delta_1^*(m_2)$ *and* $b \cong \gamma_1^*(\overline{m}_2)$*, i.e. we have the pullbacks on the right in* (3)*.*

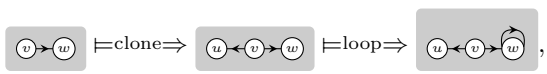
A right witness *is a left witness for the "mirrored" situation, that is, a morphism* $d\colon R_1 \to D_2$ *such that*

- $\underline{n}_1 := \delta_2 \circ d$ *is a reversible match for* ${\rho_1}^{\mathrm{r}}$*, and*
- $d \cong \gamma_2^*(n_1)$ *and* $d \cong \delta_1^*(\underline{n}_1)$*.*

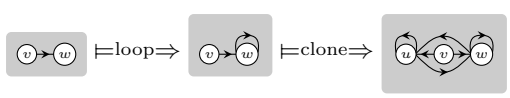
The pair of SqPO^{r}*-diagrams in* (3) *is* independent *if there exist a left and a right witness.*

The pullback requirements in (3) are vacuous if γ_1 and δ_1 are monos, i.e. commutativity of Squares (†) and (‡) is sufficient. The following example illustrates the subtle interplay of node merging and cloning with addition of loops.

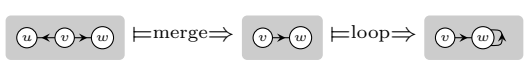
Example 3. In the following pair of rule applications,



cloning node w is not independent of adding a loop at node w as the local structure changes. We can check formally that there is no left witness for the corresponding pair of SqPO^{r}-diagrams, because there is no suitable morphism making Square (†) in (3) a pullback square; nevertheless, the rules can be applied in the reverse order, but we obtain a different result.



The role of Square (‡) in (3) features in the following pair of rule applications.



Nodes u and w can only be merged as long as they have the same "neighbourhood", and adding a loop to w changes the local connectivity of w; formally, there is again no left witness, as we cannot find a suitable morphism making Square (‡) a pullback square. Adding a loop at either one of the nodes u and w makes merging u with w impossible when following the SqPO^{r} approach.

Remark 3. For the case of adhesive categories and linear rules (i.e. rules that consist of a pair of monos), commutativity of the squares on the right in (3) is enough (as mentioned above); moreover, in adhesive categories, the requirement that left witnesses are reversible matches is automatically true and we recover the usual definition of independence (of the DPO approach with linear rules).

The definition of independence is chosen sufficiently strong to obtain the following general theorem of commutativity of derivations; its proof is similar to the standard results of the literature (see, e.g. [12]), using the lemmata of Section 1.

Theorem 1 (Commutativity). *Let $\mathbb{C}$ be a category that has pullbacks and pushouts such that all pushouts are stable. Then for each pair of independent* SqPO^{r}*-diagrams with left and right witness as below on the left*

$$\begin{tikzcd}
L_1 \arrow[d, "m_1"'] & K_1 \arrow[l, "\alpha_1"'] \arrow[r, "\beta_1"] \arrow[d, "o_1"] & R_1 \arrow[d, "n_1"] & L_2 \arrow[l, "\alpha_2"'] \arrow[d, "m_2"] & K_2 \arrow[r, "\beta_2"] \arrow[d, "o_2"] & R_2 \arrow[d, "n_2"] \\
A_1 & D_1 \arrow[l, "\gamma_1"] \arrow[r, "\delta_1"'] & B_1 \arrow[r, equal] & A_2 \arrow[l, "\gamma_2"] & D_2 \arrow[r, "\delta_2"'] & B_2
\end{tikzcd}
\quad \rightsquigarrow \quad
\begin{tikzcd}
L_2 \arrow[d, "\overline{m}_2"'] & K_2 \arrow[l, "\alpha_2"'] \arrow[r, "\beta_2"] \arrow[d, "o'_2"] & R_2 \arrow[d, "n'_2"] & L_1 \arrow[d, "m'_1"] & K_1 \arrow[l, "\alpha_1"'] \arrow[r, "\beta_1"] \arrow[d, "o'_1"] & R_1 \arrow[d, "\underline{n}_1"] \\
A_1 & E_2 \arrow[l, "\gamma'_2"] \arrow[r, "\delta'_2"'] & C \arrow[r, equal] & C & E_1 \arrow[l, "\gamma'_1"] \arrow[r, "\delta'_1"'] & B_2
\end{tikzcd}$$

there exists a corresponding pair of independent SqPO^{r}*-diagrams as above on the right with $\overline{m}_2 = \gamma_1 \circ b$ and $\underline{n}_1 = \delta_2 \circ d$, reversing the order of rule application.*

The commutativity result for pairs of independent diagrams is interesting in itself, as it applies to arbitrary rules, improves over previous results [4], and is completely symmetric. However, the main motivation to introduce the SqPO^{r} approach is to make the theory of applications conditions available for arbitrary rule spans.

3 On Nested Application Conditions

Application conditions for rules are an elegant and intuitive means to restrict the allowed matches of each rule individually. We give a short review of nested application conditions before we demonstrate that conditions can be moved freely between left and right-hand sides of rules as in previous work on DPO rewriting with linear rules [13,10]. Before we recall the full definition of NACs, we informally describe a simple example.

Example 4. We might want to apply the loop rule only at those matches that map to a node without any incoming edge, and we illustrate this application condition by adding a "forbidden" dashed edge.

Though most examples only require simple negative application conditions, one occasionally encounters situations where one wants the full generality of *nested application conditions* to restrict matches of certain rules, individually.

Definition 6 (Nested Application Condition). *A* nested application condition (NAC) *c on an object $P \in \mathbb{C}$, written $c \rhd P$ or $P \lhd c$, is defined inductively as follows.*

Base Case *The* trivial NAC *is* $\mathtt{tt} \rhd P$.
Inductive Steps *There are three constructors for* NACs.

Existential Morphism $\exists(a, c') \rhd P$ *is a* NAC *on* P *if* $a\colon P \to Q$ *in* $\mathbb{C}$ *is a morphism and* $c' \rhd Q$ *is a* NAC *on* Q.

Negation $\neg c' \rhd P$ *is a* NAC *on* P *if* $c' \rhd P$ *is so.*

Conjunction $\bigwedge_{i\in\mathcal{I}} c_i \rhd P$ *is a* NAC *if* $\{c_i \rhd P\}_{i\in\mathcal{I}}$ *is a family of* NACs *on* P *indexed over a non-empty set* $\mathcal{I} \neq \varnothing$.

The negative application condition that we described in Example 4 can be formulated as $\neg\exists(\textcircled{v} \text{-}\subseteq\!\to \textcircled{w} \rightarrowtail \textcircled{v}, \mathtt{tt})$ on the left-hand side of the loop-rule.

Concerning the semantics of NACs, several routes have been taken in the literature; the next definition strikes a compromise between generality and relevance for the present paper; we write $|P{\downarrow}\mathbb{C}|$ for the collection of all $\mathbb{C}$-morphisms with domain P, for any $P \in \mathbb{C}$.

Definition 7 (Semantics of NACs). *Let* $\mathcal{X}$ *be a collection of morphisms in* $\mathbb{C}$*, referred to as* splitting set*; for each* NAC $c \rhd P$*, its set of* instances*, denoted by* $[\![c \rhd P]\!] \subseteq |P{\downarrow}\mathbb{C}|$*, is defined by mutual recursion as follows.*

- *For every object* $P \in \mathbb{C}$*, we define* $[\![\mathtt{tt} \rhd P]\!] = |P{\downarrow}\mathbb{C}|$.
- *If* $c = \exists(a, c') \rhd P$ *with* $a\colon P \to Q$ *in* $\mathbb{C}$ *and* $c' \rhd Q$ *a* NAC*, we define*
$$[\![\exists(a, c') \rhd P]\!] = \big([\![c' \rhd Q]\!] \cap \mathcal{X}\big) \circ a := \Big\{ n \circ a \;\Big|\; (Q \xrightarrow{n} A) \in [\![c' \rhd Q]\!] \cap \mathcal{X} \Big\}.$$
- *If* $c = \neg c' \rhd P$ *with* $c' \rhd P$ *a* NAC*, we define*
$$[\![\neg c' \rhd P]\!] = |P{\downarrow}\mathbb{C}| \setminus [\![c' \rhd P]\!]\,.$$
- *If* $c = \bigwedge_{i\in\mathcal{I}} c_i \rhd P$ *with* $\{c_i \rhd P\}_{i\in\mathcal{I}}$ *a family of* NACs*, we define*
$$\Big[\!\!\Big[\bigwedge_{i\in\mathcal{I}} c_i \rhd P\Big]\!\!\Big] = \bigcap_{i\in\mathcal{I}} [\![c_i \rhd P]\!]\,.$$

We write $m \vDash^{\mathcal{X}} c \rhd P$*, or* $m \vDash^{\mathcal{X}} c$ *for short, if* $m \in [\![c \rhd P]\!]$.

Note that the splitting set features only at one place in this definition, namely in the clause that gives semantics to the $\exists$-constructor.[8] Every element of $[\![\exists(a, c') \rhd P]\!]$ is factored as an element of the splitting set $\mathcal{X}$ after a, thus "splitting off" from each candidate $f \in |P{\downarrow}\mathbb{C}|$ some morphism in $\mathcal{X}$ that satisfies c'. A second reason for the name *splitting set* is its function in Theorem 2, where we shall decompose rewriting diagrams using pushout splitting for which we need splitting sets to be *robust*.

Definition 8 (Robust Splitting Set). *A collection of monos* $\mathcal{M}$ *in a category* $\mathbb{C}$ *is a* robust splitting set *if*

- *the set* $\mathcal{M}$ *contains all identities and is closed under composition,*
- *the category* $\mathbb{C}$ *has pushouts and pullbacks along* $\mathcal{M}$*,*
- *the set* $\mathcal{M}$ *is stable under pushout and pullback, and*

[8] There is indeed a hidden existential quantifier in the definiens of $[\![\exists(a, c') \rhd P]\!]$, namely $\big\{ n \circ a \;\big|\; n \in [\![c' \rhd Q]\!] \cap \mathcal{X} \big\} = \big\{ f \in |P{\downarrow}\mathbb{C}| \;\big|\; \exists n \in [\![c' \rhd Q]\!] \cap \mathcal{X}.\, f = n \circ a \big\}$.

– *pushouts along morphisms in $\mathcal{M}$ yield pullback squares.*

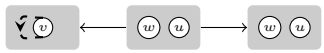
The reader that is familiar with adhesive categories and related concepts [9] will recognise these properties (see also Section 5).

For our running example $\mathbb{G}$, we chose the class of regular monos as splitting set, where a mono $m\colon G \to H$ is regular iff it reflects edges, i.e. if it satisfies the equation $(m \times m)(E_G) = E_H \cap \big(m(V_G) \times m(V_G)\big)$.

Lemma 5. *Regular monos are a robust splitting set.*

The use of regular monos as splitting set has some subtle consequences.

Example 5 (Cloning Nodes without Loops). Consider the clone-rule with the application condition that the node to be cloned does not have a loop.



The corresponding NAC for the clone-rule is $c_{\neg\circlearrowleft} := \neg\exists(\textcircled{v} \xrightarrow{\subseteq} \textcircled{v}\circlearrowleft, \mathtt{tt})$; however, (ab)using the fact that we have fixed regular monos as our splitting set, we could as well use the following NAC $c_\flat := \neg\exists(\textcircled{v} \xrightarrow{\mathrm{id}} \textcircled{v}, \mathtt{tt})$ (where the morphism is the identity on a single node). The two are equivalent, in the sense that $[\![c_{\neg\circlearrowleft}]\!] = [\![c_\flat]\!]$. While this example is admittedly somewhat pathological, it illustrates the functioning of the regular monos as the splitting set.

Our main result about NACs is the following (cf. [13, Theorem 6] and [10, Lemma 3]).

Theorem 2 (NAC translation). *Let $\mathcal{M}$ be a robust splitting set in a category with pushouts; then for each rule $\rho = L \xleftarrow{\alpha} K \xrightarrow{\beta} R$, and every* NAC *$c \rhd R$ (resp. $\bar{c} \rhd L$), there exists a* NAC *$c_\rho \rhd L$ (resp. $\bar{c}_{\rho^{\mathrm{r}}} \rhd R$) such that for every* SqPO^{r}*-diagram*

$$\begin{tikzcd}
c_\rho \rhd L \arrow[d, "m"'] & K \arrow[l, "\alpha"'] \arrow[r, "\beta"] \arrow[d, "o"] \arrow[dl, phantom, "\lrcorner"] \arrow[dr, phantom, "\lrcorner"] & R \lhd c \arrow[d, "n"] \\
A & D \arrow[l, "\gamma"] \arrow[r, "\delta"'] & B
\end{tikzcd}$$

we have $n \vDash^{\mathcal{M}} c$ iff $m \vDash^{\mathcal{M}} c_\rho$ (resp. $m \vDash^{\mathcal{M}} \bar{c}$ iff $n \vDash^{\mathcal{M}} \bar{c}_{\rho^{\mathrm{r}}}$).

Much more interesting than repeating all minute details of the translation of NACs is the fact that there is no hope that exactly the same result could be obtained for SqPO-rewriting, as illustrated by the following example.

Example 6 (Failure of NAC *Translation).* Applying the loop-rule (using plain SqPO) to $\textcircled{v}$ or to $\textcircled{v}\circlearrowleft$ we get the same result, and thus the same co-match; note that the match into $\textcircled{v}\circlearrowleft$ is not reversible. Adding the NAC $\neg\exists(\textcircled{v} \xrightarrow{\mathrm{id}} \textcircled{v}, \mathtt{tt})$ to the loop-rule forbids application to the graph $\textcircled{v}\circlearrowleft$ but it can still be be applied to $\textcircled{v}$. Thus, a corresponding equivalent application condition on the right-hand side of the loop-rule cannot exist, because either it admits the comatch into $\textcircled{v}\circlearrowleft$ or not, but in both cases it does not distinguish between the two matches. Hence, we have shown that $\neg\exists(\textcircled{v} \xrightarrow{\mathrm{id}} \textcircled{v}, \mathtt{tt})$ does not have any exact counterpart on the right-hand side in the sense of Theorem 2.

It is not obvious how one could work around this counter-example in general. However, for the case of monic matches, the next section describes how Theorem 2 can be applied even to SqPO-rewriting, at least in favourable cases.

4 On Reversibility of Sesqui-Pushout Rewriting

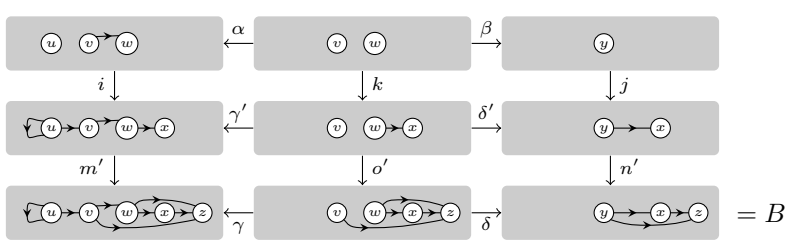
After the generalisation of key results of DPO rewriting to SqPO^{r} rewriting, we shall describe a method that gives for each SqPO-diagram (with monic match and co-match) a minimal rule-instance that can be used to achieve the same rewriting effect using SqPO^{r}-rewriting. This method can be understood as a natural measure for how far away a SqPO-derivation is from being reversible. More precisely, we can divide each SqPO direct derivations into a SqPO-derivation (the *rule-instantiation*) over a SqPO^{r}-diagram such that vertical composition of the FPBC and pushout squares yield the original SqPO-diagram. Intuitively, the rule-instance contains just enough additional context to cover the side-effects of the original SqPO-diagram.

Theorem 3 (Instantiation Thoerem). *Let $\mathbb{C}$ be a category with pullbacks in which pushouts are stable; then for each* SqPO*-diagram with match $m\colon L \to A$ such that A is finitely powered, there exists a least subobject $[m'\colon L' \rightarrowtail A]$ of A such that the match m factors as $m = m' \circ i$ (for a unique i) and the original* SqPO*-diagram is the vertical composition of an* SqPO*-diagram with match i over a* SqPO^{r}*-diagram with match m', i.e. we have the following diagram.*

$$\begin{tikzcd}
L \arrow[d, "m"'] & K \arrow[l, "\alpha"'] \arrow[r, "\beta"] \arrow[d, "o"] \arrow[dl, phantom, "\ulcorner"] & R \arrow[d, "n"] \arrow[dl, phantom, "\ulcorner"] \\
A & D \arrow[l, "\gamma"] \arrow[r, "\delta"'] & B
\end{tikzcd}
\quad \rightsquigarrow \quad
\begin{tikzcd}
L \arrow[d, "i"'] & K \arrow[l, "\alpha"'] \arrow[r, "\beta"] \arrow[d, "k"] \arrow[dl, phantom, "\ulcorner"] & R \arrow[d, "j"] \arrow[dl, phantom, "\ulcorner"] \\
L' \arrow[d, "m'"'] & K' \arrow[l, "\gamma'"'] \arrow[r, "\delta'"'] \arrow[d, "o'"] \arrow[dl, phantom, "\ulcorner"] & R' \arrow[d, "n'"] \arrow[dl, phantom, "\ulcorner"] \\
A & D \arrow[l, "\gamma"] \arrow[r, "\delta"'] & B
\end{tikzcd}$$

For the case of simple graphs, we illustrate the idea that the construction of the (proof of the) theorem just adds minimal context in the neighbourhood of nodes that are merged or deleted.[9]

Example 7. The below diagram (where i, m', and α are inclusions) is an example for a decomposition of an SqPO-diagram according to Theorem 3.



[9] For the case of multi-graphs, the situation is slightly different because merging is in general not the inverse operation to cloning.

It is important to remember that we are working in the category of simple graphs because the outer double square diagram would not be an SqPO-diagram in the category of multi-graphs, as there would be an additional edge from y to z in B. The upper row is an SqPO-diagram and the lower row a SqPO^{r}-diagram; moreover, the subobject $[m']$ is minimal in the sense of Theorem 3, which we will explain using the metaphor of minimal context.

We want to argue (informally) that $[m']$ is obtained by just adding enough context to the left-hand side L. First, we have to add the edges at node u, which have been left "dangling" in the left square/column of the outer SqPO-diagram; second, and slightly more intricate, we also need to add the node x and the edge (y, x) because merging nodes v and w (in the right column) has "side-effects" as the nodes v and w differ in their local structure w.r.t. node x in B. We do not need to add z, as its connectivity to nodes v and w is the same.

As announced before, we can now lift Theorem 2 to SqPO-rewriting, as follows. For any rule $\rho = L \leftarrow\alpha- K -\beta\rightarrow R$ with NAC $c \rhd L$ and rule application $A \models\langle\rho,m\rangle\Rightarrow B$, we can use Theorem 3 to factor the match into $m = m' \circ i$, obtaining a rule instance $\rho' = L' \leftarrow\alpha'- K' -\beta'\rightarrow R'$. Now we can use [13, Corollary 3] to shift $c \rhd L$ from ρ to ρ' along i, i.e. there is a NAC $i(c) \rhd L'$ such that, for every morphism $f\colon L \rightarrow X$, we have $f \models^{\mathcal{M}} i(c)$ iff $f \circ i \models^{\mathcal{M}} c$; finally, we apply the construction of Theorem 2 to ρ' and thus obtain $c'_i \rhd R'$.

In favourable cases, there are only a finite number of rule-instances that matter for a given graph transformation system. In fact, we expect this to be the case for systems where all reachable graphs have bounded node degree and SqPO^{r}-diagrams are required to have monic matches and co-matches. If there are only finitely many (relevant) rule instances, we can "compile" a set of SqPO-rules into an equivalent set of SqPO^{r}-rules. This means that, at least in favourable cases, the theory of NACs might even be applicable for general SqPO-rewriting.

5 Related and Future Work

The properties of robust splitting sets (see Definition 8) are reminiscent of vertical weak adhesive HLR categories (see [9] for an overview of this and related concepts), which more or less exactly fulfil these requirements. To obtain the commutativity result of Theorem 1, even if we restrict matches and co-matches of SqPO^{r}-diagrams to the relevant class of monos, we need the additional requirement of stability of these pushouts under pullback. However, in our running example $\mathbb{G}$, we do not need to restrict to regular matches as actually all pushouts are stable; we take this as evidence that SqPO^{r}-rewriting calls for a revision of the categorical frameworks for graph rewriting.

Concerning other span-based rewriting approaches besides SqPO-rewriting, we mention [17,7,18]; SqPO^{r}-rewriting is probably best understood as a very restricted instance of these (for a restricted notion of rule in the case of [18]) since none of these approaches requires a double pushout diagram. A concrete example where the difference becomes apparent is the last forbidden application

of the merge rule in Example 2: it is forbidden by SqPO^{r}-rewriting but would be allowed by all of the three other approaches [17,7,18].

As future work, a thorough comparison with recent proposals for the transformation of simple graphs is in place; e.g. we conjecture that for simple graphs, SqPO^{r}-diagrams comprise a pair of minimal pushout complements [2], using spans of (arbitrary) monos for rules. Finally, we plan a fundamental study of SqPO-rewriting and SqPO^{r}-rewriting in the span bi-category to unify the main algebraic approaches to rewriting by universal properties in the span-bicategory whenever the category of rewriting is suitably well-behaved, following up on previous work on adhesive categories [15].

6 Conclusion

We have proposed SqPO^{r}-rewriting as a symmetric variant of SqPO-rewriting, which ensures that each rule application has a corresponding inverse application using the reversed rule. We have established a commutativity result, which incorporates sequential and parallel commutativity into a single theorem. Furthermore, we have shown that nested application conditions can be expected to function in the same way as for double pushout rewriting with linear rules as our Theorem 2 generalises a corner stone of the theory of NACs if we have a suitable class of monos that endows the category of rewriting with the structure of a vertical weak adhesive HLR category. Finally we have given a construction that allows to give for each SqPO-diagram (with monic match and co-match) the "best approximation" by a SqPO^{r}-diagram using a minimally extended rule-instance, which makes Theorem 2 even applicable to SqPO-rewriting. All these results hold in the category of simple graphs with the class of regular monos as splitting set; we take this as a promising indicator that SqPO^{r}-rewriting should be studied in more detail, for the particular case of simple graphs but also in the context of categorical frameworks for graph rewriting.

Acknowledgements Thanks to Ilias Garnier for help finding the term *splitting set* and feed-back on an early draft of this paper. We also thank the anonymous reviewers for their insightful comments.

References

1. Baldan, P., Corradini, A., Heindel, T., König, B., Sobociński, P.: Unfolding Grammars in Adhesive Categories. In: Proceedings of the 3rd International Conference on Algebra and Coalgebra in Computer Science. pp. 350–366. Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg (2009)
2. Braatz, B., Golas, U., Soboll, T.: How to delete categorically — Two pushout complement constructions. Journal of Symbolic Computation 46(3), 246 – 271 (2011), applied and Computational Category Theory
3. Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R., Löwe, M.: Handbook of graph grammars and computing by graph transformation. chap. Algebraic

Approaches to Graph Transformation. Part I: Basic Concepts and Double Pushout Approach, pp. 163–245. World Scientific Publishing Co., Inc., River Edge, NJ, USA (1997)
4. Corradini, A., Heindel, T., Hermann, F., König, B.: Sesqui-pushout Rewriting. In: Proceedings of the Third International Conference on Graph Transformations. pp. 30–45. Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg (2006)
5. Danos, V., Feret, J., Fontana, W., Harmer, R., Krivine, J.: Abstracting the differential semantics of rule-based models: Exact and automated model reduction. In: Logic in Computer Science (LICS), 2010 25th Annual IEEE Symposium on. pp. 362–381 (July 2010)
6. Danos, V., Feret, J., Fontana, W., Harmer, R., Hayman, J., Krivine, J., Thompson-Walsh, C.D., Winskel, G.: Graphs, rewriting and pathway reconstruction for rule-based models. In: D'Souza, D., Kavitha, T., Radhakrishnan, J. (eds.) FSTTCS. LIPIcs, vol. 18, pp. 276–288. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2012)
7. Duval, D., Echahed, R., Prost, F.: Graph rewriting with polarized cloning. CoRR abs/0911.3786 (2009)
8. Dyckhoff, R., Tholen, W.: Exponentiable morphisms, partial products and pullback complements. Journal of Pure and Applied Algebra 49(1-2), 103–116 (1987)
9. Ehrig, H., Golas, U., Hermann, F.: Categorical Frameworks for Graph Transformation and HLR Systems Based on the DPO Approach. Bulletin of the EATCS 102, 111–121 (2010)
10. Ehrig, H., Habel, A., Lambers, L.: Parallelism and concurrency theorems for rules with nested application conditions. ECEASST 26 (2010)
11. Garner, R., Lack, S.: On the axioms for adhesive and quasiadhesive categories. Theory and Applications of Categories 27(3), 27–46 (2012)
12. Habel, A., Müller, J., Plump, D.: Double-pushout Graph Transformation Revisited. Mathematical Structures in Computer Science 11(5), 637–688 (Oct 2001)
13. Habel, A., Pennemann, K.H.: Correctness of high-level transformation systems relative to nested conditions. Mathematical Structures in Computer Science 19(2), 245–296 (2009)
14. Hayman, J., Heindel, T.: Pattern graphs and rule-based models: The semantics of kappa. In: Pfenning, F. (ed.) Foundations of Software Science and Computation Structures, Lecture Notes in Computer Science, vol. 7794, pp. 1–16. Springer Berlin Heidelberg (2013)
15. Heindel, T., Sobociński, P.: Being Van Kampen is a universal property. Logical Methods in Computer Science 7(1) (2011)
16. Lack, S., Sobociński, P.: Adhesive and quasiadhesive categories. RAIRO – Theoretical Informatics and Applications 39(3), 511–545 (2005)
17. Löwe, M.: Refined graph rewriting in span-categories: A framework for algebraic graph transformation. In: Proceedings of the 6th International Conference on Graph Transformations. pp. 111–125. Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg (2012)
18. Monserrat, M., Rosselló, F., Torrens, J., Valiente, G.: Single-pushout rewriting in categories of spans I: The general setting. Tech. rep., Informe d'investigació, Department of Software (LSI) Universitat Politècnica de Catalunya (1997)