Algebraic graph rewriting with controlled embedding

Andrea Corradini, Dominique Duval, Rachid Echahed, Frederic Prost, Leila Ribeiro

Algebraic Graph Rewriting with Controlled Embedding*

Andrea Corradini[a], Dominique Duval[b], Rachid Echahed[c], Frédéric Prost[c,*], Leila Ribeiro[d]

[a] *Dipartimento di Informatica, Pisa (Italy)*
[b] *Laboratoire Jean Kuntzmann, Univ. Grenoble Alpes, CNRS, Grenoble (France)*
[c] *Laboratoire d'Informatique de Grenoble, Univ. Grenoble Alpes, CNRS, Grenoble (France)*
[d] *Institute of Informatics, Univ. Federal do Rio Grande do Sul, Porto Alegre (Brazil)*

Abstract

Graph transformation is a specification technique suitable for a wide range of applications, specially the ones that require a sophisticated notion of state. In graph transformation, states are represented by graphs and actions are specified by rules. Most algebraic approaches to graph transformation proposed in the literature ensure that if an item is preserved by a rule, so are its connections with the graph where it is embedded. But there are applications in which it is desirable to specify different embeddings. For example when cloning an item, there may be a need to handle the original and the copy in different ways. We propose a new algebraic approach to graph transformation, AGREE: Algebraic Graph Rewriting with controllEd Embedding, where rules allow one to specify how the embedding should be carried out. We define this approach in the framework of classified categories which are categories endowed with partial map classifiers. This new approach leads to graph transformations in which effects may be non-local, e.g. a rewrite step may alter a node of the host graph which is outside the image of the left-hand side of the considered rule. We propose a syntactic condition on AGREE rules which guarantees the locality of transformations. We also compare AGREE with other algebraic approaches to graph transformation.

Keywords: Rewrite Systems, Graph Transformation, Algebraic Methods

*This work has been partly funded by projects TGV (CNRS-INRIA-FAPERGS/(156779 and 12/0997-7)), VeriTeS (CNPq 485048/2012-4 and 309981/2014-0).
*Corresponding author
Email addresses: andrea@di.unipi.it (Andrea Corradini), Dominique.Duval@univ-grenoble-alpes.fr (Dominique Duval), Rachid.Echahed@univ-grenoble-alpes.fr (Rachid Echahed), Frederic.Prost@univ-grenoble-alpes.fr (Frédéric Prost), leila@inf.ufrgs.br (Leila Ribeiro)

1. Introduction

Graphs are used to describe a wide range of situations in a precise yet intuitive way. Different kinds of graphs are involved in modelling techniques depending on the investigated fields, which include computer science [2], chemistry [34], biology [14], quantum computing [6], etc. When system states are represented by graphs, it is natural to use rewrite rules that transform graphs to describe the system evolution. There are two main streams in the research on graph transformations: (i) the algorithmic approaches which describe explicitly, with concrete algorithms, the result of applying a rule to a graph [26, 20], and (ii) the algebraic approaches [25, 30] which define graph transformation steps abstractly using basic constructs borrowed from category theory. We consider the latter in this work and are interested in a family of algebraic graph transformation systems where rewrite rules are defined as spans of the form $L \leftarrow K \rightarrow R$. In such systems, a rewrite step, as depicted below, transforms a graph G into a graph H, using a match (graph homomorphism) $m : L \rightarrow G$, in two steps. The first one, depicted by square X, consists in building the graph D, by "replacing" L by K in G. This may be accomplished in different ways that will be discussed below. Then a second step is performed, building the graph H by "replacing" K by R in D. This is typically done by means of a pushout construction (square PO).

$$\begin{array}{ccccc} L & \xleftarrow{\ l\ } & K & \xrightarrow{\ r\ } & R \\ {\scriptstyle m}\downarrow & X & \downarrow & PO & \downarrow \\ G & \xleftarrow{\qquad} & D & \xrightarrow{\qquad} & H \end{array}$$

When X is a pushout, D is constructed as a pushout complement (POC), and the approach is known as *double-pushout* (DPO) [25]. This approach has been widely investigated in the literature [13, 21]. It is well suited when the required transformations add, delete, preserve, or merge graph items. However, the DPO approach fails to specify cloning of nodes. Indeed, to clone a node one needs to use a non injective morphism $l : K \rightarrow L$. In such a situation the POC is not always unique and thus transformations are not deterministic. To overcome this situation, the object D can be defined as the final pullback complement ($FPBC$), as it is done in the *sesqui-pushout* approach (SqPO) [12]. In this case the square X is a pullback. Cloning a node in this approach copies *all* its incident edges. There is no flexibility for determining the incident edges to be copied.

In order to overcome the restrictions imposed by the construction of $FPBC$, we propose, in this paper, a new algebraic approach called *Algebraic Graph Rewriting with controllEd Embedding* (AGREE), where the square X is a pullback that can be constructed from an additional morphism $t : K \rightarrow T_K$. Thus an AGREE rule is a span of the form $L \leftarrow K \rightarrow R$, enriched with a third morphism $t : K \rightarrow T_K$ called the *embedding*. Intuitively, the embedding morphism specifies, in the case of node cloning, how a copy of a node is connected to the rest of the graph. A graph G to be transformed can be described as the disjoint

union of the *image* of L (still denoted L here) and its *context*, which itself is made of the *strict complement* $G \setminus L$ of L in G, i.e. the largest subgraph of G disjoint from L, and the *adjacent edges* $\widetilde{L}$, i.e. the edges of G not in L but with at least one incident node in L. Similarly, H is the disjoint union of R, $H \setminus R$ and $\widetilde{R}$. In the SqPO approach, $H \setminus R$ is isomorphic to $G \setminus L$ and $\widetilde{R}$ is built in a systematic way from $\widetilde{L}$ in order to fit with the replacement of L by R.

A first attempt to offer a flexible way to deal with the adjacent edges was proposed in the *rewriting with polarized cloning* approach (PSqPO) [16], by distinguishing incoming and outgoing adjacent edges. The AGREE approach goes beyond PSqPO and provides a powerful tool for controlling the modification of the adjacent edges. In fact, the AGREE transformation system allows one to modify the whole context: both the strict complement and the adjacent edges. When the embedding t satisfies some additional property, AGREE becomes *local*, i.e. it modifies the adjacent edges in a flexible way without modifying the strict complement. When t satisfies a still more restrictive property, then AGREE coincides with SqPO. In this case a final pullback complement results from a simple pullback construction, and it follows that AGREE subsumes SqPO, However, it is fair to stress that AGREE transformations are defined only for *monic* matches on categories with a *partial map classifier*, as defined later, while SqPO transformations are defined also for non-monic matches and in categories satisfying much less restrictive requirements.

Let us consider a basic rule ρ which clones one node. Rule ρ can be drawn informally as in Figure 1, where the white circle represents a clone of the black circle.

Figure 1: Rule ρ specifying the cloning of a node.

What should be the result of applying ρ to a given graph? Different approaches may provide different answers to this question. In Figure 2, we show three possible results of cloning a specific node, represented by a black circle, in a given graph G. All graphs H_1, H_2 and H_3 can be obtained as AGREE transformations of graph G using suitable variants of rule ρ. Instead, only graphs H_1 and H_2 can be obtained as PSqPO steps while H_1 is the unique graph that can be obtained via SqPO.

Figure 2: Three possible rewriting steps using rule ρ.

The paper is organized as follows. The AGREE approach is presented in Section 2. As usual for the algebraic approaches, AGREE rewriting is defined in a more general setting of categories satisfying suitable requirements, mainly the existence of a partial map classifier. While AGREE rewriting does not satisfy the *locality* property in general, we propose in Section 3 sufficient conditions

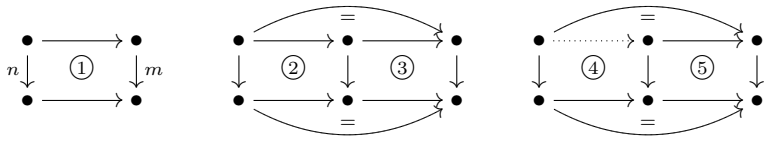
which ensure local transformation of AGREE rewrite steps. Then, in Section 4, the AGREE approach is compared to related work with some emphasis on sesqui-pushout rewriting (restricted to monic matches), rewriting with polarised cloning as well as rewriting in span-categories [31]. Concluding remarks are given in Section 5. This paper is mainly based on the conference paper [7].

2. Algebraic Graph Rewriting with Controlled Embedding

In this section, we introduce the AGREE algebraic rewriting approach by defining rules, matches and rewrite steps. One main difference with respect to the DPO and SqPO approaches is that a rule has an additional component $t : K \rightarrowtail T_K$, called the *embedding*, that enriches the interface K and can be used to control the embedding of items in K to items outside the image of the match. Another important difference is that a rewrite step requires only a pullback and a pushout construction, in contrast with the "complement" constructions like the *pushout complement* of DPO [25] and the *final pullback complement* used in SqPO [12] and in the *gluing construction* of [31]. We start recalling some definitions and properties concerning pullbacks and partial map classifiers in Section 2.1, then several classified categories of graphs are presented in Section 2.2, and finally the AGREE approach is defined in Section 2.3. We refer to [1] for basic categorical notions like pullback, pushout etc., and to [5, 28] for the notion of partial map classifier.

2.1. Some properties of pullbacks and partial map classifiers

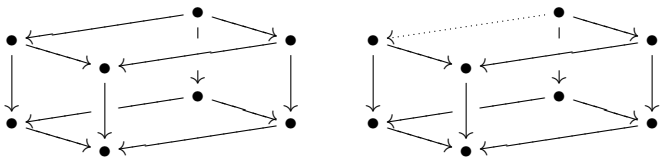
We recall some properties of pullbacks in any category (illustrated in Diagram (1)), that are used repeatedly in the paper. Pullbacks *preserve monomorphisms*: if square ① is a pullback and m is a monomorphism, then n is a monomorphism as well. Pullbacks satisfy the following *composition* and *decomposition* properties: if diagram ②+③ is commutative with square ③ a pullback, then ② is a pullback if and only if the outer square ②+③ is a pullback. Pullbacks satisfy the following dynamic version of the *decomposition* property: if diagram ④+⑤ is commutative with square ⑤ and the outer square ④+⑤ pullbacks, then there is a unique arrow (the dotted one) such that the top triangle and square ④ commute and, in addition, ④ is a pullback.



(1)

These properties imply Lemma 1, which states, that commutative squares and pullbacks can be pulled back in any category. To avoid ambiguities: in cubes as in Diagram (2), the *left*, *right*, *front* and *back* faces refer respectively to the front-left, back-right, front-right and back-left faces.

Lemma 1 (cube lemma for pullbacks). *Given the cube at the left of Diagram (2), if the left, bottom, right and top faces are pullbacks then: if the front face is commutative so is the back face, and if the front face is a pullback so is the back face. Given the part in solid arrows of the cube at the right of Diagram (2), if the left, bottom and right faces are pullbacks and the front face commutes, then there is a unique arrow (the dotted arrow) such that the top face is a pullback.*



(2)

Definition 2 (reflection).
Given objects X, Y, Z and arrows $f : X \to Z$, $g : Y \to Z$, we say that f *is reflected along* g if there is an arrow $h : X \to Y$ such that (h, id_X) is pullback of (g, f). Then h is uniquely determined by f and g and we say that f *is reflected along* g *by* h, or that h *reflects* f *along* g.

$$\begin{array}{ccc} X & \xrightarrow{id_X} & X \\ {\scriptstyle h}\downarrow & PB & \downarrow{\scriptstyle f} \\ Y & \xrightarrow{g} & Z \end{array}$$

Note that if f is monic, then h is a monomorphism. The properties in the following lemma are obvious consequences of the definition of pullback.

Lemma 3 (some properties of reflection). *Given objects X, Y, Z and arrows $f : X \to Z$, $g : Y \to Z$ and $h : X \to Y$, the followings hold:*

1. *f is reflected along g by h if and only if for all objects W and arrows $a : W \to X$, $b : W \to Y$, if $f \circ a = g \circ b$ then $h \circ a = b$;*
2. *if g is a monomorphism and $g \circ h = f$, then h reflects f along g.*

As mentioned in the Introduction, the AGREE approach requires the notion of partial map classifier that we recall here. Our terminology is based on [28], it may differ slightly from other papers, for instance from [5]. In any category $\mathbf{C}$, a class $\mathcal{M}$ of monomorphism is *stable (under pullback)* if for each pair of morphisms $(m : Z \rightarrowtail X, f : Y \to X)$ with m in $\mathcal{M}$ and for each pullback $(f' : W \to Z, m' : W \rightarrowtail Y)$ of m and f, the monomorphism m' is in $\mathcal{M}$. A class $\mathcal{M}$ of monomorphisms is *admissible* if it contains all identities, it is closed under composition, each pair of morphisms $(m : Z \rightarrowtail X, f : Y \to X)$ with m in $\mathcal{M}$ has a pullback, and $\mathcal{M}$ is stable under pullback.

In a category $\mathbf{C}$ with an admissible class of monomorphisms $\mathcal{M}$, an *$\mathcal{M}$-partial map* $(m, f) : Z \rightharpoonup Y$ is a span made of a monomorphism $m : X \rightarrowtail Z$ in $\mathcal{M}$ and an arrow $f : X \to Y$ in $\mathbf{C}$, up to the equivalence relation $(m', f') \sim (m, f)$ whenever there is an isomorphism h with $m' \circ h = m$ and $f' \circ h = f$. A category $\mathbf{C}$ with an admissible class of monomorphisms $\mathcal{M}$ has a *$\mathcal{M}$-partial map classifier* (T, η) if T is a functor $T : \mathbf{C} \to \mathbf{C}$ and η is a natural transformation $\eta : Id_{\mathbf{C}} \dot{\to} T$, such that for each object Y of $\mathbf{C}$ the arrow η_Y is in $\mathcal{M}$ and for each $\mathcal{M}$-partial

map $(m, f) : Z \rightharpoonup Y$ there is a unique arrow $\varphi(m, f) : Z \to T(Y)$ such that square (3) ① is a pullback. Then the monomorphism $\eta_Y : Y \rightarrowtail T(Y)$ is *the $\mathcal{M}$-partial map classifier of Y*. In addition, for each monomorphism $m : X \rightarrowtail Z$ in $\mathcal{M}$ we use the notation $\overline{m} = \varphi(m, id_X)$, so that $\overline{m}$ is uniquely defined by pullback (3) ②. Thus η_X is reflected along $\overline{m}$ by m.

$$\begin{array}{ccc} X & \xrightarrow{\quad f \quad} & Y \\ {\scriptstyle m}\downarrow & PB\,① & \downarrow{\scriptstyle \eta_Y} \\ Z & \xrightarrow{\quad \varphi(m,f) \quad} & T(Y) \end{array} \qquad \begin{array}{ccc} X & \xrightarrow{\quad id_X \quad} & X \\ {\scriptstyle m}\downarrow & PB\,② & \downarrow{\scriptstyle \eta_X} \\ Z & \xrightarrow{\quad \overline{m}=\varphi(m,id_X) \quad} & T(X) \end{array} \tag{3}$$

Informally, if $(m, f) : Z \rightharpoonup Y$ is a partial map, the total arrow $\varphi(m, f) : Z \to T(Y)$ should agree with (m, f) on the items of Z on which (m, f) is defined, and should map any item of Z on which (m, f) is not defined in a unique possible way to some item of $T(Y)$ which does not belong to (the image via η_Y of) Y. And $\overline{m}$ can be seen as a kind of inverse for the monomorphism m.

Remark 4. Every elementary topos has an $\mathcal{M}$-partial map classifier, with $\mathcal{M}$ the class of all monomorphisms [5]. The category **Set** is an elementary topos, as well as each *presheaf category* $\mathbf{Set}^{\mathbf{C}}$ where $\mathbf{C}$ is a small category and each slice category $\mathbf{C}{\downarrow}X$ where $\mathbf{C}$ is an elementary topos.

In the rest of the paper, when the admissible class of monomorphisms $\mathcal{M}$ consists of *all* the monomorphisms of $\mathbf{C}$ we will omit explicit references to it, calling $\mathcal{M}$-partial maps simply *partial maps* and $\mathcal{M}$-partial map classifier simply *partial map classifier*.

For example, in **Set** the partial map classifier is made of the functor T such that $T(X) = X + \{*\}$ and $T(f) = f + id_{\{*\}}$, and the natural transformation η is made of the inclusions $\eta_X : X \to X + \{*\}$. For each partial function $(m, f) : Z \rightharpoonup Y$, the function $\varphi(m, f) : Z \to Y + \{*\}$ extends f by mapping x to $f(x')$ when $x = m(x')$ and x to $*$ when x is not in the image of m. For each monomorphism $m : X \rightarrowtail Z$ the function $\overline{m} : Z \to X + \{*\}$ is the inverse of m on $m(X)$ and it maps every element of Z outside $m(X)$ to $*$.

In Section 2.2 we will describe some categories of graphs with a partial map classifier.

Definition 5 (classified category). A *classified category* $(\mathbf{C}, \mathcal{M}, T, \eta)$ is made of a category $\mathbf{C}$ with all pullbacks, an admissible class of monomorphisms $\mathcal{M}$ on $\mathbf{C}$ and an $\mathcal{M}$-partial map classifier (T, η).

Notice that *all* pullbacks are required, not only pullbacks over monomorphisms in $\mathcal{M}$. This condition will be exploited in the definition of AGREE rewriting (Definition 11).

Proposition 6 (some properties of classified categories). *In a classified category $(\mathbf{C}, \mathcal{M}, T, \eta)$:*

1. *T preserves pullbacks.*

2. *The natural transformation η is* cartesian*, which means that for each arrow $f : X \to Y$ in $\mathbf{C}$ the naturality square (4) is a pullback, or equivalently, that $\varphi(\eta_X, f) = T(f)$. It follows that $\overline{\eta_X} = id_{T(X)}$.*

$$\begin{array}{ccc} X & \xrightarrow{\quad f \quad} & Y \\ \downarrow{\scriptstyle \eta_X} & PB & \downarrow{\scriptstyle \eta_Y} \\ T(X) & \xrightarrow{\quad T(f) \quad} & T(Y) \end{array} \tag{4}$$

3. *Let $(m, f) : X \rightharpoonup Y$ be an $\mathcal{M}$-partial map, then*
$$\varphi(m, f) = \varphi(\eta_X, f) \circ \varphi(m, id_X) = T(f) \circ \overline{m}.$$

Proof.

1. This is [5, Prop.2.2.vi.]. In fact T is a right adjoint and thus it preserves all limits.
2. This is [5, Prop.2.2.ii.].
3. Apply the composition property of pullbacks to pullback (3) ② followed by pullback (4). □

2.2. Examples of classified categories

We introduce here some classified categories: the categories of *graphs* $\mathbf{Gr}$, of *typed graphs* $\mathbf{Gr}{\downarrow}G_0$, and of *polarized graphs* $\mathbf{Gr}^{\pm}$. Other classified (presheaf) categories will be considered in the examples of Section 3.3.

Definition 7 (graphs). The category of *graphs* $\mathbf{Gr}$ is defined as the presheaf category $\mathbf{Set}^{\mathbf{C_{Gr}}}$ where $\mathbf{C_{Gr}}$ is the category having two objects E, N and two non-identity arrows $s, t : E \to N$. Spelling out the definition, an object of $\mathbf{Gr}$ is a *graph* $X = (N_X, E_X, s_X, t_X)$, made of a set of *nodes* N_X, a set of *edges* E_X and two functions $s_X, t_X : E_X \to N_X$, called *source* and *target*, respectively. As usual, we write $n \xrightarrow{e} p$ when $e \in E_X$, $n = s_X(e)$ and $p = t_X(e)$. An arrow in $\mathbf{Gr}$ from graph X to graph Y is a *(graph) (homo)morphism* $f : X \to Y$, made of two functions $f_N : N_X \to N_Y$ and $f_E : E_X \to E_Y$, such that $f_N(n) \xrightarrow{f_E(e)} f_N(p)$ in Y for each edge $n \xrightarrow{e} p$ in X. A monomorphism $m : X \rightarrowtail Y$ is a morphism such that both components are injective.

By Remark 4 category $\mathbf{Gr}$ has a partial map classifier (T, η). It is defined as follows for each graph G:

- The graph $T(G)$ contains as nodes all nodes of G and one additional node $*$. Also, $T(G)$ contains all the edges of G plus one edge $*_{(n,p)} : n \to p$ for each pair of nodes (n, p) in $(N_G + \{*\}) \times (N_G + \{*\})$.
- $\eta_G : G \to T(G)$ embeds G into $T(G)$.

If $f : X \to G$ is a morphism and $m : X \rightarrowtail H$ is a monomorphism, defining a partial graph morphism $(m, f) : H \rightharpoonup G$, the total morphism $\varphi(m, f) : H \to T(G)$ is defined as follows:

- For each item $x \in X_N + X_E$, $\varphi(m,f)(m(x)) = f(x)$: this is well defined because m is a monomorphism and defines $\varphi(m,f)$ on the image of m.
- For each node $n \in H_N \setminus m(X_N)$, $\varphi(m,f)_N(n) = *$.
- For each edge $n \xrightarrow{e} p \in H_E \setminus m(X_E)$, $\varphi(m,f)_E(e) = *_{(\varphi(m,f)_N(n), \varphi(m,f)_N(p))}$.

The node $*$ and the edges $*_{(n,p)}$ will be called the $*$-*items* of $T(G)$, they form the context of G in $T(G)$.

Definition 8 (typed graphs). Given a fixed graph G_0, called the *type graph*, the category of *graphs typed over* G_0 is the slice category $\mathbf{Gr} \downarrow G_0$. An object of this category is a pair (G, t_G) consisting of a graph G and a morphism $t_G : G \to G_0$. An arrow $f : (G, t_G) \to (G', t_{G'})$ is a graph morphism $f : G \to G'$ such that $t_{G'} \circ f = t_G$.

Again by Remark 4, category $\mathbf{Gr} \downarrow G_0$ has a partial map classifier (T, η). For each graph (G, t_G) typed over G_0, the typed graph $T(G, t_G)$ contains G, as well as a node $*_n$ for each node $n \in N_{G_0}$. Additionally, for each edge $n \xrightarrow{e} p$ of G_0, it also includes one edge $*_{(e,x,y)} : x \to y$ for each pair of nodes (x, y) in $(t_G^{-1}(n) + \{*_n\}) \times (t_G^{-1}(p) + \{*_p\})$. The typing morphism $t : T(G, t_G) \to G_0$ is defined in the expected way: $t(x) = t_G(x)$ for all $x \in E_G + N_G$, $t_N(*_n) = n$ for all $n \in N_G$; and $t_E(*_{(e,x,y)}) = e$ for the remaining edges. The typed graph monomorphism $\eta_G : G \to T(G, t_G)$ simply embeds G into $T(G)$.

Given a partial typed graph morphism $(m, f) : (H, t_H) \rightharpoonup (G, t_G)$, where $f : (X, t_X) \to (G, t_G)$ and $m : (X, t_X) \rightarrowtail (H, t_H)$, the total morphism $\varphi(m, f) : (H, t_H) \to T(G, t_G)$ is defined as follows:

- For each item $x \in X_N + X_E$, $\varphi(m,f)(m(x)) = f(x)$.
- For each node $n \in H_N \setminus m(X_N)$, $\varphi(m,f)_N(n) = *_{t_H(n)}$.
- For each edge $n \xrightarrow{e} p \in H_E \setminus m(X_E)$, $\varphi(m,f)_E(e) = *_{(e,x,y)}$, where $x = \varphi(m,f)_N(n)$ and $y = \varphi(m,f)_N(p)$.

We refer to [7, Example 2] for a concrete example of this construction.

Definition 9 (polarized graphs [17]). A *polarized graph* $\mathbb{X} = (X, N_X^+, N_X^-)$ is a graph X with a pair (N^+, N^-) of subsets of the set of nodes N_X such that for each edge $n \xrightarrow{e} p$, $n \in N_X^+$ and $p \in N_X^-$. A node $n \in N_X$ *has polarity* $+$ if $n \in N_X^+$, it *has polarity* $-$ if $n \in N_X^-$, and it *has polarity* $\pm$ if $n \in N_X^+ \cap N_X^-$. A *morphism* of polarized graphs $f : \mathbb{X} \to \mathbb{Y}$, where $\mathbb{X} = (X, N_X^+, N_X^-)$ and $\mathbb{Y} = (Y, N_Y^+, N_Y^-)$, is a morphism of graphs $f : X \to Y$ which preserves the polarization, i.e. such that $f(N_X^+) \subseteq N_Y^+$ and $f(N_X^-) \subseteq N_Y^-$. This defines the *category* $\mathbf{Gr}^{\pm}$ of polarized graphs. A morphism of polarized graphs $f : \mathbb{X} \to \mathbb{Y}$ is *strict*, or *strictly preserves the polarization*, if $f(N_X^+) = f(N_X) \cap N_Y^+$ and $f(N_X^-) = f(N_X) \cap N_Y^-$.

The category of polarized graphs (that will be used later in Section 4.3) is a category with an $\mathcal{M}$-partial map classifier for a family $\mathcal{M}$ which is a proper subset of all monomorphisms. Indeed, it is easy to check that strict monomorphisms form a stable system of monomorphisms (denoted $\mathcal{S}$) for the category $\mathbf{Gr}^{\pm}$, and that $\mathbf{Gr}^{\pm}$ has an $\mathcal{S}$-partial map classifier (T, η). Morphism $\eta_{\mathbb{K}}$ embeds a polarized graph $\mathbb{K}$ into $T(\mathbb{K})$, which is the disjoint union of $\mathbb{K}$ with a node $*$ having polarity $\pm$ and with an edge $*_{(n,p)} : n \to p$ for each pair of nodes $(n, p) \in (N_K^+ + \{*\}) \times (N_K^- + \{*\})$. For a partial morphism of polarized graphs (m, p) the total morphism $\varphi(m, f)$ is defined as for the case of graphs.

2.3. The AGREE *approach to graph transformation*

We start this section by considering the classified category of graphs $\mathbf{Gr}$ (with $\mathcal{M}$ the class of all monomorphisms) in order to introduce the AGREE algebraic rewriting approach in an intuitive way. An AGREE rule is made of three graph morphisms with the same source: $l : K \to L$, $r : K \to R$ and $t : K \rightarrowtail T_K$ with t monic. When specifying a transformation using an AGREE rule, the designer describes with the *left-hand side* L the items that must be present to trigger the application of the rule. The morphism l from the *gluing graph* K to L describes which items of L will be preserved, cloned, or deleted. More precisely, an item of L is *deleted* if it is not in the image of l, it is *preserved* if it is the image of exactly one item of K along l, and it is *cloned* if it is the image of more than one item of K along l. The morphism r to the *right-hand side* R defines the items that will be kept, merged, or created: items of R with a unique antecedent along r are items of K that are *kept*, different items of K are *merged* if they have the same image along r, and items of R that are not in the image of r are *created*.

Thus, given an injective *match* $m : L \rightarrowtail G$, the morphisms l and r describe how the *image* of L in G is modified by the rule. In addition, *the embedding component* $t : K \rightarrowtail T_K$, which is typical of AGREE, describes how the *context* of L in G, i.e. the items of G which are not in the image of the match, are modified by the rule. Let $\eta_L : L \rightarrowtail T(L)$ be the partial map classifier of L, as described in Section 2.1 (see Diagram 5 below). Since the embedding component $t : K \rightarrowtail T_K$ is a monomorphism, $(t : K \rightarrowtail T_K, l : K \to L)$ is a partial map $(t, l) : T_K \rightharpoonup L$. Thus $l : K \to L$ can be extended in a unique way to $\varphi(t, l) : T_K \to T(L)$ (forming a pullback), which coincides with l on the image of t and which maps each item of T_K outside the image of t to the corresponding $*$-item of $T(L)$. The match $m : L \rightarrowtail G$ determines the morphism $\overline{m} = \varphi(m, id_L) : G \to T(L)$, which is a kind of inverse of m: it maps $m(x)$ to x for each item x in L (since m is a monomorphism this is well-defined) and it maps each node and edge in the context of L in G to the corresponding $*$-item of $T(L)$. Then, the notions of deletion, preservation, and cloning can be extended to the context of L in G: an item x of G which is not in the image of m is *deleted* if $\overline{m}(x)$ is not in the image of $\varphi(t, l)$, it is *preserved* if $\overline{m}(x)$ is the image of exactly one item of T_K along $\varphi(t, l)$, and it is *cloned* if $\overline{m}(x)$ is the image of more than one item of T_K along $\varphi(t, l)$.

In order to define AGREE rewriting on any classified category, we introduce the notion of *pullback complement over a monomorphism*, which is defined and computed simply from a *pullback*. In this way we get a notion of pullback complement directly from the universal property of pullbacks.

Definition 10 (pullback complement over a monomorphism). Let $(\mathbf{C}, \mathcal{M}, T, \eta)$ be a classified category. Let $l : K \to L$ be an arrow in $\mathbf{C}$ and let $m : L \rightarrowtail G$ and $t : K \rightarrowtail T_K$ be monomorphisms in $\mathcal{M}$.
Let (g, n') be the pullback of $(\overline{m}, \varphi(t, l))$. Then there is a unique $n : K \to D$ such that square (5) ② and triangle (5) ④ commute (by the universal property of pullbacks), and in addition square (5) ② is a pullback (by pullback decomposition) and $n \in \mathcal{M}$ (because $\mathcal{M}$ is stable under pullbacks). In this situation (n, g) is called *the pullback complement of* (l, m) *over* t and pullback (5) ② is denoted PBC/t.

$$\begin{array}{ccccc} L & \xleftarrow{\ l\ } & K & & \\ \uparrow & & \uparrow & \searrow & \\ {\scriptstyle m}\downarrow & PBC/t\ ② & {\scriptstyle n}\downarrow & ④ & \\ G & \xleftarrow{\ g\ } & D & & {\scriptstyle t}\downarrow \\ {\scriptstyle \overline{m}}\downarrow & PB\ ① & {\scriptstyle n'}\downarrow & \swarrow & \\ T(L) & \xleftarrow[\varphi(t,l)]{} & T_K & & \end{array} \qquad (5)$$

Definition 11 (AGREE **rewriting).** Let $(\mathbf{C}, \mathcal{M}, T, \eta)$ be a classified category with pushouts along monomorphisms in $\mathcal{M}$. An AGREE *rule* is a triple of arrows with the same source $\rho = (l : K \to L, r : K \to R, t : K \rightarrowtail T_K)$, with t in $\mathcal{M}$.

Arrows l and r are the *left-* and *right-hand side*, respectively, and t is called the *embedding*. An AGREE *match* of rule ρ is a monomorphism $L \overset{m}{\rightarrowtail} G$ in $\mathcal{M}$. The AGREE *rewrite step* $G \Rightarrow_{\rho,m} H$ is constructed in two phases: first (n, g) is the pullback complement of (l, m) over t, so that n is in $\mathcal{M}$, second (p, h) is the pushout of (r, n).

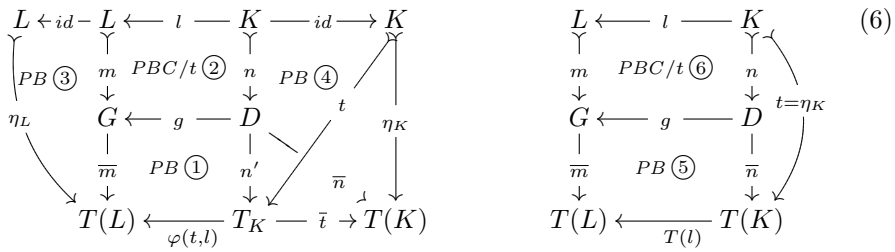
$$\begin{array}{ccccc} L & \xleftarrow{\ l\ } & K & \xrightarrow{\ r\ } & R \\ & & {\scriptstyle t}\downarrow & & \\ & & T_K & & \end{array}$$

$$\begin{array}{ccccccc} L & \xleftarrow{\ l\ } & K & & \xrightarrow{\ r\ } & R \\ \uparrow & & \uparrow & & & \downarrow \\ {\scriptstyle m}\downarrow & PBC/t & {\scriptstyle n}\downarrow & PO & & {\scriptstyle p}\downarrow \\ G & \xleftarrow{\ g\ } & D & & \xrightarrow{\ h\ } & H \end{array}$$

Remark 12. Since the pullback complement of (l, m) over t has been defined by the pullback of $\varphi(t, l)$ and $\overline{m}$ (Definition 10), an AGREE rewrite step is essentially a pullback followed by a pushout.

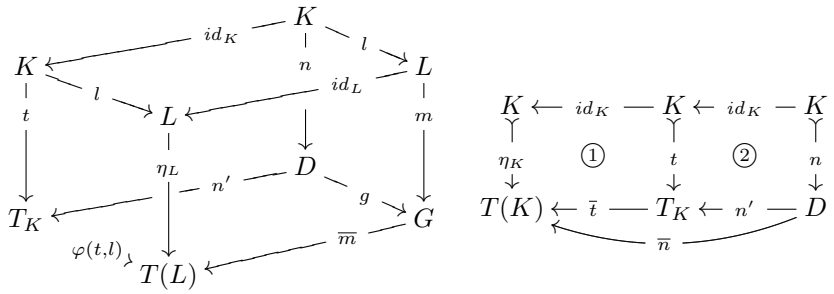
Proposition 13 says more about the commutative triangles (5) ③ and (5) ④.

Proposition 13 (some properties of PBC/t**).** *With the notations and assumptions of Definition 10, squares (6) ③ and (6) ④ are pullbacks and* $\overline{t} \circ n' = \overline{n}$. *When in addition* $T_K = T(K)$ *and* $t = \eta_K$, *then* $\varphi(t, l) = T(l)$ *and* $n' = \overline{n}$, *as depicted in Diagram (6) ⑤+⑥.*



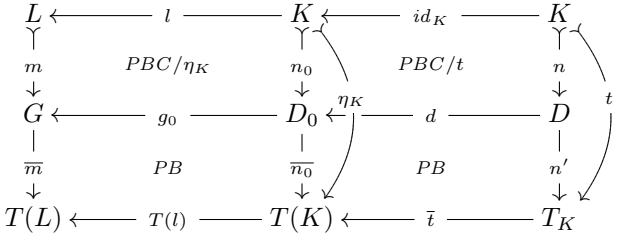
(6)

PROOF. Square (6) ③ is a pullback by definition of $\overline{m}$. For square (6) ④ consider the cube on the left of (7). The left face is a pullback by definition of $\varphi(t, l)$, the front face is a pullback by definition of $\overline{m}$, the bottom and right faces are pullbacks by definition of an AGREE rewrite step (5) ① and (5) ②, and the top face is trivially a pullback. Then by Lemma 1 the back face is also a pullback, which means that square (6) ④ is a pullback. Thus square (7) ② is a pullback, while (7) ① is a pullback by definition of $\bar{t}$. By composition ①+② is a pullback, and by definition of $\overline{n}$ we get $\bar{t} \circ n' = \overline{n}$. When $t = \eta_K$ then $\bar{t} = id_{T(K)}$, so that the equalities $\bar{t} \circ n' = \overline{n}$ (just proved) and $\varphi(t, l) = T(l) \circ \bar{t}$ (Proposition 6 item 3) become respectively $n' = \overline{n}$ and $\varphi(t, l) = T(l)$. □



(7)

Remark 14. It follows from Proposition 13 and from the composition property of pullbacks that, as depicted in Diagram (8), the pullback complement of l and m over t is composed of the pullback complement of l and m over η_K followed by the pullback complement of id_K and n_0 over t. Roughly speaking, in this way first the image of the match is modified, then its context.



(8)

Remark 15. Definitions 10 and 11 introduce the squares ①, ②, and ③ of Diagram (9). An additional square ④ can be added to the construction, making it more symmetric: arrow p' is the unique arrow such that ④ and ⑤ commute, because ② is a pushout. More properties of ④ are mentioned in Section 4.4.

$$\begin{array}{ccccccc}
L & \xleftarrow{\ l\ } & K & \xrightarrow{\ r\ } & R & & \\
\downarrow{\scriptstyle m} & PBC/t\,① & \downarrow{\scriptstyle n} & PO\,② & \downarrow{\scriptstyle p} & \searrow^{\eta_R} & \\
G & \xleftarrow{\ g\ } & D & \xrightarrow{\ h\ } & H & = & \\
\downarrow{\scriptstyle \overline{m}} & PB\,③ & \downarrow{\scriptstyle n'} & =④ & \downarrow{\scriptstyle p'}\ ⑤ & & \\
T(L) & \xleftarrow{\varphi(t,l)} & T_K & \xrightarrow{\varphi(t,r)} & T(R) & &
\end{array} \tag{9}$$

3. Locality in AGREE

In Sections 3.1 and 3.2 we study restrictions on AGREE rewriting which guarantee that all transformations are *local*, in the sense that they do not modify the strict complement of L in G. For graphs, this means that local transformations are allowed to modify only the image of L in G and the adjacent edges. Then some applications of AGREE rewriting are discussed in Section 3.3.

3.1. Local AGREE *transformation system*

Now assume that the category $\mathbf{C}$ has an *initial object* 0 and let us consider the pullback complement over the monomorphism η_0. For each object L of $\mathbf{C}$ let $0_L : 0 \to L$ denote the unique arrow from 0 to L. The notion of complement of a subset in a set can be generalized to graphs as follows: the *strict complement* of a subgraph L in a graph G is the largest subgraph $G \setminus L$ of G which does not intersect L. Then G is made of the graph L, the graph $G \setminus L$, and the set of *adjacent edges*, i.e. the set of edges of G not in L but incident to some node in L. When $G = T(L)$ and η_L is the inclusion, this is the partition of $T(L)$ as: the graph L, the graph $T(L) \setminus L$ made of the node $*$ and the loop $*_{(*,*)}$ on it, and the set of adjacent edges $*_{(n,p)} : n \to p$ for all $(n,p) \neq (*,*)$. Thus for any graph G and any $m : L \rightarrowtail G$ in $\mathcal{M}$, the partition of G is the inverse image by $\overline{m} : G \to T(L)$ of the partition of $T(L)$. In addition, $T(L) \setminus L$ is the image of $T(0_L) : T(0) \to T(L)$, where 0 denotes the empty graph and $0_L : 0 \to L$ the inclusion, so that $T(0)$ can be identified with the strict complement of L in $T(L)$. This is now generalized.

Definition 16 (strict complement).

Let $(\mathbf{C}, \mathcal{M}, T, \eta)$ be a classified category with an initial object 0. The *strict complement* of a monomorphism $m : L \rightarrowtail G$ in $\mathcal{M}$ is the pullback complement $0 \overset{0_{G\setminus L}}{\rightarrowtail} G\setminus L \overset{G\setminus m}{\rightarrow} G$ of $0 \overset{0_L}{\rightarrow} L \overset{m}{\rightarrowtail} G$ over η_0. It follows that $0_{G\setminus L}$ is in $\mathcal{M}$ and that the arrow from $G \setminus L$ to $T(0)$ in Diagram (10) is $\overline{0_{G\setminus L}}$ (by Proposition 13).

$$\begin{array}{ccccc} L & \longleftarrow{\scriptstyle 0_L} & 0 & & \\ \downarrow{\scriptstyle m} & PBC/\eta_0\ \textcircled{2} & \downarrow{\scriptstyle 0_{G\setminus L}} & \searrow^{\eta_0} & \\ G & \longleftarrow{\scriptstyle G\setminus m} & G\setminus L & & \\ \downarrow{\scriptstyle \overline{m}} & PB\ \textcircled{1} & \downarrow{\scriptstyle \overline{0_{G\setminus L}}} & & \\ T(L) & \longleftarrow{\scriptstyle T(0_L)} & T(0) & & \end{array} \tag{10}$$

Notice that the object $G \setminus L$ depends on m, not only on L and G, but this notation will always be used in a context where it is not ambiguous.

Remark 17 ($T(L) \setminus L$ is isomorphic to $T(0)$). In Diagram (10) let $G = T(L)$ and $m = \eta_L$, so that $\overline{m} = id_{T(L)}$. Since square (10) ① is a pullback it follows that $\overline{0_{T(L)\setminus L}} : T(L) \setminus L \to T(0)$ is an isomorphic.

In the general case the embedding t of an AGREE rule may give raise to non-local effects on the rewritten object, as illustrated by the following example.

Example 18. In category **Set**, the rule depicted in Figure 3 simply preserves a single element and the embedding $t : K \rightarrowtail T_K$ is the identity. If applied to set G, as shown, its effect is to delete all the elements not matched by m. We say that this rewrite step is *non-local* because it modifies the strict complement of the image of L in G.

Figure 3: An example of non-local rule.

To prevent non-local effects, restrictions on AGREE rewriting may be defined, ensuring that all transformations are *local* in the sense that they do not modify the strict complement of L in G. For graphs, this means that local transformations are allowed to modify only the image of L in G and the adjacent edges. Intuitively, in graph rewriting, an AGREE rewrite step as in Definition 11 is *local*, if the strict complement of L in G is isomorphic to the strict complement of K in D, and an AGREE rewrite rule with embedding $t : K \rightarrowtail T_K$ is *local*, if the strict complement of L in $T(L)$ is isomorphic to the strict complement of K in T_K. The next definition formalizes this idea.

Definition 19 (locality in AGREE**).** Let $(\mathbf{C}, \mathcal{M}, T, \eta)$ be a classified category with an initial object and with pushouts along monomorphisms in $\mathcal{M}$. An AGREE rewrite step as in Definition 11 is *local* if $G \setminus m : G \setminus L \to G$ is reflected along $g : D \to G$. An AGREE rule $\rho = (l, r, t)$ is *local* if $T(L) \setminus \eta_L : T(L) \setminus L \to T(L)$ is reflected along $\varphi(t, l)$.

Remark 20. There are many ways to characterise locality. Obviously, a rule is local, if and only if applying this rule to the match $\eta_L : L \rightarrowtail T(L)$ is a local

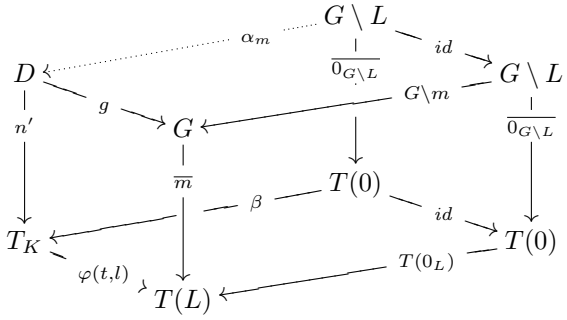
step. The definition of a local step means that there is an arrow $\alpha_m : G \setminus L \to D$ such that (11) ① is a pullback, and the definition of a local rule means that there is an arrow $\alpha_{\eta_L} : T(L) \setminus L \to T(L)$ such that (11) ② is a pullback. Since $T(0_L) \circ \overline{0_{T(L)\setminus L}} = T(L) \setminus \eta_L$ and $\overline{0_{T(L)\setminus L}} : T(L) \setminus L \to T(0)$ is an isomorphism (by Remark 17), a local rule can also be characterised by the existence of $\beta : T(0) \to T_K$ such that (11) ③ is a pullback, with $\beta \circ \overline{0_{T(L)\setminus L}} = \alpha_{\eta_L}$.

$$\begin{array}{ccc} G \setminus L & \xleftarrow{id} & G \setminus L \\ {\scriptstyle G\setminus m}\downarrow & ① & \downarrow{\scriptstyle \alpha_m} \\ G & \xleftarrow{g} & D \end{array} \qquad \begin{array}{ccc} T(L) \setminus L & \xleftarrow{id} & T(L) \setminus L \\ {\scriptstyle T(L)\setminus \eta_L}\downarrow & ② & \downarrow{\scriptstyle \alpha_{\eta_L}} \\ T(L) & \xleftarrow{\varphi(t,l)} & T_K \end{array} \qquad \begin{array}{ccc} T(0) & \xleftarrow{id} & T(0) \\ {\scriptstyle T(0_L)}\downarrow & ③ & \downarrow{\scriptstyle \beta} \\ T(L) & \xleftarrow{\varphi(t,l)} & T_K \end{array} \tag{11}$$

Theorem 21 shows that local rules always give rise to local steps.

Theorem 21 (locality of AGREE **rewrite steps).** *Let* $(\mathbf{C}, \mathcal{M}, T, \eta)$ *be a classified category with an initial object and with pushouts along monomorphisms in* $\mathcal{M}$*. Let* $\rho = (l, r, t)$ *be an* AGREE *rule. Then* ρ *is local if and only if for each* AGREE *match* $m : L \rightarrowtail G$ *the* AGREE *rewrite step* $G \Rightarrow_{\rho,m} H$ *is local.*

Proof. *(If part)* It is immediate to check that the condition of locality for ρ of Definition 19 coincides with the condition of locality for the rewrite step using rule ρ and the match $\eta_L : L \rightarrowtail T(L)$.
(Only if part) In Diagram (12), the bottom face is a pullback by Remark 20 because the rule is local, the left face is a pullback by definition of AGREE rewriting, the right face is trivially a pullback, and the front face is pullback (10) ① in the definition of strict complements. Thus by Lemma 1 there is a unique arrow $\alpha_m : G \setminus L \to D$ such that the top and back faces are pullbacks. The fact that the top face is a pullback means that the rewrite step is local. □



(12)

3.2. Strict complement of arrow

In this section, we state Proposition 24 which gives another characterisation of locality, under the following Assumption 22.

Assumption 22. *The arrow* $T(0_L)$ *is reflected along* $T(l)$ *by* $T(0_K)$*, or equivalently, square (13)* ① *is a pullback.*

Since T preserves pullbacks (item 1 in Proposition 6), a sufficient condition is that arrow 0_L is reflected along l by 0_K, or equivalently, square (13) ② is a pullback.

$$\begin{array}{ccc} T(0) & \xleftarrow{id_{T(0)}} & T(0) \\ {\scriptstyle T(0_L)}\downarrow & ① & \downarrow{\scriptstyle T(0_K)} \\ T(L) & \xleftarrow{T(l)} & T(K) \end{array} \qquad \begin{array}{ccc} 0 & \xleftarrow{id_0} & 0 \\ {\scriptstyle 0_L}\downarrow & ② & \downarrow{\scriptstyle 0_K} \\ L & \xleftarrow{l} & K \end{array} \tag{13}$$

Recall that an initial object 0 is *strict* if each arrow with target 0 must have 0 as source, or equivalently, if each arrow with target 0 is id_0. Then it is easy to check that 0_L is a monomorphism for each object L and that square (13) ② is a pullback for each arrow l. Thus since T preserves pullbacks, when the initial object 0 is strict, Assumption 22 is satisfied. All examples in this paper are based on elementary toposes, which have strict initial objects.

Proposition 23 (strict complement of arrow).

Let $(\mathbf{C}, \mathcal{M}, T, \eta)$ be a classified category with an initial object 0. Given an AGREE *rewrite step as in Definition 11, there is a unique arrow $g \setminus l : D \setminus K \to G \setminus L$ such that squares (14) ① and (14) ② commute. If in addition Assumption 22 is satisfied, then square (14) ① is a pullback. The arrow $g \setminus l$ is called the* strict complement *of l in g.*

$$\begin{array}{ccc} G & \xleftarrow{g} & D \\ {\scriptstyle G\setminus m}\uparrow & ① & \uparrow{\scriptstyle D\setminus n} \\ G\setminus L & \xleftarrow{g\setminus l} & D\setminus K \\ {\scriptstyle \overline{0_{G\setminus L}}}\downarrow & ② & \downarrow{\scriptstyle 0_{D\setminus K}} \\ T(0) & \xleftarrow{id_{T(0)}} & T(0) \end{array} \tag{14}$$

Proof. Let us prove that Diagram (15) is an instance of (1) ④+⑤.

$$\begin{array}{ccccc} & & \overset{③}{\overline{n}\circ T(l)} & & \\ T(L) & \xleftarrow{\overline{m}} & G & \xleftarrow{g} & D \\ {\scriptstyle T(0_L)}\uparrow & ① & \uparrow{\scriptstyle G\setminus m} & ② & \uparrow{\scriptstyle D\setminus n} \\ T(0) & \xleftarrow{\overline{0_{G\setminus L}}} & G\setminus L & \xleftarrow{g\setminus l} & D\setminus K \\ & & \underset{④}{\overline{0_{D\setminus K}}} & & \end{array} \tag{15}$$

Triangle (15) ③ is commutative because square (6) ⑤ is commutative. Square (15) ① is pullback (10) ①. The outer square in (15) is (16) ⑤+⑥, with (16) ⑥ another instance of pullback (10) ① and with (16) ⑤ commutative since it is the image by T of the commutative square (13) ②. Thus square (16) ⑤+⑥ is commutative. If in addition Assumption 22 is satisfied then square (16) ⑤ is a pullback, and thus square (16) ⑤+⑥ also is a pullback.

$$\begin{array}{ccccc} T(L) & \xleftarrow{T(l)} & T(K) & \xleftarrow{\overline{n}} & D \\ {\scriptstyle T(0_L)}\uparrow & ⑤ & \uparrow{\scriptstyle T(0_K)} & ⑥ & \uparrow{\scriptstyle D\setminus n} \\ T(0) & \xleftarrow{id_{T(0)}} & T(0) & \xleftarrow{\overline{0_{D\setminus K}}} & D\setminus K \end{array} \tag{16}$$

Finally, in the general case as well as in the specific case where Assumption 22 is satisfied, the result follows from the decomposition property of pullbacks. □

We conclude by providing new characterizations of locality in AGREE. In particular, the last statement of the next proposition shows that the definition of locality for AGREE proposed in [7] coincides with the definition of this paper when **C** is an elementary topos.

Proposition 24. *Let* $(\mathbf{C}, \mathcal{M}, T, \eta)$ *be a classified category with an initial object and with pushouts along monomorphisms in* $\mathcal{M}$*. If Assumption 22 is satisfied, then using the notation of Definition 19 the followings hold:*

1. *An* AGREE *rewrite step is local iff* $g \setminus l : G \setminus L \to D \setminus K$ *is an isomorphism.*
2. *An* AGREE *rule is local iff* $\varphi(t, l) \setminus l : T_K \setminus K \to T(L) \setminus L$ *is an isosomorphism.*
3. *An* AGREE *rule is local iff* $\bar{t}$ *is reflected along* $T(0_K)$*.*

Proof. 1. By Proposition 23 square (14) ① is a pullback, so that a step is local if and only if $g \setminus l : D \setminus K \to G \setminus L$ is an isomorphism.
2. This follows from the previous point, by setting $m = \eta_L$.
3. Consider Diagram (17) below. Square ① is a pullback because it is square (13) ①, therefore ①+② is a pullback (and the rule is local by Remark 20) if and only if ② is a pullback. □

$$\begin{array}{ccccc} T(0) & \xleftarrow{id} & T(0) & \xleftarrow{id} & T(0) \\ \downarrow{\scriptstyle T(0_L)} & ① & \downarrow{\scriptstyle T(0_K)} & ② & \downarrow{\scriptstyle \beta} \\ T(L) & \xleftarrow{T(l)} & T(K) & \xleftarrow{\bar{t}} & T_K \\ & \nwarrow & \overset{=}{\downarrow}{\scriptstyle \varphi(t,l)} & \nearrow & \end{array} \tag{17}$$

3.3. Applications of AGREE *rewriting*

In order to illustrate the expressive power of AGREE rules, we present three examples of local AGREE rewriting.

Example 25 (Copy of web pages). Using the AGREE approach, it is possible to model a web page copy operation. The idea is that graphs represent web pages (nodes) and hyperlinks among them (edges). It is reasonable to expect that the result of copying a page of a graph is such that the new hyperlinks are created only in the new page, and not in the pages pointing to the original one.

This effect can be modelled using the rule $(K_1 \to L_1, K_1 \to R_1, K_1 \rightarrowtail T_{K_1})$ in category **Gr** shown in Figure 4. Nodes represent web pages and edges represent links. The different node colours and shapes are used only to define the morphisms. The black node represents the page to be copied, and it is preserved by all morphisms. The white node with an inscribed symbol **c** represents the the cloned page, i.e. the copy of the black page. When the rule is applied to graph G_1, only outgoing links from the matched node are copied to the clone:

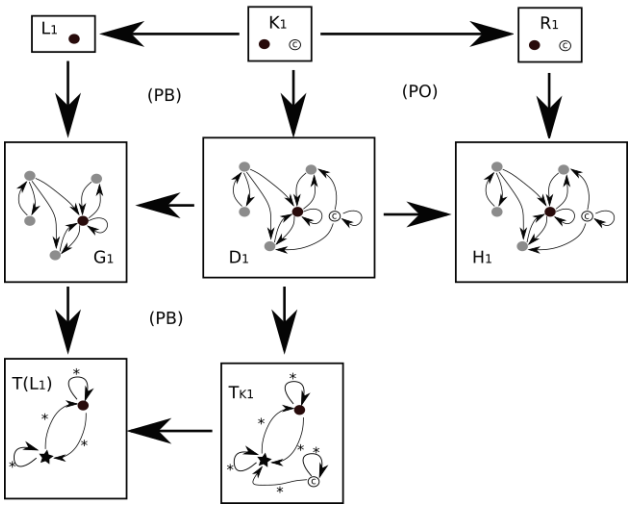


Figure 4: Rule for copying a web page and example of application.

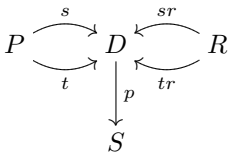
the links from other nodes to the matched node are not copied. All grey nodes of G_1 and D_1, representing the context, are mapped to $*$-nodes of $T(L_1)$ and T_{K_1}, respectively.

Example 26 (Social network anonymization). Huge network data sets, like social networks (describing personal relationships and cultural preferences) or communication networks (the graph of phone calls or email correspondents) become more and more common. These data sets are analyzed in many ways varying from the study of disease transmission to targeted advertising. Selling network data sets to third-parties is a significant part of the business model of major internet companies. Usually, in order to preserve the confidentiality of the sold data set, only "anonymized" data is released. The structure of the network is preserved, but personal identification information is erased and replaced by random numbers. This anonymized network may then be subject to further processing to make sure that it is not possible to identify the nodes of the network (see [27] for a discussion about re-identification issues). We show how AGREE rewriting can be used for such an anonymization procedure. We focus on the first task of the anonymization process: the creation of a clone of a portion of a social network in which only non-sensitive links are copied.

We assume that the data of the social network are allocated to sites, and that links among data (possibly allocated to different sites) are either public or private: the latter denote sensitive information that should not be disclosed. The AGREE rule depicted below in Figure 6 models the operation of cloning a site including all the data allocated to it. But it does so only for public links between copied data, or from copied data to data allocated to other sites. Since the number of data to be cloned is not known, we cannot represent data as

nodes of a graph, because an AGREE rule can copy only a fixed number of nodes. Instead, we exploit the fact that the application of a rule can copy an unspecified number of edges. Therefore we model social networks using a suitably defined category of *2-graphs*, i.e. graphs with *2-edges* among edges. Intuitively, nodes represent sites, (unary) edges represent data and 2-edges represent links between data.

Definition 27 (SN-graphs). The category of *SN-graphs* is the presheaf category $\mathbf{Set}^{\mathbf{C}_{SN}}$ where $\mathbf{C}_{SN}$ is the free category generated by the graph depicted to the right.



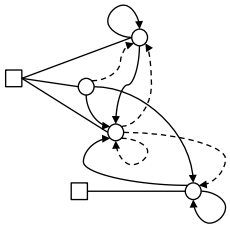


Figure 5: An SN-graph representing a social network. Data (circles) are allocated to sites (boxes) and can be linked by private (dashed) or public (solid) links.

Spelling out the definition, an SN-graph is made of a set S of *sites*, a set D of *data*, two sets P and R of *public* and *private links* respectively, and of functions mapping each data to a site ($p : D \to S$), and each link to its source and to its target data ($s, t : P \to D$ and $sr, tr : R \to D$). A morphism of SN-graphs $f : N \to N'$ is a four-tuple of functions ($f_S : S \to S', f_D : D \to D', f_P : P \to P', f_R : R \to R'$) preserving the structure in the expected way. An example of SN-graph is depicted in Figure 5. It consists of two sites (depicted as white boxes) and four data (white circles, connected with the corresponding site with an undirected edge). Plain and dashed arrows represent public and private links, respectively, drawn from the source to the target.

By Remark 4 the category of SN-graphs has partial maps classifiers and, like all toposes, it has pushouts along monomorphisms, therefore we can apply AGREE rewriting. Figure 6 shows a rule that generates a public copy of a site and its application to the SN-graph of Figure 5 (since the right-hand side of the rule is an identity morphism, the obvious pushout is not depicted). The morphisms are determined on sites by the inscriptions in the corresponding boxes (but $\boxed{2}$ is mapped vertically to $\boxed{*}$, and $\boxed{c}$ is mapped horizontally to $\boxed{1}$), and on data they are defined in the expected way preserving the layout and the links.

Clearly, the only non-obvious part of the rule of Figure 6 is the embedding morphims $t : K \rightarrowtail T_K$. It is worth showing how this morphism can be defined

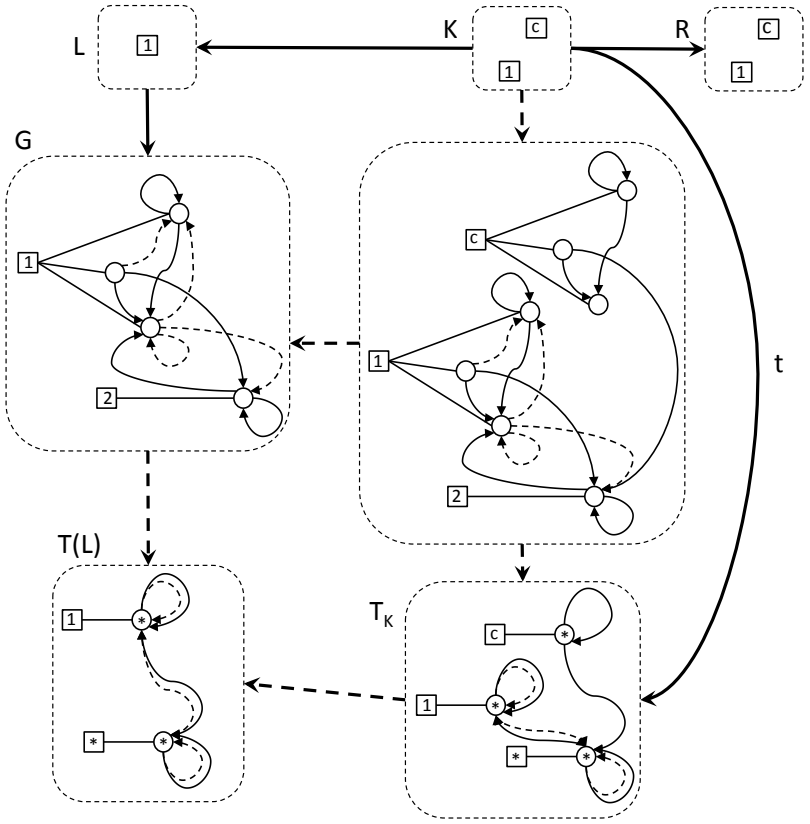


Figure 6: Rule for creating a public copy of a portion of a social network and example of application. Doubly headed arrows in the bottom graphs represent pairs of arrows in both directions.

in a reasonably easy way, at least when, as in this case, the rule is intended to copy a certain structure but not all links involving it. The idea is to start, as a first approximation of the embedding, from the partial map classifier $\eta_K : K \rightarrowtail T(K)$. As discussed later in Section 4.2, a rule with such an embedding has the same effect of a rule in the sesqui-pushout approach, thus for each cloned node it will copy all the incident edges. Therefore it is sufficient to delete from $T(K)$ the edges representing links that should not be copied. More precisely, Figure 7 shows the embedding $t : K \rightarrowtail T_K$ of our rule, together with the partial map classifier $\eta_K : K \rightarrowtail T(K)$. Comparing them, it is obvious, that T_K was obtained from $T(K)$ by deleting all dashed edges incident to the data on the cloned site (because private links should not be disclosed), the solid edges between the copied data and their clones, and the solid edge from data on other sites to the cloned data, for the same reasons as in Example 25.

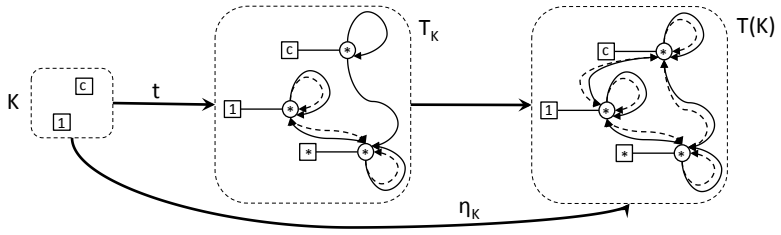


Figure 7: Comparing the embedding $t : K \rightarrowtail T_K$ with the partial map classifier η_K.

Example 28 (Forking strategies). In Unix-like operating systems there are standard mechanisms to create a new child process from its parent process. This is done through specific system calls typically named `fork()` or `clone()`. Both the parent and child processes run exactly the same code. They differ by the return code of the system call. Actually there are many variants of such system calls (see [33] for concrete examples). The variants differ in the way the address space of parent and child processes are separated or shared. For instance it is possible to create a child process with an address space completely different from the one of its parent process. It is also possible that both the parent and the child processes share parts of their execution context.

$$P \xleftarrow{s} E \xrightarrow{t} AS \xleftarrow{in} A \underset{tl}{\overset{sl}{\rightleftarrows}} L \tag{18}$$

In this example we show how such subtleties can be modeled with AGREE rules. We assume that each process can have one or more *address spaces* containing *addresses* (e.g. memory locations), and that address spaces can be shared among processes. We represent the possible states as functors from the free category $\mathbf{C}_P$ generated by the graph of Diagram (18) to **Set**, i.e. as objects of the presheaf category $\mathbf{Set}^{\mathbf{C}_P}$. Therefore a state has a set of *processes* P, a set of *address spaces* AS, and a set of *addresses* A. An *edge* $e \in E$ represents the fact that process $s(e)$ has access to the address space $t(e)$. Each address $a \in A$ belongs to the address space $in(a)$. Finally, a *link* $l \in L$ models the fact that the memory cell $sl(l)$ contains the address of location $tl(l)$.[1] For the sake of generality, we allow an address to point also to addresses outside its own address space.

States are depicted graphically as in Figure 8. Processes are represented by white boxes, address spaces by white circles and addresses by smaller black circles. Dashed arrows represent edges, solid arrows links, and undirected arrows the allocation of addresses to address spaces.

[1] Therefore each address should be the source of at most one link. Rules should be designed in order to guarantee the preservation of this property.

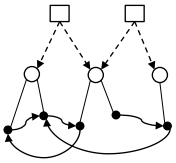


Figure 8: An abstract representation of two processes (boxes) accessing three address spaces (white circles) one of which is shared, containing a total of five addresses (black circles) linked in a suitable way.

Figure 9 shows a rule that creates a child process $\boxed{c}$ with a copy $\textcircled{c}$ of the address space $\textcircled{a}$ of the parent process $\boxed{1}$, sharing no addresses with it, and its application to the state of Figure 8. Morphisms are uniquely determined by requiring that they preserve the identity of items, as inscribed in boxes and circles, with the exception that $\boxed{c}$ and $\textcircled{c}$ are mapped to $\boxed{1}$ and $\textcircled{a}$ by morphisms from right to left, and that items whose identity cannot be preserved by the vertical morphisms are mapped to $\boxed{*}$ or $\textcircled{*}$.

Note that links among addresses in $\textcircled{a}$ are preserved in the copy, and links from addresses in $\textcircled{a}$ to other address spaces are copied as well (see the link to the first address in $\textcircled{b}$), while links *from* other address spaces to $\textcircled{a}$ are not copied. This effect is obtained by including in T_K a link from the address in $\textcircled{c}$ to that in $\textcircled{*}$, but not the reverse arrow. Note also that in T_K there is no edge from $\boxed{*}$ to $\textcircled{c}$, meaning that if the address space $\textcircled{a}$ was shared by another process its copy $\textcircled{c}$ would not be shared. A more liberal policy allowing $\textcircled{c}$ to be shared by process $\boxed{1}$ and/or by other processes can be specified easily by adding to T_K suitable edges.

An operation that creates a child process sharing an address space with the parent process can be specified easily by changing the rule so that $\boxed{1}$ and $\boxed{c}$ point to the same address space in K, and using as embedding arrow $\eta_K : K \rightarrowtail T(K)$. More elaborated operations that mix, for different address spaces of the parent process, the copying and the sharing policies can also be specified in a flexible way by AGREE rules. However, each relevant address space has to be included explicitly in the left-hand side of the rule, which means that it is not possible with a single rule to specify, for example, that the child process must have a copy of *all* the address spaces of the parent process. This level of genericity (a sort of universal quantification) is not expressible in AGREE, that in this respect is less expressive that the *gluing construction* proposed in [31], as discussed in Section 4.4.

4. Relation with other algebraic approaches to graph rewriting

In this Section we compare AGREE rewriting respectively to SqPO rewriting (Section 4.2), to graph rewriting with polarized cloning (Section 4.3) and to the gluing construction in span-categories (Section 4.4). Since all these graph transformation approaches are based on *final pullback complements*, in Section 4.1

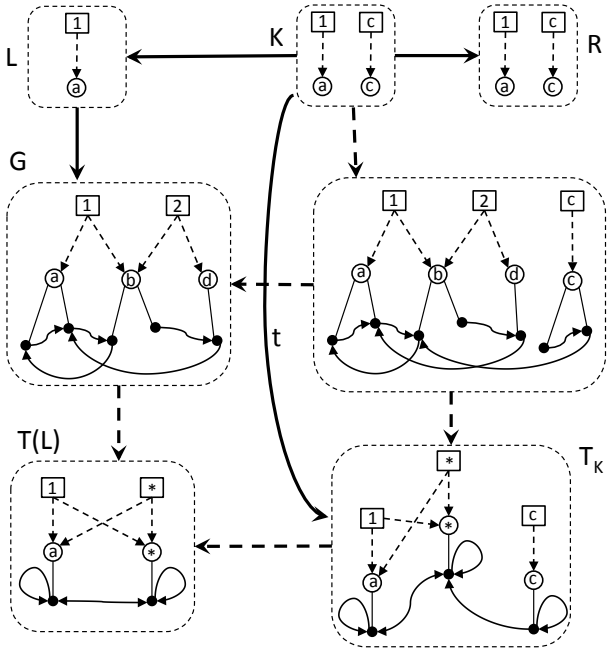


Figure 9: Rule for forking a process with an independent copy of an address space.

we recall this notion and we compare it to the notion of *final pullback complement over a monomorphism* (Definition 10). Other approaches are discussed in Section 4.5.

4.1. Final pullback complements

Final pullback complements are defined in the following way in [19].

Definition 29 (final pullback complement). Let $\mathbf{C}$ be any category. In Diagram (19) $K \overset{n}{\to} D \overset{a}{\to} G$ is a *final pullback complement* (*FPBC*) of $K \overset{l}{\to} L \overset{m}{\to} G$ if:

1. square (19) ① is a pullback, and
2. for each pullback $G \overset{m}{\leftarrow} L \overset{d}{\leftarrow} K' \overset{e}{\to} D' \overset{f}{\to} G$ and arrow $K' \overset{h}{\to} K$ such that $l \circ h = d$, there is a unique arrow $D' \overset{g}{\to} D$ such that $a \circ g = f$ and $g \circ e = n \circ h$.

$$\begin{array}{ccccc} & & d & & \\ & \swarrow & & \searrow & \\ L & \xleftarrow{l} & K & \xleftarrow{h} & K' \\ \downarrow m & & \downarrow n & & \downarrow e \\ G & \xleftarrow{a} & D & \xleftarrow{g} & D' \\ & \nwarrow & & \nearrow & \\ & & f & & \end{array} \qquad (19)$$

Then square (19) ② is a pullback by the property of pullback decomposition. When $K \overset{l}{\to} L \overset{m}{\to} G$ has a final pullback complement, it is unique up to isomorphism.

Proposition 30 shows that under suitable assumptions final pullback complements coincide with pullback complements over the partial map classifier $\eta_K : K \rightarrowtail T(K)$.

Proposition 30 (building final pullback complements). *Let $(\mathbf{C}, \mathcal{M}, T, \eta)$ be a classified category. Let $l : K \to L$ be an arrow in $\mathbf{C}$ and let $m : L \rightarrowtail G$ be a monomorphism in $\mathcal{M}$. Then the final pullback complement of l and m exists and it is the pullback complement of l and m over η_K according to Definition 10.*

First a proof of Proposition 30 under the assumption that the final pullback complement exists was given in [7]. Then Proposition 30 as stated here was proved in [32].

4.2. AGREE *subsumes* SqPO *rewriting with injective matches*

As recalled in the Introduction, in the SqPO approach [12] a rule is a span $L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R$ and a rewrite step for a match $L \stackrel{m}{\rightarrow} G$, if it exists, is made of a first phase where the final pullback complement D is constructed, followed by a pushout where the result H is computed.

Definition 31 (SqPO **rewriting).**
A SqPO *rule* is a span $L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R$. A SqPO *match* is an arrow $L \stackrel{m}{\rightarrow} G$. Then there is a SqPO *rewrite step* $G \Rightarrow^{\mathrm{SqPO}}_{\rho,m} H$ if the diagram to the right can be constructed, where (n, g) is the final pullback complement of (l, m), and (p, h) is the pushout of (r, n).

$$\begin{array}{ccccc} L & \xleftarrow{\ l\ } & K & \xrightarrow{\ r\ } & R \\ {\scriptstyle m}\downarrow & FPBC & {\scriptstyle n}\downarrow & PO & {\scriptstyle p}\downarrow \\ G & \xleftarrow{\ g\ } & D & \xrightarrow{\ h\ } & H \end{array}$$

Comparing to the definition of AGREE rewriting (Definition 11) and using Proposition 30, it is obvious that the AGREE approach is a conservative extension of the SqPO approach with monic matches, because the two coincide if the embedding of the AGREE rule is the partial map classifier η_K of K. More precisely, as stated by the following proposition, the application of rule $\rho = (l, r)$ at monic match m using the SqPO approach has exactly the same effect of the application to m of the same rule ρ enriched with the embedding $\eta_K : K \rightarrowtail T(K)$ using the AGREE approach.

Proposition 32 (AGREE **vs.** SqPO**).** *Let $(\mathbf{C}, \mathcal{M}, T, \eta)$ be a classified category with pushouts along monomorphisms, where $\mathcal{M}$ is the set of all monomorphisms. Let $L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R$ be a* SqPO *rule and let $\eta_K : K \rightarrowtail T(K)$ be the partial map classifier of K. Then for each monomorphisms $m : L \rightarrowtail G$*

$$G \Rightarrow^{\mathrm{SqPO}}_{(l,r),m} H \qquad \textit{if and only if} \qquad G \Rightarrow^{\mathrm{AGREE}}_{(l,r,\eta_K),m} H$$

4.3. AGREE *subsumes polarized node cloning on graphs*

We now show that AGREE rewriting allows to simulate rewriting with polarized cloning on graphs, which is defined in [17] by using the polarized graphs of Definition 9. Polarization is used in rewriting to control the copies of the adjacent edges (i.e. the edges that are not matched but incident to the matched nodes).

Remark 33. The *underlying graph* of a polarized graph $\mathbb{X} = (X, N_X^+, N_X^-)$ is X. This defines a functor $Q : \mathbf{Gr}^{\pm} \to \mathbf{Gr}$ which has both a right and a left adjoint functor denoted P and $P^{\pm} : \mathbf{Gr} \to \mathbf{Gr}^{\pm}$ resp., i.e. $P^{\pm} \dashv Q \dashv P$. Functor P maps each graph X to the polarized graph *induced by* X, defined as $\mathbb{X} = (X, N_X, N_X)$, and each graph morphism $f : X \to Y$ to itself; it is easy to check that $P(f) : P(X) \to P(Y)$ is a *strict* polarized graph morphism. Furthermore we have that $Q \circ P = Id_{\mathbf{Gr}}$, and we denote the unit of adjunction $Q \dashv P$ as $u : Id_{\mathbf{Gr}^{\pm}} \dot{\to} P \circ Q$, thus $u_{\mathbb{X}} : \mathbb{X} \to P(Q(\mathbb{X}))$. Functor $P^{\pm}$ maps each graph X to the polarized graph $\mathbb{X} = (X, N_X^+, N_X^-)$, where a node is in N_X^+ (resp. in N_X^-) if and only if it has at least one outgoing (resp. incoming) edge in X. Since Q has a left adjoint, we have that Q preserves limits and in particular pullbacks.

The category $\mathbf{Gr}^{\pm}$ has final pullback complements along strict monomorphisms: their construction is given in [16, Appendix].

Definition 34 (PSqPO **rewriting).** A PSqPO *rule* ρ is made of a span of graphs $L \overset{l}{\leftarrow} K \overset{r}{\to} R$ and a polarized graph $\mathbb{K} = (K, N_K^+, N_K^-)$ with underlying graph K. A PSqPO *match* of the PSqPO rewrite rule ρ is a monomorphism $m : L \rightarrowtail G$ in $\mathbf{Gr}$. A PSqPO *rewrite step* $G \Rightarrow^{\mathrm{PSqPO}}_{\rho,m} H$ is constructed as follows:

1. The left-hand side l of the rule ρ gives rise to an arrow $\widehat{l} = P(l) \circ u_{\mathbb{K}} : \mathbb{K} \to P(L)$ in $\mathbf{Gr}^{\pm}$. The match m gives rise to a strict monomorphism $P(m) : P(L) \rightarrowtail P(G)$ in $\mathbf{Gr}^{\pm}$. Then $\mathbb{K} \overset{n}{\to} \mathbb{D} \overset{g}{\to} P(G)$ is constructed as the final pullback complement of $\mathbb{K} \overset{\widehat{l}}{\to} P(L) \overset{P(m)}{\to} P(G)$ in category $\mathbf{Gr}^{\pm}$.
2. Since $Q(\mathbb{K}) = K$, we get $Q(n) : K \to Q(\mathbb{D})$ in $\mathbf{Gr}$. Then $R \overset{p}{\to} H \overset{h}{\leftarrow} Q(\mathbb{D})$ is built as the pushout of $R \overset{r}{\leftarrow} K \overset{Q(n)}{\to} Q(\mathbb{D})$ in category $\mathbf{Gr}$.

$$\begin{array}{ccc} L & & P(L) \xleftarrow{\ \widehat{l}\ } \mathbb{K} \\ \uparrow & & \uparrow \quad\quad \uparrow \\ m & \overset{P}{\longmapsto} & P(m) \quad FPBC \quad n \\ \downarrow & & \downarrow \quad\quad \downarrow \\ G & & P(G) \xleftarrow{\ g\ } \mathbb{D} \end{array} \qquad \begin{array}{ccc} K & \xrightarrow{\ r\ } & R \\ \downarrow & & \downarrow \\ Q(n) & PO & p \\ \downarrow & & \downarrow \\ Q(\mathbb{D}) & \xrightarrow{\ h\ } & H \end{array}$$

Recall that, as observed in Section 2.2, category $\mathbf{Gr}^{\pm}$ has an $\mathcal{S}$-partial map classifier (T, η), where $\mathcal{S}$ is made of the strict monomorphisms. This will be exploited in the next result, which can be seen as a generalization of Proposition 32.

Proposition 35 (AGREE **vs.** PSqPO**).** *Let ρ be a* PSqPO *rule made of span $L \xleftarrow{l} K \xrightarrow{r} R$ and polarized graph $\mathbb{K} = (K, N_K^+, N_K^-)$. Consider the component on $\mathbb{K}$ of the natural transformation $\eta : Id_{\mathbf{Gr}^\pm} \dot{\to} T$, and let $T_K = Q(T(\mathbb{K}))$ and $t = Q(\eta_{\mathbb{K}}) : Q(\mathbb{K}) \to Q(T(\mathbb{K}))$, thus $t : K \to T_K$. Furthermore, let $m : L \rightarrowtail G$ be a monomorphism. Then*

$$G \Rightarrow^{\text{PSqPO}}_{\rho,m} H \qquad \textit{if and only if} \qquad G \Rightarrow^{\text{AGREE}}_{(l,r,t),m} H$$

Proof. The first phase of PSqPO rewriting consists of building the *FPBC* of $(P(m), \widehat{l})$ in category $\mathbf{Gr}^\pm$. According to Proposition 30, since $P(m)$ is strict, such a *FPBC* can be obtained as square (20) ①, where both squares ① and ② are pullbacks in $\mathbf{Gr}^\pm$. The second phase consists of taking the pushout of arrows $K \xrightarrow{r} R$ and $Q(n) : K \to Q(\mathbb{D})$ in $\mathbf{Gr}$. By applying functor Q to the left part of Diagram (20) we obtain its right part, where both squares ③ and ④ are pullbacks because Q preserves limits. In fact, recall that $Q \circ P = Id_{\mathbf{Gr}}$, that $K = Q(\mathbb{K})$ and that $t = Q(\eta_{\mathbb{K}})$; the fact that $T(L) = Q(T(P(L)))$ can be checked easily by comparing the construction of the $\mathcal{S}$-partial map classifiers in $\mathbf{Gr}$ and in $\mathbf{Gr}^\pm$.

$$\begin{array}{ccc} P(L) & \longleftarrow \widehat{l} \longrightarrow & \mathbb{K} \\ {\scriptstyle P(m)}\downarrow & PB\,① & \downarrow{\scriptstyle n} \\ P(G) & \longleftarrow g \longrightarrow & \mathbb{D} \\ {\scriptstyle \overline{P(m)}}\downarrow & PB\,② & \downarrow{\scriptstyle \overline{n}} \\ T(P(L)) & \longleftarrow T(\widehat{l}) - & T(\mathbb{K}) \end{array} \qquad \begin{array}{ccc} L & \longleftarrow l \longrightarrow & K \\ {\scriptstyle m}\downarrow & PB\,③ & \downarrow{\scriptstyle Q(n)} \\ G & \longleftarrow & Q(\mathbb{D}) \\ {\scriptstyle \overline{m}}\downarrow & PB\,④ & \downarrow{\scriptstyle Q(n)'} \\ T(L) & \longleftarrow & T_K = Q(T(\mathbb{K})) \end{array} \tag{20}$$

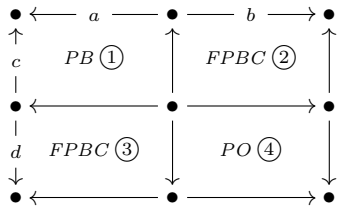
Now, the first phase of AGREE rewriting with rule (l, r, t) and match m consists of taking the pullback in $\mathbf{Gr}$ of $\overline{m}$ and the only arrow $T_K \to T(L)$ that makes square (20) ③+④ a pullback. This arrow is precisely $Q(T(\widehat{l}))$, and therefore the pullback is exactly (20) ④. The second phase consists of taking the pushout of $K \xrightarrow{r} R$ and of the only arrow $K \to Q(\mathbb{D})$ that makes the diagram commute; but $Q(n)$ is such an arrow, thus the pushout is the same computed by the PSqPO approach. This concludes the proof. □

4.4. Span rewriting subsumes AGREE *with monic right-hand side*

The *gluing construction* is introduced in the context of graph rewriting in span-categories in [31]. It is compared to AGREE rewriting in [32]: an AGREE rewrite step can be seen as a gluing construction under the assumption that the rule is *right-linear*, i.e. that the right-hand side of the rule is monic.

Definition 36 (gluing construction). In any category $\mathbf{C}$, the *gluing* of two spans (a, b) (the *rule*) and (c, d) (the *match*), if it exists, is given by Diagram (21)

where square ① is a pullback, squares ② and ③ are final pullback complements and square ④ is a pushout.



(21)

Gluing construction [31]

When c is the identity only ③ and ④ are left, so that obviously the gluing construction coincides with SqPO rewriting. When d is the identity only ① and ② are left, and it can be proved that the gluing construction, under some assumptions, coincides with AGREE rewriting in this case: this is Proposition 37, which is proved in [32, Theorem 11].

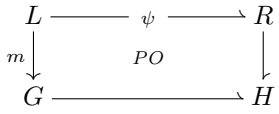
The proof proceeds by comparing ①+② in (21) with ③+④ in (9), which means, by proving that ④ in (9) is a final pullback complement when r in (9) is a monomorphism.

Proposition 37 (gluing construction vs. AGREE**).** *An* AGREE *rewrite step with respect to an* AGREE *rule with monic right-hand side is a gluing construction.*

The gluing construction in its generality is able to express transformations that cannot be specified with AGREE rules. As shown in [31], a single rule can specify the firing of a Petri Net transition, independently of the number of pre- and post-conditions, by exploiting a form of universal quantification. Therefore, making reference to Example 28, it is possible to specify the creation of a child process with a copy of *all* the address spaces of the parent process, which is impossible in AGREE. This greater expressiveness requires to specify the match as a suitable span, instead of as a simple monomorphism as in AGREE.

4.5. Comparison with other approaches

The idea of controlling explicitly how the right-hand side of a rule is embedded in the context graph is not new in graph rewriting, as it is a standard ingredient of the algorithmic approaches. For example, in Node Label Controlled (NLC) graph rewriting and its variations [26], productions are equipped with *embedding rules* which allow one to specify how the right-hand side of a production has to be embedded in the context graph obtained by deleting the corresponding left-hand side.

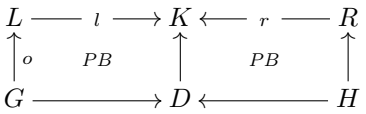


SPO rewrite step [30]

In the SPO approach [30, 23], a rule is a *partial* graph morphism $\psi : L \rightharpoonup R$ and a match is a total morphism $m : L \to G$. A graph G rewrites into a graph H using rule ψ and match m if a square like the one above can be constructed, which is a pushout in the category of graphs and partial morphisms. Deleting, adding and merging items can easily be specified with SPO rules and the approach is appropriate for specifying deletion of nodes without knowing the precise embedding of the deleted nodes, thanks to partial morphisms. The deletion of a node causes the deletion of all edges connected to it (symmetrically to what happens in the cloning of nodes in the SqPO approach). However, since a rule is defined as a single graph morphism, cloning items cannot be specified directly in SPO.

In the framework of *adaptive star grammars* [15], node cloning is performed by means of rewrite rules of the form $S ::= R$ where graph S has a shape of a star (i.e. a central node with several neighbors that are not connected to one another) and R is a graph. The cloning operation, see [15, Definitions 5 and 6], shares the same restrictions as the sesqui-pushout approach: nodes are cloned with all their incident edges.

The *double-pullback approach* (DPB) presented in [3] is an alternative to the algebraic approaches discussed in the introduction. The main differences with respect to the SPO, DPO and SqPO approaches are that the match $m : L \to G$ of L in G is replaced by an *occurrence* $o : G \to L$ of L in G and pushouts are replaced by pullbacks. Rules are cospans of the form $L \to K \leftarrow R$.



DPB rewrite step [3]

The first step constructs graph D as a *pullback complement PBC* and the second step builds H as a pullback. As in the case of building a pushout complement, the pullback complement is not a universal construction and conditions for its existence and uniqueness must be checked. It is proved in [3] that DPB subsumes DPO under suitable assumptions. This approach does not satisfy locality properties in general.

5. Conclusion

A new approach to algebraic graph rewriting, called AGREE is introduced in this paper. The main new feature provided by this approach is the possibility to specify in a rule which edges are cloned as a side effect of the copy of a node. This feature, described by the embedding morphism t, offers new facilities to specify applications in which nodes can be copied in unknown contexts. In such cases, the left-hand sides of the rules are not enough to describe all possible edges that are copied together with a node as illustrated in the examples of section 3.3.

The expressive power of AGREE allows one to define rules with the ability to modify the context surrounding the image of the left-hand side. Such global transformations, which are not possible in classical algebraic approaches such as DPO, SPO, SqPO, etc., led us to investigate and define precisely the notions of local transformations and rules. To our knowledge these notions have not been defined before in the literature. Furthermore, we propose sufficient conditions on AGREE rules that ensure the locality of AGREE transformations.

Ensuring the parallel independence of rewrite rules is a topic that has been studied for the most important algebraic graph rewrite methods [13, 24, 12, 29, 4, 22, 10]. Unlike such previous contributions, the AGREE approach proposes a somewhat elaborated cloning mechanism which requires new parallel independence analysis. Recently, necessary and sufficient conditions that characterise parallel independence of AGREE rules have been proposed in [32]. These conditions generalise those proposed earlier in [11].

Concerning the applicability of our approach to other structures, in practice the requirement of the existence of partial maps classifiers might be a hurdle for its use. Actually, AGREE rewriting works well in the framework of categories of typed/colored graphs, which are used in several applications, as they are slice categories over graphs, and thus toposes. However, extension of AGREE to categories of attributed graphs [18], which are not toposes, is not always possible as partial map classifiers may not exist for attributes. An attempt to extend the AGREE approach to the framework of attributed graph categories has been proposed in [8, 9] where rules (two spans together with their typing morphisms) and matches (a classical match and a co-match) are more complex than those involved in AGREE rewrite steps.

Acknowledgments

We are grateful to the anonymous reviewers of former versions of this paper for the insightful and constructive criticisms.

References

[1] Adamek, J., Herrlich, H., Strecker, G.: Abstract and Concrete Categories. Wiley Interscience (1990), `http://katmat.math.uni-bremen.de/acc/`, on-line version (2004)

[2] Andries, M., Engels, G., Habel, A., Hoffmann, B., Kreowski, H.J., Kuske, S., Plump, D., Schürr, A., Taentzer, G.: Graph transformation for specification and programming. Science of Computer Programming 34(1), 1 – 54 (1999)

[3] Bauderon, M., Jacquet, H.: Pullback as a generic graph rewriting mechanism. Applied Categorical Structures 9(1), 65–82 (2001)

[4] Bonchi, F., Gadducci, F., Heindel, T.: Parallel and sequential independence for borrowed contexts. In: ICGT 2008. LNCS, vol. 5214, pp. 226–241. Springer (2008)

[5] Cockett, J., Lack, S.: Restriction categories II: partial map classification. TCS 294(1–2), 61–102 (2003)

[6] Coecke, B., Duncan, R.: Interacting quantum observables: Categorical algebra and diagrammatics. New J. Phys. 043016(13) (2011)

[7] Corradini, A., Duval, D., Echahed, R., Prost, F., Ribeiro, L.: AGREE - algebraic graph rewriting with controlled embedding. In: ICGT 2015. LNCS, vol. 9151, pp. 35–51. Springer (2015)

[8] Corradini, A., Duval, D., Echahed, R., Prost, F., Ribeiro, L.: The pullback-pushout approach to algebraic graph transformation. In: ICGT 2017. LNCS, vol. 10373, pp. 3–19. Springer (2017)

[9] Corradini, A., Duval, D., Echahed, R., Prost, F., Ribeiro, L.: The PBPO graph transformation approach. J. Log. Algebr. Meth. Program. 103, 213–231 (2019)

[10] Corradini, A., Duval, D., Löwe, M., Ribeiro, L., Machado, R., Costa, A., Azzi, G.G., Bezerra, J.S., Rodrigues, L.M.: On the essence of parallel independence for the double-pushout and sesqui-pushout approaches. In: Graph Transformation, Specifications, and Nets - In Memory of Hartmut Ehrig. vol. 10800, pp. 1–18. Springer (2018)

[11] Corradini, A., Duval, D., Prost, F., Ribeiro, L.: Parallelism in AGREE transformations. In: ICGT 2016. LNCS, vol. 9761, pp. 37–53. Springer (2016)

[12] Corradini, A., Heindel, T., Hermann, F., König, B.: Sesqui-pushout rewriting. In: ICGT 2006. LNCS, vol. 4178, pp. 30–45. Springer (2006)

[13] Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R., Löwe, M.: Algebraic approaches to graph transformation - part I: basic concepts and double pushout approach. In: Rozenberg [35], pp. 163–246

[14] 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: FSTTCS 2012. LIPIcs, vol. 18, pp. 276–288. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2012)

[15] Drewes, F., Hoffmann, B., Janssens, D., Minas, M.: Adaptive star grammars and their languages. TCS 411(34-36), 3090–3109 (2010)

[16] Duval, D., Echahed, R., Prost, F.: Graph rewriting with polarized cloning. CoRR abs/0911.3786 (2009)

[17] Duval, D., Echahed, R., Prost, F.: Graph transformation with focus on incident edges. In: ICGT 2012. LNCS, vol. 7562, pp. 156–171. Springer (2012)

[18] Duval, D., Echahed, R., Prost, F., Ribeiro, L.: Transformation of attributed structures with cloning. In: FASE 2014. LNCS, vol. 8411, pp. 310–324. Springer (2014)

[19] Dyckhoff, R., Tholen, W.: Exponentiable morphisms, partial products and pullback complements. Journal of Pure and Applied Algebra 49(1-2), 103–116 (1987)

[20] Echahed, R.: Inductively sequential term-graph rewrite systems. In: ICGT 2008. LNCS, vol. 5214, pp. 84–98. Springer (2008)

[21] Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science. An EATCS Series, Springer (2006)

[22] Ehrig, H., Habel, A., Lambers, L.: Parallelism and concurrency theorems for rules with nested application conditions. ECEASST 26 (2010)

[23] Ehrig, H., Heckel, R., Korff, M., Löwe, M., Ribeiro, L., Wagner, A., Corradini, A.: Algebraic approaches to graph transformation - part II: single pushout approach and comparison with double pushout approach. In: Rozenberg [35], pp. 247–312

[24] Ehrig, H., Löwe, M.: Parallel and distributed derivations in the single-pushout approach. TCS 109(1&2), 123–143 (1993)

[25] Ehrig, H., Pfender, M., Schneider, H.J.: Graph-grammars: An algebraic approach. In: 14th Annual Symposium on Switching and Automata Theory. pp. 167–180 (1973)

[26] Engelfriet, J., Rozenberg, G.: Node replacement graph grammars. In: Rozenberg [35], pp. 1–94

[27] Hay, M., Miklau, G., Jensen, D., Towsley, D.F., Li, C.: Resisting structural re-identification in anonymized social networks. VLDB J. 19(6), 797–823 (2010)

[28] Heindel, T.: Adhesivity with partial maps instead of spans. Fundamenta Informaticae 118(1-2), 1–33 (2012)

[29] Hermann, F., Corradini, A., Ehrig, H.: Analysis of permutation equivalence in $\mathcal{M}$-adhesive transformation systems with negative application conditions. Mathematical Structures in Computer Science 24(4) (2014)

[30] Löwe, M.: Algebraic approach to single-pushout graph transformation. TCS 109(1&2), 181–224 (1993)

[31] Löwe, M.: Graph rewriting in span-categories. In: ICGT 2010. LNCS, vol. 6372, pp. 218–233. Springer (2010)

[32] Löwe, M.: Characterisation of parallel independence in agree-rewriting. In: Lambers, L., Weber, J. (eds.) ICGT 2018. LNCS, vol. 10887, pp. 118–133. Springer (2018)

[33] Mitchell, M., Oldham, J., Samuel, A.: Advanced Linux Programming. Landmark Series, New Riders (2001)

[34] Rosselló, F., Valiente, G.: Chemical graphs, chemical reaction graphs, and chemical graph transformation. ENTCS 127(1), 157–166 (2005)

[35] Rozenberg, G. (ed.): Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations. World Scientific (1997)