Adhesivity is not enough: Local Church-Rosser revisited*

Paolo Baldan[1], Fabio Gadducci[2], and Pawel Sobociński[3]

[1] Dipartimento di Matematica Pura e Applicata, Università di Padova
[2] Dipartimento di Informatica, Università di Pisa
[3] School of Electronics and Computer Science, University of Southampton

Abstract. Adhesive categories provide an abstract setting for the double-pushout approach to rewriting, generalising classical approaches to graph transformation. Fundamental results about parallelism and confluence, including the local Church-Rosser theorem, can be proven in adhesive categories, provided that one restricts to linear rules. We identify a class of categories, including most adhesive categories used in rewriting, where those same results can be proven in the presence of rules that are merely left-linear, i.e., rules which can merge different parts of a rewritten object. Such rules naturally emerge, e.g., when using graphical encodings for modelling the operational semantics of process calculi.

Keywords: Adhesive and extensive categories, double-pushout rewriting, local Church-Rosser property, parallel and sequential independence.

Introduction

The strength of graph transformation formalisms in the specification of distributed and concurrent systems lies on the relative ease of use, due to their visual nature. However, equally relevant is the fact that these formalisms often come equipped with a rich theory of concurrency, including confluence properties that can be pivotal in developing suitable verification techniques.

Focusing on the double pushout (DPO) approach to graph transformation, *parallel* and *sequential independence* are central properties in the corresponding concurrency theory. While the former is essentially a local confluence property, the latter identifies suitable conditions for establishing whenever two rewrite steps, even though performed in sequence, do not interfere with each other and thus can be potentially applied in any order (as well as concurrently).

The local Church-Rosser theorem tightly connects parallel and sequential independence: two sequentially independent steps can be applied to the same start graph, resulting in a pair of parallel independent steps; analogously, two parallel independent steps can be sequentialised in any order. This allows for defining concurrent rewrites as equivalence classes up to *shift equivalence* [2], identifying (as for the better-known permutation equivalence of λ-calculus) those rewrite sequences that differ for the scheduling of independent steps.

* Supported by the MIUR project `SisteR` and the University of Padua project `AVIAMO`.

Rewriting over *adhesive categories* (ACs) was proposed as an abstraction of the DPO approach to graph transformation. Many well-known categories with graph-like objects are adhesive. Moreover, since ACs subsume many properties of e.g. HLR categories [5], several results about parallelism and confluence, including the local Church-Rosser theorem, can be proven with no additional assumptions if one restricts to linear rules, i.e., to spans of monos [16].

The restriction to linear rules is common in the DPO literature. It has been folklore, though, that local Church-Rosser should "usually" hold for left-linear rules, i.e. where only the left leg of the span is required to be injective. This has been established within the concrete setting of a category of labelled directed graphs [9, 13] and, in the context of HLR rewriting, sufficient axioms were exhibited in [7]. Those axioms have not been shown to be preserved by common operations such as functor category and slice and thus, to the best of our knowledge, this result has not been proved for a class of categories that would include, say, the category of hyper-graphs **HGraph** or of graphs with second order edges (both presheaf categories, thus adhesive) where linear rules may not suffice.

Whenever distinct parts of the state are to be fused as a result of a transformation, it becomes necessary to use a non-injective morphism as the right-hand side of the rule. A notable example is given by the encodings of process calculi as graph transformation systems, where the exchange of channel names and the creation of connections are modeled as node fusions [10]. In order to extend the results about independence and parallelism to left-linear rules, adhesivity does not appear to be enough. Roughly, while the distinguishing feature of ACs is that pushouts along monos are well behaved, for non-linear rules some of the pushouts involved in the technical lemmas are not necessarily of this kind.

Instead of looking for an *axiomatic* characterisation of the class of categories in which the result could be proven, this paper takes a different approach. First we show that the local Church-Rosser theorem for left-linear rules holds in any AC **C** with a strict initial object (hence extensive [16, Lemma 4.1]), where monos are coproduct injections and pushouts are stable under pullback, an example being **Set**, but not, for example, **Graph**. Then, we note that the technical results used in the proof of the theorem mention only pushouts and pullbacks; hence whenever they hold in **C**, they hold in any functor category over **C** as well as in any slice or coslice category over **C**. Since these operations can be iterated, the result holds in a family of categories that contains most known examples of ACs, such as, in particular, **Graph** and other graph-like categories.

An analogous result can be proven for quasi-adhesivity: a local Church-Rosser theorem for rules where the left-leg is a regular mono holds in any quasi-adhesive category **C** with a strict initial object (hence extensive [16, Lemma 6.3]), where *regular* monos are coproduct injections and pushouts are stable under pullback. Again, this extends to any category obtained from this class by iterating functor, slice and coslice category constructions (although here the presence of equalisers in the base category is needed [16, Lemma 6.6]). A notable example in the base class is **Inj**, the category of injective functions in **Set**. By the closure properties, the category **IGraph** of *graphs with interface* (expressible as $\mathbf{Inj}^{\bullet\rightrightarrows\bullet}$) can be recast, thus subsuming the proposal for their rewriting [11].

The paper has the following structure. We first recall the basics of DPO rewriting in adhesive (and quasi-adhesive) categories and the local Church-Rosser theorem for linear rules. Then, we cut directly to our main contribution, providing a class of categories where the local Church-Rosser theorem holds for left-linear rules. Afterwards we provide an application of our results for the rewriting of graphs with interface, used in the graphical encoding of nominal calculi. We conclude summing up our contribution and laying the plans for future work.

1 Background

In this section we introduce the basics of the double-pushout (DPO) approach to rewriting [2,4], including the notion of sequential and parallel independence. We also introduce adhesive categories [16] as an abstract setting for DPO rewriting.

1.1 DPO Rewriting

Hereafter **C** denotes a fixed category.

Definition 1 (rule and direct derivation). *A (*DPO*)* rule *on* **C** *is a span* $p\colon L \xleftarrow{l} K \xrightarrow{r} R$ *in* **C**. *The objects* L, K, *and* R *are called left-hand side, context and right-hand side of the rule, respectively.*

A match *of* p *in an object* G *of* **C** *is an arrow* $m_L\colon L \to G$. *A* direct derivation $p/m\colon G \Longrightarrow H$ *from* G *to* H *via rule* p *at the match* m_L *is a diagram as depicted in Fig. 1, where the two squares are pushouts in* **C** *and* $m = \langle m_L, m_K, m_R\rangle$.

The notion of sequential independence aims at characterising direct derivations which, even if performed in sequence, do not interfere with each other and thus could be potentially applied in any order (and concurrently).

$$\begin{tikzcd}
L \arrow[d, "m_L"'] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d, "m_K"] & R \arrow[d, "m_R"] \\
G & D \arrow[l, "l^*"] \arrow[r, "r^*"'] & H
\end{tikzcd}$$

Fig. 1. A direct derivation.

Definition 2 (sequential independence). *Let* p_1, p_2 *be rules and let* $p_1/m_1\colon G \Longrightarrow H_1$, $p_2/m_2\colon H_1 \Longrightarrow H$ *be direct derivations as in Fig. 2. They are called* sequential independent *if there exist arrows* $i_1\colon R_1 \to D_2$ *and* $i_2\colon L_2 \to D_1$ *such that* $l_2^* \circ i_1 = m_{R_1}$ *and* $r_1^* \circ i_2 = m_{L_2}$.

A strictly related notion is parallel independence which is aimed at characterising independent direct derivations starting from the same object.

Definition 3 (parallel independence). *Let* p_1, p_2 *be rules and* $p_1/m_1\colon G \Longrightarrow H_1$, $p_2/m_2\colon G \Longrightarrow H_2$ *direct derivations as in Fig. 3. They are called* parallel independent *if there exist arrows* $i_1\colon L_1 \to D_2$ *and* $i_2\colon L_2 \to D_1$ *such that* $l_2^* \circ i_1 = m_{L_1}$ *and* $l_1^* \circ i_2 = m_{L_2}$.

$$\begin{tikzcd}
L_1 \arrow[d, "m_{L_1}"'] & K_1 \arrow[l, "l_1"'] \arrow[r, "r_1"] \arrow[d, "m_{K_1}"'] & R_1 \arrow[dr, "m_{R_1}"'] \arrow[drrr, dotted, "i_1" near start] & & L_2 \arrow[dl, "m_{L_2}"] \arrow[dlll, dotted, "i_2"' near start] & K_2 \arrow[l, "l_2"'] \arrow[r, "r_2"] \arrow[d, "m_{K_2}"'] & R_2 \arrow[d, "m_{R_2}"] \\
G & D_1 \arrow[l, "l_1^*"] \arrow[rr, "r_1^*"'] & & H_1 & & D_2 \arrow[ll, "l_2^*"] \arrow[r, "r_2^*"'] & H
\end{tikzcd}$$

Fig. 2. Sequential independence for $p_1/m_1 \colon G \Longrightarrow H_1$ and $p_2/m_2 \colon H_1 \Longrightarrow H$.

$$\begin{tikzcd}
R_1 \arrow[d, "m_{R_1}"'] & K_1 \arrow[l, "r_1"'] \arrow[r, "l_1"] \arrow[d, "m_{K_1}"'] & L_1 \arrow[dr, "m_{L_1}"'] \arrow[drrr, dotted, "i_1" near start] & & L_2 \arrow[dl, "m_{L_2}"] \arrow[dlll, dotted, "i_2"' near start] & K_2 \arrow[l, "l_2"'] \arrow[r, "r_2"] \arrow[d, "m_{K_2}"'] & R_2 \arrow[d, "m_{R_2}"] \\
H_1 & D_1 \arrow[l, "r_1^*"] \arrow[rr, "l_1^*"'] & & G & & D_2 \arrow[ll, "l_2^*"] \arrow[r, "r_2^*"'] & H_2
\end{tikzcd}$$

Fig. 3. Parallel independence for $p_1/m_1 \colon G \Longrightarrow H_1$ and $p_2/m_2 \colon G \Longrightarrow H_2$.

The two notions are often connected by two properties, which are denoted under the collective name of the local Church-Rosser theorem. The first one is clearly the counterpart of standard local confluence.

Definition 4 (parallel confluence). *Let p_1, p_2 be rules and $p_1/m_1 \colon G \Longrightarrow H_1$, $p_2/m_2 \colon G \Longrightarrow H_2$ parallel independent derivations as in Fig. 3. Then, the derivations satisfy the* parallel local confluence property *if there exist a graph H and derivations $p_2/m_2^* \colon H_1 \Longrightarrow H$, with match $r_1^* \circ i_2$, and $p_1/m_1^* \colon H_2 \Longrightarrow H$, with match $r_2^* \circ i_1$, such that p_1/m_1 and p_2/m_2^* as well as p_2/m_2 and p_1/m_1^* are sequential independent.*

The second part moves instead from sequential independent derivations.

Definition 5 (sequential confluence). *Let p_1, p_2 be rules and $p_1/m_1 \colon G \Longrightarrow H_1$, $p_2/m_2 \colon H_1 \Longrightarrow H$ sequential independent derivations as in Fig. 2. Then, the derivations satisfy the* sequential local confluence property *if there exist a graph H_2 and a derivation $p_2/m_2^* \colon G \Longrightarrow H_2$, with match $l_1^* \circ i_2$, such that p_1/m_1 and p_2/m_2^* are parallel independent.*

Sequential and parallel local confluence are the basis of the concurrency theory of DPO rewriting. When they hold, concurrent derivations can be seen as equivalence classes of concrete derivations up to *shift equivalence* [2], identifying (as for the better-known *permutation equivalence* of λ-calculus) those derivations that differ only for the scheduling of independent steps.

1.2 Rewriting in adhesive categories and local confluence

Most categories of graph-based objects satisfy both local confluence properties when restricted to *linear* rules. A general setting where the theorem can be proven is that of *adhesive categories* (ACs) [16].

$$\begin{tikzcd}
& A \arrow[dl] \arrow[dr] & \\
C \arrow[dr] & & B \arrow[dl] \\
& D &
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
& & A' \arrow[dll] \arrow[dr] \arrow[ddd] & \\
C' \arrow[dr] \arrow[ddd] & & & B' \arrow[dll] \arrow[ddd] \\
& D' \arrow[ddd] & & \\
& & A \arrow[dll] \arrow[dr] & \\
C \arrow[dr] & & & B \arrow[dll] \\
& D & &
\end{tikzcd}$$

(b)

Fig. 4. A pushout square (a) and a commutative cube (b).

Definition 6 (adhesive categories). *A category* **C** *is called* adhesive *if*

- *it has pushouts along monos;*
- *it has pullbacks;*
- *pushouts along monos are* Van Kampen *(*VK*) squares.*

Referring to Fig. 4, a VK *square is a pushout like (a), such that for each commuting cube as in (b) having (a) as bottom face and the back faces of which are pullbacks, the front faces are pullbacks if and only if the top face is a pushout.*

The prototypical AC is **Set**, the category of sets and total functions. Exploiting the closure properties of ACs, it is immediate to deduce that also $\mathbf{Graph} = \mathbf{Set}^{\bullet \rightrightarrows \bullet}$, mentioned before, is adhesive. Likewise, the category of directed hyper-graphs **HGraph** is a category of presheaves and thus adhesive. In fact, $\mathbf{HGraph} = \mathbf{Set}^{\mathbf{M}}$ where **M** is a category with objects $\mathbb{N} \times \mathbb{N} \cup \{\bullet\}$, and where, from any (m, n) there are $m + n$ arrows into $\bullet$.

Example 1. Figure 7 depicts a rule in the AC **HGraph** of hyper-graphs. The left-hand side, context and right-hand side are separated by vertical lines. The arrows from the context to the left- and right-hand side are represented by the positions of the items and by the labels (ignoring for now $\{p\}$ and the dotted arrow). Graphically, nodes are circles and edges are boxes enclosing their label, with either incoming or outgoing tentacles, connecting edges to their source and target nodes, respectively. For the sake of readability, these tentacles are either ordered clock-wise, or when necessary, they are labelled by numbers 0, 1 and 2.

Intuitively, in this graphical interpretation, the application of a rule first removes all the items of G matched by $L - l(K)$, leading to object D. Then the items of $R - r(K)$ are added to D, thus obtaining H. When the rule is not linear, taking the second pushout can also cause some merging.

ACs subsume many properties of HLR categories [6], and this fact ensures the validity of several results about parallelism and confluence. In particular, a local Church-Rosser theorem for *linear* rules holds with no additional assumptions.

Definition 7 ((left-)linear rules in ACs). *A rule* $p\colon L \xleftarrow{l} K \xrightarrow{r} R$ *in an AC is called* left-linear *if* l *is mono, and* linear *if both* l *and* r *are so.*

Actually, the first half of local confluence holds for left-linear rules [16, Theorem 7.7], provided that the AC in question has enough pushouts.

Proposition 1 (parallel confluence in ACs). *Let p_1, p_2 be left-linear rules in an AC with all pushouts and $p_1/m_1\colon G \Longrightarrow H_1$, $p_2/m_2\colon G \Longrightarrow H_2$ parallel independent derivations, as in Fig. 3. Then, they satisfy the parallel local confluence property.*

Instead, the restriction to linear rules is needed for the second half of the local Church-Rosser theorem.

Proposition 2 (sequential confluence in ACs). *Let p_1, p_2 be linear rules in an AC and $p_1/m_1\colon G \Longrightarrow H_1$, $p_2/m_2\colon H_1 \Longrightarrow H$ sequential independent derivations as in Fig. 2. Then, they satisfy the sequential local confluence property.*

1.3 Quasi-adhesivity

A theory for rewriting can be developed also in the wider class of quasi-adhesive categories (QACs). Recall that a mono is *regular* if it is obtained as an equalizer.

Definition 8 (quasi-adhesive categories). *A category* **C** *is called* quasi-adhesive *if*

- *it has pushouts along regular monos;*
- *it has pullbacks;*
- *pushouts along regular monos are* VK *squares.*

Then, the two confluence properties can be established for derivations in QACs, exactly as for those in ACs, by adapting the notion of (left-)linear rule.

Definition 9 ((left-)linear rules in QACs). *A rule $p\colon L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$ in a QAC is called* left-linear *if l is regular mono, and* linear *if both l and r are so.*

Even though every AC is a QAC, no confusion may arise: for ACs Definitions 7 and 9 denote the same class of rules since in ACs all monos are regular [15].

Example 2. A prototypical example of quasi-adhesive category is **Inj**: objects are injective functions in **Set** and arrows are pairs of functions between the corresponding sources and targets, making the diagram commute. An arrow is mono when both components are mono; it is a regular mono when it is mono and the resulting diagram is a pullback [15].

A relevant, graph-based example is the category of *graphs with interface* **IGraph**, whose objects are injective graph morphisms. Such category can be defined as $\mathbf{Inj}^{\bullet \rightrightarrows \bullet}$, hence it is quasi-adhesive by the closure properties in [16], since **Inj** is quasi-adhesive and has equalisers. Similarly, the category **IHGraph** of hyper-graphs with interface can be defined as $\mathbf{Inj}^{\mathbf{M}}$, for **M** the category used in Section 1.2 to present the category of hyper-graphs as a presheaf category.

Graphs with interface have been used in the modelling of process calculi, the basic idea being that nodes in the interface represent the free names of the process itself [10]. They are also at the basis of the borrowed context approach to the labelled semantics of graph transformation [8].

Consider again rule p_π in Fig. 7. Its three components can be seen as graphs with interface. When the interface is discrete (i.e., it contains no edge, as it happens for the examples in this paper), it is simply represented as a set. For instance, the interface of the left-hand side is just $\{p\}$ and the dotted arrow indicates the image of p into the left-hand side graph. Regular monos in **IHGraph** are easily proven to be the pairs of injective graph morphisms such that the interface is reflected as well as preserved, hence rule p_π is left-linear.

2 Church-Rosser for left-linear rules

The results about local confluence in (quasi-)adhesive categories cannot be proven when we consider rules which are only left-linear. Still, the result has been shown to hold in several concrete categories, most notably the category of directed graphs **Graph** [13]. The aim here is to prove the result in an abstract general setting, so that we can conclude that it holds in most of the categories used in DPO rewriting. We will first consider a setting intended to treat the adhesive case, and then generalise it to deal with QACs.

2.1 Adhesive case

We first identify a class of ACs where Proposition 2 can be extended and shown to hold for rules that are merely left-linear.

Definition 10 (class $\mathbb{B}$). *We denote by $\mathbb{B}$ the class of adhesive categories* **C** *which satisfy the following properties*

i) **C** *has all pushouts;*
ii) **C** *has a* strict *initial object* 0 *(any arrow $f\colon a \to 0$ is an isomorphism);*
iii) its monos are coproduct injections *(for any mono $f\colon a \rightarrowtail b$, b is a coproduct of a and some c, and f is the corresponding injection);*
iv) all pushouts are stable under pullback *(for any cube like the one in Fig. 4, if the bottom face is a pushout and the lateral faces are pullbacks then the top face is a pushout).*

The category **Set** of sets and functions is clearly in $\mathbb{B}$. Moreover, recall that any AC satisfying strict initiality is also extensive [16, Lemma 4.1], thus all categories in $\mathbb{B}$ are so. Notice that membership in $\mathbb{B}$ is actually very restrictive; for example $\mathbb{B}$ does not include **Graph**, which has monos that are not coproduct injections, nor the category of sets and partial functions which does not have a strict initial object; both of which *are* examples of ACs. Indeed $\mathbb{B}$ is not in general closed under coslice nor functor category constructions.

In order to prove the local Church-Rosser theorem for left-linear rules we need just two technical lemmas. The first result concerns the validity of a decomposition property which generalises the one holding in ACs [16, Lemma 4.6].

$$\begin{tikzcd}
A \arrow[r, "k"] \arrow[d, "l"'] \arrow[dr, phantom, "(1)"] & B \arrow[r, "r"] \arrow[d, "s"] \arrow[dr, phantom, "(2)"] & C \arrow[d, "v"] \\
X \arrow[r, "u"'] & Y \arrow[r, tail, "w"'] & Z
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
A \arrow[r, "k"] \arrow[d, tail, "l"'] \arrow[dr, phantom, "(1)"] & B \arrow[r, "r"] \arrow[d, tail, "s"] \arrow[dr, phantom, "(2)"] & C \arrow[d, tail, "v"] \\
X \arrow[r, "u"'] & Y \arrow[r, "w"'] & Z
\end{tikzcd}$$

(b)

Fig. 5. Diagrams for (a) mixed decomposition and (b) pushout decomposition.

Definition 11 (mixed decomposition). *We say that a category $\boldsymbol{C}$ satisfies the* mixed decomposition *property if for any commuting diagram like the one depicted in Fig. 5(a) (where w is mono), whenever (1)+(2) is a pushout and (2) is a pullback, then (1) is a pushout.*

Lemma 1 (mixed decomposition in $\mathbb{B}$). *Let $\boldsymbol{C}$ be a category in the class $\mathbb{B}$ of Definition 10. Then, it satisfies the mixed decomposition property.*

With respect to [16, Lemma 4.6], we dropped the requirement enforcing arrows l, s, v and r to be monos. A similar result is proven in [14] for partial ACs, but with the additional requirement that the outer pushout is hereditary. As for the stricter case, the lemma above implies that all the squares of the diagram in Fig. 5(a) are both pushouts and pullbacks.

Definition 12 (pushout decomposition in $\mathbb{B}$). *We say that a category $\boldsymbol{C}$ satisfies the* pushout decomposition *property if for any commuting diagram like the one depicted in Fig. 5(b) (where l, s and v are mono), whenever the regions (1)+(2) and (2) are pushouts then (1) is a pushout.*

Lemma 2 (pushout decomposition in $\mathbb{B}$). *Let* $\mathbf{C}$ *be a category in class $\mathbb{B}$. Then, it satisfies the pushout decomposition property.*

The crucial observation is that Lemmas 1 and 2 mention only monos, pushouts and pullbacks, and since all these are built “pointwise” for product, functor category, slice and coslice, the lemmas hold in any category obtained from a category in class $\mathbb{B}$ by iterating these operations.

Proposition 3. *If mixed and pushout decompositions hold in a category* $\mathbf{C}$*, then they hold in* $\mathbf{C}/C$*,* $C/\mathbf{C}$ *and* $\mathbf{C}^{\mathbf{X}}$ *for any* $C \in \mathbf{C}$ *and any small category* $\mathbf{X}$.

Finally, since most known ACs are constructed from **Set** by iterating the above operations, the local Church-Rosser theorem for left-linear rules holds in all these categories. Examples include the aforementioned **Graph**, **HGraph** and their typed versions (slices) as well as the category of sets and partial functions.

Proposition 4 (sequential confluence for left-linear rules). *Let* $\mathbf{C}$ *be any category obtained from a category in class $\mathbb{B}$ by iterated application of the functor category, slice and coslice constructions; let p_1, p_2 be left-linear rules over* $\mathbf{C}$*, and let $p_1/m_1 \colon G \Longrightarrow H_1$, $p_2/m_2 \colon H_1 \Longrightarrow H$ be sequential independent derivations as in Fig. 2. Then they satisfy the sequential local confluence property.*

2.2 Quasi-adhesive case

The theory above can be easily generalised to quasi-adhesivity, roughly by replacing ACs with QACs and monos with regular monos.

Definition 13. *Let $\mathbb{QB}$ be the class of quasi-adhesive categories* $\mathbf{C}$ *such that*

i) $\mathbf{C}$ *has all pushouts;*
ii) $\mathbf{C}$ *has a strict initial object;*
iii) regular monos are coproduct injections;
iv) all pushouts are stable under pullback.

Then Proposition 4 holds replacing class $\mathbb{B}$ with $\mathbb{QB}$, because Lemmas 1 and 2 can be reproved in a straightforward fashion.

It is not difficult to show that **Inj** is in $\mathbb{QB}$ using the facts that it is a quasi-topos and quasi-adhesive [15]; the fact that regular monos are coproduct injections is an easy exercise. We thus obtain the local Church-Rosser theorem for it and any functor category over it, hence in particular **IHGraph**, the category of hyper-graphs with interface in which we work in the following section.

3 Graph Rewriting for the Concurrent Semantics of π

A recently developed area of application for graphs with interface is the visual modelling of nominal calculi. Here we focus on the deterministic fragment of the π-calculus and on its graphical semantics, along the lines of [11] (see Section 2 there for the syntax and operational semantics of the calculus). The results in this paper are needed for formally describing the concurrency in the graphical semantics, thus obtaining one of the few such semantics available for π-calculus.

The idea is quite simple: we work in the (quasi-adhesive) category of graphs with interface **IHGraph** and each process is associated with a graph having a discrete interface. The topology of the graph represents a (simplified) syntactic tree of the process. The interface contains a node p, denoting the root of the graph, and a set of nodes denoting free names in the process (this can be larger than the actual set of free names of the process). As an example, the encoding of the process $(\nu a)(\bar{b}a.\bar{a}a \mid b(d).\bar{d}c)$ can be found in Fig. 6. The interface contains the root p and two other nodes representing the free names b and c (the different sorts for processes and names are visually represented by $\bullet$ and $\circ$ nodes, respectively). Each input or output prefix operator, like $\bar{b}a$, corresponds to an edge labelled by *in* or *out*, with one incoming tentacle and three (ordered) outgoing ones, denoting the continuation (labelled by 0) and the channel names, respectively. For the sake of readability, these outgoing tentacles are either ordered clock-wise, or whenever necessary, they are labelled by numbers 0, 1 and 2.

The restriction operator (νa) is modelled by an edge ν connecting the root to the restricted name a and by dropping the node a from the interface. Note the lack of an edge for the parallel operator: parallelism is reduced to being linked to the same node, as with the components $\bar{b}a.\bar{a}a$ and $b(d).\bar{d}c$ of our process.

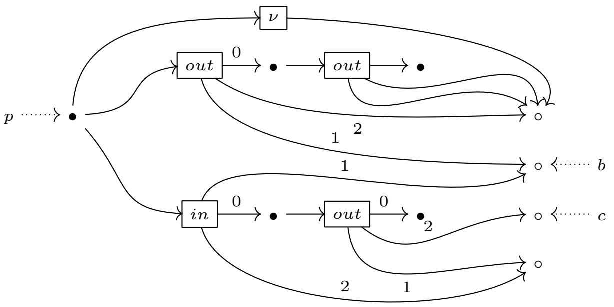

Fig. 6. The graphical encoding of process $(\nu a)(\overline{b}a.\overline{a}a \mid b(d).\overline{d}c)$.

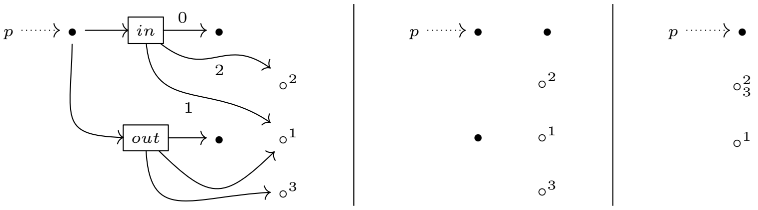

Fig. 7. The rule p_π for synchronization.

A single rule p_π (depicted in Fig. 7) suffices for simulating process reduction. As explained before, a node may be the target of a dotted arrow, meaning that the node is in the image of a node of the interface (the source of the arrow). The nodes may be labeled by natural numbers, which are used for describing the (interface preserving) span of arrows constituting the rule. E.g. the nodes identified by 2 and 3 are merged by the rule.

The structural rules are taken care of by the matching mechanism: the embedding of a graph into a larger one models the closure of reduction with respect to contexts. The presence of the interface $\{p\}$ guarantees that a reduction may occur only on the top operators, i.e., never inside a prefix such as $\overline{b}a$ or $b(d)$. Graph isomorphism takes care of the closure with respect to structural congruence. For example, the graph on the left of Fig. 8 is the encoding of the target process of the reduction $(\nu a)(\overline{b}a.\overline{a}a \mid b(d).\overline{d}c) \to (\nu a)(\overline{a}a \mid \overline{a}c)$.

The presence of the interface node p in the context graph of p_π implies that p can be shared by concurrent reductions. This permits the simultaneous execution of reductions involving top operators.

However, rule p_π is left-linear only: the left leg of the rule is a regular mono (since it reflects the interface), while the right leg is not even mono. Hence, even though category **IHGraph** is quasi-adhesive, the results previously available for rewriting in QACs do not apply, and we need to resort to the theory in Section 2 to formally analyse the concurrency in the system.

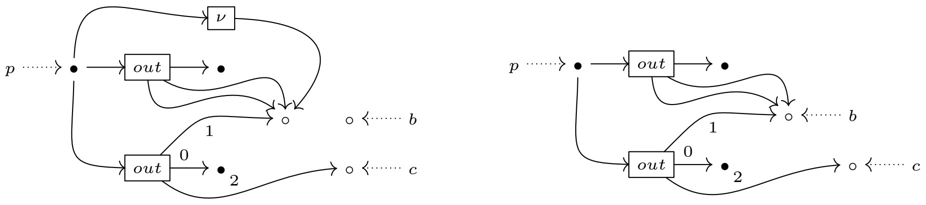

Fig. 8. The graphical encoding of process $(\nu a)(\overline{a}a \mid \overline{a}c)$ (left) and $\overline{b}b \mid \overline{b}c$ (right).

In order to make the example more illustrative, we add rule p_ν in Fig. 9. It models the revelation of a restricted name $(\nu a)P \to P\{^b/_a\}$, which is associated to the free name b occurring in the process, while the corresponding restriction operator is removed. As before, the rule is only left-linear.

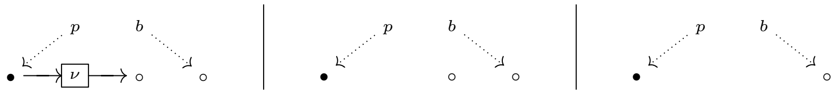

Fig. 9. The rule p_ν for revealing a restricted name.

Rule p_ν can be applied to $(\nu a)(\overline{a}a \mid \overline{a}c)$, resulting in the process $\overline{b}b \mid \overline{b}c$: its graphical encoding is depicted on the right of Fig. 8. It is intuitively clear that the two direct derivations represented by first applying the synchronization, and then the revelation, are sequential independent, so they should be executable simultaneously. In order to prove this formally, since the involved rules are left-linear only, we need to resort to the quasi-adhesive variant of Proposition 4.

4 Conclusions

We have identified a class of categories where the local Church-Rosser theorem, a fundamental result in the DPO approach to rewriting, holds also for left-linear rules and arbitrary matches. This class includes most of the adhesive and quasi-adhesive categories actually used as domain categories for rewriting.

There are many examples where left-linear rules arise naturally. One that we consider relevant is related to the graphical encodings of nominal calculi: changes to the physical or logical topology of a system determined by phenomena like name passing or fusion or code mobility are naturally modelled by rules whose right-hand sides are not monomorphisms. By our results these calculi can be equipped with a concurrent semantics, as obtained by exploiting the local Church-Rosser theorem (see e.g. [10,12]).

A further advancement in the theory would be to consider situations where the left-linearity of the rules does not guarantee the existence of the pushout complement, as in the case of the (quasi-adhesive) category of term graphs [3] and of the category of graphs with equivalences [1], possibly constraining the match without necessarily requiring it to be a monomorphism.

References

1. Baldan, P., Gadducci, F., Montanari, U.: Concurrent rewriting for graphs with equivalences. In: CONCUR'06. LNCS, vol. 4137, pp. 279–294. Springer (2006)
2. Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R., Löwe, M.: Algebraic approaches to graph transformation I: Basic concepts and double pushout approach. In: Rozenberg, G. (ed.) Handbook of Graph Grammars and Computing by Graph Transformation, vol. 1, pp. 163–245. World Scientific (1997)
3. Corradini, A., Gadducci, F.: On term graphs as an adhesive category. In: TERMGRAPH'04. ENTCS, vol. 127(5), pp. 43–56. Elsevier (2005)
4. Drewes, F., Habel, A., Kreowski, H.J.: Hyperedge replacement graph grammars. In: Rozenberg, G. (ed.) Handbook of Graph Grammars and Computing by Graph Transformation, vol. 1, pp. 95–162. World Scientific (1997)
5. Ehrig, H., Ehrig, K., Prange, U., Täntzer, G.: Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science, Springer (2006)
6. Ehrig, H., Habel, A., Padberg, J., Prange, U.: Adhesive high-level replacement categories and systems. In: ICGT'04. LNCS, vol. 3256, pp. 144–160. Springer (2004)
7. Ehrig, H., Habel, A., Parisi-Presicce, F.: Basic results for two types of high-level replacement systems. In: GETGRATS Closing Workshop. ENTCS, vol. 51, pp. 127–138. Elsevier (2002)
8. Ehrig, H., König, B.: Deriving bisimulation congruences in the DPO approach to graph rewriting with borrowed contexts. Mathematical Structures in Computer Science 16(6), 1133–1163 (2006)
9. Ehrig, H., Kreowski, H.J.: Parallelism of manipulations in multidimensional information structures. In: MFCS'76. LNCS, vol. 45, pp. 284–293. Springer (1976)
10. Gadducci, F.: Graph rewriting for the π-calculus. Mathematical Structures in Computer Science 17(3), 407–437 (2007)
11. Gadducci, F., Lluch-Lafuente, A.: Graphical encoding of a spatial logic for the π-calculus. In: CALCO'07. LNCS, vol. 4624, pp. 209–225. Springer (2007)
12. Gadducci, F., Monreale, G.V.: A decentralized implementation of mobile ambients. In: ICGT'08. LNCS, vol. 5214, pp. 115–130. Springer (2008)
13. Habel, A., Müller, J., Plump, D.: Double-pushout graph transformation revisited. Mathematical Structures in Computer Science 11(5), 637–688 (2001)
14. Heindel, T.: Hereditary pushouts reconsidered. In: ICGT'10. LNCS, vol. 6372, pp. 250–265. Springer (2010)
15. Johnstone, P.T., Lack, S., Sobociński, P.: Quasitoposes, quasiadhesive categories and Artin glueing. In: CALCO'07. LNCS, vol. 4624, pp. 312–326. Springer (2007)
16. Lack, S., Sobociński, P.: Adhesive and quasiadhesive categories. Theoretical Informatics and Applications 39(3), 511–545 (2005)