AGREE – Algebraic Graph Rewriting with Controlled Embedding*

A. Corradini[1] and D. Duval[2] and R. Echahed[3] and F. Prost[3] and L. Ribeiro[4]

[1] Dipartimento di Informatica, Università di Pisa
[2] LJK - Université de Grenoble Alpes and CNRS
[3] LIG - Université de Grenoble Alpes and CNRS
[4] INF - Universidade Federal do Rio Grande do Sul

Abstract. The several algebraic approaches to graph transformation proposed in the literature all ensure that if an item is preserved by a rule, so are its connections with the context 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 conservative extension of classical algebraic approaches to graph transformation, for the case of monic matches, where rules allow one to specify how the embedding of preserved items should be carried out.

1 Introduction

Graphs are used to describe a wide range of situations in a precise yet intuitive way. Different kinds of graphs are used in modelling techniques depending on the investigated fields, which include computer science, chemistry, biology, quantum computing, etc. When system states are represented by graphs, it is natural to use 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 a concrete algorithm, the result of applying a rule to a graph (see e.g. [14, 11]), and (ii) the algebraic approaches which define abstractly a graph transformation step using basic constructs borrowed from category theory. In this paper we will consider the latter.

The basic idea of all approaches is the same: states are represented by graphs and state changes are represented by rules that modify graphs. The differences are the kind of graphs that may be used, and the definitions of when and how rules may be applied. One critical point when defining graph transformation is that one cannot delete or copy part of a graph without considering the effect of the operation on the rest of the graph, because deleted/copied items may be linked to others. For example, rule $\rho 1$ in Figure 1(a) specifies that a node shall be deleted and rule $\rho 2$ that a node shall be duplicated (C indicates the copy).

* This work has been partly funded by projects CLIMT (ANR/(ANR-11-BS02-016), TGV (CNRS-INRIA-FAPERGS/(156779 and 12/0997-7)), VeriTeS (CNPq 485048/2012-4 and 309981/2014-0), PEPS égalité (CNRS).

What should be the result of applying these rules to the grey node of graph G in Figure 1(b)? Different approaches give different answers to this question.

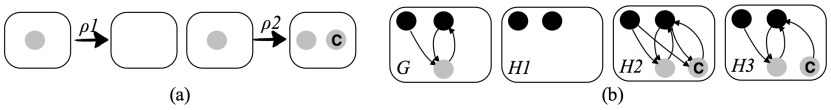
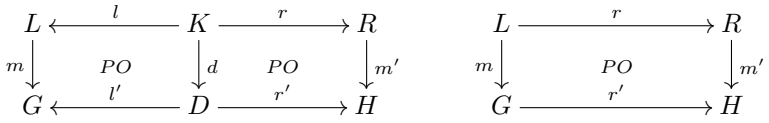


Fig. 1. (a) Delete/Copy Rules (b) Resulting Graphs

The most popular algebraic approaches are the double-pushout (DPO) and the single-pushout (SPO), which can be illustrated as follows:



Double pushout rewrite step Single pushout rewrite step

In the DPO approach [13, 6], a rule is defined as a span $\rho = L \leftarrow K \rightarrow R$ and a match is a morphism $m : L \rightarrow G$. A graph G rewrites into a graph H using rule ρ and match m if the diagram above to the left can be constructed, where both squares are pushouts. Conditions for the existence and uniqueness of graph D need to be studied explicitly, since it is not a universal construction. With DPO rules it is easy to specify the addition, deletion, merging or cloning of items, but their applicability is limited. For example, rule $\rho 1$ of Figure 1 is not applicable to the grey node of G (as it would leave dangling edges), and a rule like $\rho 2$ is usually forbidden as the *pushout complement* D would not be unique.

In the SPO approach [16, 12], a rule is a *partial* graph morphism $\psi : L \rightarrow R$ and a match is a total morphism $m : L \rightarrow G$. A graph G rewrites into a graph H using rule ψ and match m if a square like the one above to the right 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 in unknown context, thanks to partial morphisms. The deletion of a node causes the deletion of all edges connected to it, and thus applying rule $\rho 1$ to G would result in graph $H1$ in Figure 1(b). However, since a rule is defined as a single graph morphism, copying of items (as in rule $\rho 2$) cannot be specified directly in SPO.

A more recent algebraic approach is the sesqui-pushout approach (SqPO) [5]. Rules are spans like in the DPO, but in the left square of a rewriting step D is built as a *final pullback complement*. This characterises D with a universal property, enabling to apply rule $\rho 1$, obtaining the same result as in the SPO approach ($H1$), as well as rule $\rho 2$, obtaining $H2$ as result. Also $\rho 2$ has a side effect: when a node is copied all the edges of the original node are copied as well. Rules do not specify explicitly which context edges are deleted/copied, this is determined by the categorical constructions that define rule application. In general, in all algebraic approaches, the items that are preserved by a rule will retain the connections they have with items which are not in the image of the match. This holds also for items that are copied in the SqPO approach.

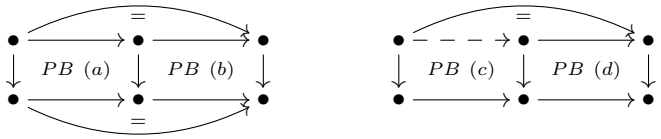
However, there are situations in which the designer should be able to specify which of the edges connecting the original node should be copied when a node is copied, depending for example on the direction of the edges (incoming or outgoing), or on their labels, if any. For example, if the graphs of Figure 1 represent web pages (nodes) and hyperlinks among them (edges) it would be reasonable to expect that the result of copying the grey page of G with rule $\rho 2$ would be graph $H3$ rather than $H2$, so that new hyperlinks are created only in the new page, and not in the pages pointing to the original one. As another example, the `fork` and `clone` system commands in Linux both generate a clone of a process, but with different semantics. Both commands precisely differ in the way the environment of the cloned process is dealt with: see [18] for more details.

These examples motivate the rewriting approach that we introduce in this paper. In order to give the designer the possibility of controlling how the nodes that are preserved or cloned by a rule are embedded in the context graph, we propose a new algebraic approach to graph transformation where rules are triples of arrows with the same source $r = (K \stackrel{l}{\to} L, K \stackrel{r}{\to} R, K \stackrel{t}{\rightarrowtail} T_K)$. Arrows l and r are the usual left- and right-hand sides, while t is a mono called the *embedding*: it will play a role in controlling which edges from the context are copied. The resulting rewriting approach, called AGREE (for Algebraic Graph Rewriting with controllEd Embedding) is presented in Sect. 3. As usual for the algebraic approaches, AGREE rewriting will be introduced abstractly for a category satisfying suitable requirements, that will be introduced in Sect. 2. For the knowledgeable reader we anticipate that we will require the existence of *partial map classifiers* [3]. After discussing an example of social networks in Sect. 4, in Sect. 5 we show that AGREE rewriting can simulate both SqPO rewriting (restricted to mono matches) and *rewriting with polarised cloning* [8]. Finally some related and future works are briefly discussed in Sect. 6.

2 Preliminaries

We start recalling some definitions and a few properties concerning pullbacks, partial maps and partial map classifiers: a survey on them can be found in [2, 3]. Let $\mathbf{C}$ be a category with all pullbacks. We recall the following properties:

- monos are stable under pullbacks, i.e. if $B' \stackrel{f'}{\leftarrow} A' \stackrel{m'}{\to} A$ is the pullback of $B' \stackrel{m}{\rightarrowtail} B \stackrel{f}{\leftarrow} A$ and m is mono, then m' is mono as well.
- the *composition* property of pullbacks: in a commutative diagram as below on the left, if squares (a) and (b) are pullbacks, so is the composed square;



- and the *decomposition* property: in a commutative diagram as the one made of solid arrows above on the right, if square (d) and the outer square are

pullbacks, then there is a unique arrow (the dotted one) such that the top triangle commutes and square (c) is a pullback.

A *stable system of monos* of $\mathbf{C}$ is a family $\mathcal{M}$ of monos including all isomorphisms, closed under composition, and (*stability*) such that if (f', m') is a pullback of (m, f) and $m \in \mathcal{M}$, then $m' \in \mathcal{M}$. An *$\mathcal{M}$-partial map* over $\mathbf{C}$, denoted $(m, f) : Z \rightharpoonup Y$, is a span made of a mono $m : X \rightarrowtail Z$ 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 $f' \circ h = f$.

Category $\mathbf{C}$ has an *$\mathcal{M}$-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 following holds: 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 (1) is a pullback.

In this case it can be shown (see [3]) that $\eta_Y \in \mathcal{M}$ for each object $Y \in \mathbf{C}$, that T preserves pullbacks, and that the natural transformation η is *cartesian*, which means that for each $f : X \to Y$ the naturality square (2) is a pullback. For each mono $m : X \rightarrowtail Z$ in $\mathcal{M}$ we will use the notation $\overline{m} = \varphi(m, id_X)$, thus $\overline{m}$ is defined by the pullback square (3).

$$\begin{array}{ccc} X & \xrightarrow{\quad f \quad} & Y \\ {\scriptstyle m}\downarrow & PB & \downarrow{\scriptstyle \eta_Y} \\ Z & \xrightarrow{\ \varphi(m,f)\ } & T(Y) \end{array} \qquad (1)$$

$$\begin{array}{ccc} X & \xrightarrow{\quad f \quad} & Y \\ {\scriptstyle \eta_X}\downarrow & PB & \downarrow{\scriptstyle \eta_Y} \\ T(X) & \xrightarrow{\ T(f)\ } & T(Y) \end{array} \qquad (2)$$

$$\begin{array}{ccc} X & \xrightarrow{\quad id_X \quad} & X \\ {\scriptstyle m}\downarrow & PB & \downarrow{\scriptstyle \eta_X} \\ Z & \xrightarrow{\ \overline{m}\ } & T(X) \end{array} \qquad (3)$$

Before discussing some examples of categories that have $\mathcal{M}$-partial map classifiers, let us recall the definition of some categories of graphs.

Definition 1 (graphs, typed graphs). *The category of* graphs $\mathbf{Gr}$ *is defined as follows. A* graph X *is 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* $p = t_X(e)$*. A* morphism *of graphs* $f : X \to Y$ *is made of two functions* $f : N_X \to N_Y$ *and* $f : E_X \to E_Y$*, such that* $f(n) \xrightarrow{f(e)} f(p)$ *in* Y *for each edge* $n \xrightarrow{e} p$ *in* X*.*

Given a fixed graph Type, called type graph*, the category of* graphs typed over *Type is the slice category* $\mathbf{Gr} \downarrow Type$*.*

Definition 2 (polarized graphs [9]). *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* $n \xrightarrow{e} p$ *one has* $n \in N_X^+$ *and* $p \in N_X^-$*. 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* $f : X \to Y$ *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* $f(N_X^-) = f(N_X) \cap N_Y^-$*.*

2.1 Examples of Partial Map Classifiers

Informally, if $(m, f) : Z \rightharpoonup Y$ is a partial map, a total arrow $\varphi(m, f) : Z \to T(Y)$ representing it should agree with (m, f) on the "items" of Z on which it 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. For example, in **Set** the partial map classifier (T, η) is defined as $T(X) = X + \{*\}$ and $T(f) = f + id_{\{*\}}$ for functor T, while the natural transformation η is made of the inclusions $\eta_X : X \to X + \{*\}$. For each partial function $(m, f) : Z \rightharpoonup Y$, 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.

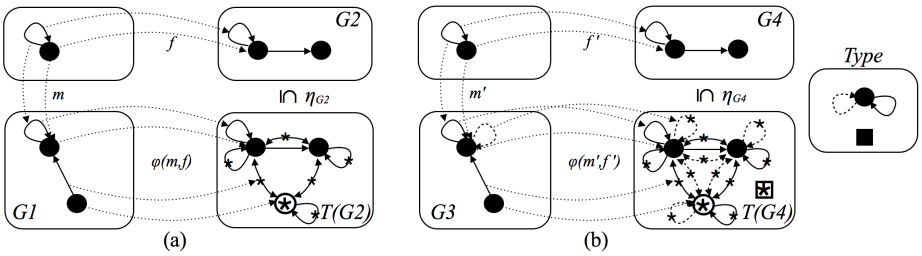


Fig. 2. Partial Map Classifiers (a) in **Gr** (b) in $\mathbf{Gr} \downarrow Type$

In **Gr** the partial map classifier (T, η) is such that $\eta_G : G \to T(G)$ embeds G into the graph $T(G)$ made of the disjoint union of G with a node $*$ and with an edge $*_{n,p} : n \to p$ for each pair of vertices (n, p) in $(N_G + \{*\}) \times (N_G + \{*\})$. The total morphism $\varphi(m, f)$ is defined on the set of nodes exactly as in **Set**, and on each edge similarly, but consistently with the way its source and target nodes are mapped. Figure 2(a) shows an example of a partial map $(m, f) : G1 \to G2$ and the corresponding extension to the total morphism $\varphi(m, f) : G1 \to T(G2)$. In the graphical notation we use edges with double tips to denote two edges, one in each direction; arrows and node marked with $*$ are added to $G2$ by the T construction.

Set and **Gr** are instances of the general result that all elementary toposes have $\mathcal{M}$-partial map classifier, for $\mathcal{M}$ the family of all monos. These include, among others, all *presheaf categories* (i.e., functor categories like $\mathbf{Set}^{\mathbf{C}^{\mathrm{op}}}$, where $\mathbf{C}$ is a small category), and the slice categories like $\mathbf{C} \downarrow X$ where $\mathbf{C}$ is a topos and X an object of $\mathbf{C}$. In fact **Gr** is the presheaf category $\mathbf{Set}^{\mathbf{C}^{\mathrm{op}}}$ where $\mathbf{C}^{\mathrm{op}}$ has two objects E, N and two non-identity arrows $s, t : E \to N$.

As a consequence also the category of typed graphs $\mathbf{Gr} \downarrow Type$ has partial maps classifiers for all monos. Figure 2(b) shows an example: the partial map classifier of a graph $G4$ typed over $Type$ is obtained by adding to $G4$ all the nodes of $Type$ and, for each pair of nodes of the resulting graph, one instance of each edge that is compatible with the type graph.

The category of *polarized graphs* of Def. 2 (that will be used later in Sect. 5.2), is an example of category which has $\mathcal{M}$-partial map classifiers for a family $\mathcal{M}$

which is a proper subset of all monos. It is easy to check that strict monos form a stable system of monos (denoted $\mathcal{S}$) for category $\mathbf{Gr}^{\pm}$, and that $\mathbf{Gr}^{\pm}$ has an $\mathcal{S}$-partial map classifier $(\mathbb{T}, \eta)$. Morphism $\eta_{\mathbb{K}}$ embeds a polarised graph $\mathbb{K}$ into $\mathbb{T}(\mathbb{K})$, which is the disjoint union of $\mathbb{K}$ with a node $*$ and with an edge $*_{n,p} : n \to p$ for each pair of nodes $(n, p) \in (N_K^+ + \{*\}) \times (N_K^- + \{*\})$. The total morphism $\varphi(m, f)$ is defined exactly as in the category of graphs.

3 Algebraic Graph Rewriting with Controlled Embedding

In this section we introduce the AGREE approach to rewriting, defining rules, matches and rewrite steps. The 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 and can be used to control the embedding of preserved items. We assume that $\mathbf{C}$ is a category with all pullbacks, with a stable system of monos $\mathcal{M}$, with an $\mathcal{M}$-partial map classifier (T, η), and with pushouts along monos in $\mathcal{M}$.

Definition 3 (AGREE **rules and matches).**
– *A* rule *is a triple of arrows with the same source* $\rho = (K \xrightarrow{l} L, K \xrightarrow{r} R, K \overset{t}{\rightarrowtail} T_K)$*, with* t *in* $\mathcal{M}$*. Arrows* l *and* r *are the* left- *and* right-hand side*, respectively, and* t *is called the* embedding.

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

– *A* match *of a rule* ρ *with left-hand-side* $K \xrightarrow{l} L$ *is a mono* $L \overset{m}{\rightarrowtail} G$ *in* $\mathcal{M}$.

$$\begin{array}{ccccc} L & \xleftarrow{\ l\ } & K & \xrightarrow{\ r\ } & R \\ \downarrow{\scriptstyle m} & PB\ (remark) & \downarrow{\scriptstyle n} & PO\ (b) & \downarrow{\scriptstyle p} \\ G & \xleftarrow{\ g\ } & D & \xrightarrow{\ h\ } & H \\ \eta_L \big\| & & \downarrow{\scriptstyle n'} & & \big\| t \\ T(L) & \xleftarrow[l'=\varphi(t,l)]{} & T_K & & \end{array} \qquad (4)$$

Definition 4 (AGREE **rewriting).** *Given a rule* $\rho = (K \xrightarrow{l} L, K \xrightarrow{r} R, K \overset{t}{\rightarrowtail} T_K)$ *and a match* $L \overset{m}{\rightarrowtail} G$*, an* AGREE rewrite step $G \Rightarrow_{\rho,m} H$ *is constructed in two phases as follows (see diagram (4)):*
(a) Let $l' = \varphi(t, l) : T_K \to T(L)$ *and* $\overline{m} = \varphi(m, id_L) : G \to T(L)$*, then* $G \xleftarrow{g} D \xrightarrow{n'} T_K$ *is the pullback of* $G \xrightarrow{\overline{m}} T(L) \xleftarrow{l'} T_K$.
(remark) In diagram (4) (g, n') *is a pullback of* $(\overline{m}, l')$ *and* (l, t) *is a pullback of* (η_L, l') *because* $l' = \varphi(t, l)$*, thus by the decomposition property there is a unique* $n : K \to D$ *such that* $n' \circ n = t$*,* $g \circ n = m \circ l$ *and* (l, n) *is a pullback of* (m, g)*. Therefore* n *is a mono in* $\mathcal{M}$ *by stability.*
(b) Let n *be as in the previous remark. Then* $R \xrightarrow{p} H \xleftarrow{h} D$ *is the pushout of* $D \xleftarrow{n} K \xrightarrow{r} R$.

Example 1. Using the AGREE approach, the web page copy operation can be modelled using the rule shown in Figure 3. This rule is typed over the type graph *Type*. Nodes denote web pages, solid edges denote links and dashed edges describe the subpage relation. The different node colours (gray and black) are used just to define the match, whereas the **c** inside some nodes is used to indicate that this is a copy. When this rule is applied to graph $G1$, only out-links are copied because the pages that link the copied one remain the same, that is, they only have a link to the original page, not to its copy. The subpage structure is not copied. Note that all black nodes of $G1$ and $D1$ are mapped to $*$-nodes of $T(L1)$ and $TK1$, respectively.

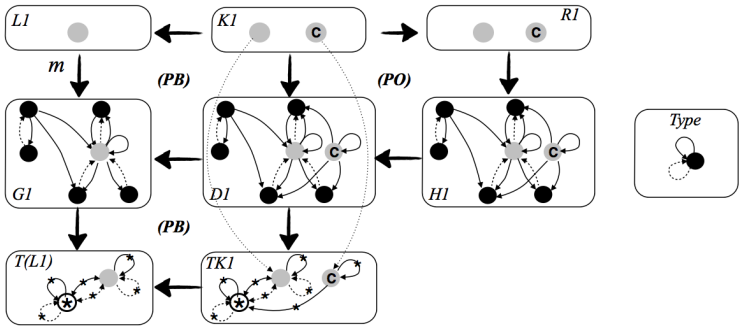
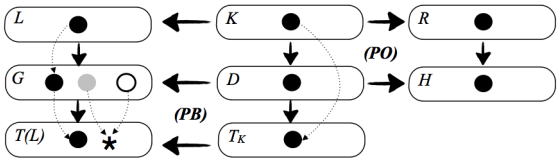


Fig. 3. Rule for copying a web page and example of application

In the general case just presented, the embedding t could have a non-local effect on the rewritten object. In the following example, based on category **Set**, the rule simply preserves a single element and $t : K \to T_K$ is the identity. If applied to set G, its effect is to delete all the elements not matched by m, as shown. We say that this rewrite step is *non-local*, because it modifies the complement of the image of L in G.



In the rest of this section we present a condition on rules that ensures the locality of the rewrite steps. In order to formulate this condition in the general setting of a category with $\mathcal{M}$-partial map classifiers, we need to consider a generalisation of the notion of complement of a subset in a set, that we call *strict complement*. For instance, in category **Gr**, the strict complement of a subgraph L in a graph G is the largest subgraph $G \setminus L$ of G disjoint from L; thus, the union of L and $G \setminus L$ is in general smaller than G. Intuitively, we will say that an AGREE rewrite step as in diagram (4) is *local* if the strict complement of L in G is preserved, i.e., if g restricts to an isomorphism between $D \setminus K$ and $G \setminus L$.

For the definitions and results that follow, we assume that category $\mathbf{C}$, besides satisfying the conditions listed at the beginning of this section, has a final object 1 and a *strict* initial object 0 (i.e., each arrow with target 0 must have 0 as source); furthermore, the unique arrow from 0 to 1, that we denote $! : 0 \to 1$, belongs to $\mathcal{M}$. For each object X of $\mathbf{C}$ we will denote by $1_X : X \to 1$ the unique arrow to the final object, and by $0_X : 0 \to X$ the unique arrow from the initial object.

For each mono $m : L \rightarrowtail G$ in $\mathcal{M}$ the *characteristic arrow* of m is defined as $\chi_m = \varphi(m, 1_L) : G \to T(1)$, (see pullback (a) in diagram (5)). Object $T(1)$ is called the $\mathcal{M}$*-subobject classifier*.

$$\begin{array}{ccccc} K & \xrightarrow{\quad l \quad} & L & \xrightarrow{\quad 1_L \quad} & 1 \\ {\scriptstyle n}\downarrow & PB\ (c) & {\scriptstyle m}\downarrow & PB\ (a) & {\scriptstyle true}\downarrow{\scriptstyle \eta_1} \\ D & \xrightarrow[g]{} & G & \xrightarrow[\chi_m = \varphi(m,1_L)]{} & T(1) \\ {\scriptstyle D\backslash n}\uparrow & PB\ (d) & {\scriptstyle G\backslash m}\uparrow & PB\ (b) & {\scriptstyle false}\uparrow{\scriptstyle T(!)\circ\overline{!}} \\ D \setminus K & \xrightarrow{\quad g\backslash l \quad} & G \setminus L & \xrightarrow{\quad 1_{G\backslash L} \quad} & 1 \end{array} \tag{5}$$

By exploiting the assumption that $! \in \mathcal{M}$ and that 0 is strict initial, it can be shown that $T(0)$ is isomorphic to 1, with $\overline{!} = 1^{-1}_{T(0)}$, and this yields an arrow $T(!) \circ \overline{!} : 1 \to T(1)$. In category **Set** (with $\mathcal{M}$ the family of all injective functions) arrows η_1 and $T(!) \circ \overline{!} : 1 \to T(1)$ are the coproduct injections of the subobject classifier (which is a two element set), and are also known as *true* and *false*, respectively. In **Set** the complement of an injective function $m : L \rightarrowtail G$ can be defined as the pullback of $\chi_m : G \to T(1)$ along *false*. We generalise this to the present setting as follows.

Definition 5 (strict complements). *Let* $\mathbf{C}$ *be a category that satisfies the conditions listed at the beginning of Section 3, has final object* 1*, strict initial object* 0*, and such that* $! \in \mathcal{M}$*. Let* $m : L \rightarrowtail G$ *be a mono in* $\mathcal{M}$*, and* $\chi_m : G \to T(1)$ *be its characteristic arrow defined by pullback (a) of diagram (5). Then the* strict complement of L in G (with respect to m) *is the arrow* $G \setminus m : G \setminus L \rightarrowtail G$ *obtained as the pullback of* χ_m *and* $false = T(!) \circ \overline{!} : 1 \to T(1)$*, as in square (b) of diagram (5).*

Furthermore, for each pair of monos $n : K \rightarrowtail D$ *and* $m : L \rightarrowtail G$ *in* $\mathcal{M}$ *and for each pair of arrows* $l : K \to L$ *and* $g : D \to G$ *such that square (c) of diagram (5) is a pullback, arrow* $g \setminus l : D \setminus K \to G \setminus L$ *as in square (d) is called the* strict complement *of* l *in* g *(with respect to* n *and* m*).*

It is easy to check that arrow $g \setminus l$ exists and is uniquely determined by the fact that square (b) is a pullback; furthermore square (d) is a pullback as well, by decomposition. We will now exploit the notion of strict complement to formalize locality of AGREE rewriting.

Definition 6 (local rules and local rewriting in AGREE**).** *An* AGREE *rule* $\rho = (l, r, t)$ *is* local *if* $\overline{t} : T_K \to T(K)$ *is such that* $\overline{t} \setminus id_K : T_K \setminus K \to$

$T(K) \setminus K$ *is an iso. An* AGREE *rewrite step as in diagram (4) is* local *if arrow* $g \setminus l : D \setminus K \to G \setminus L$ *is an iso.*

The definition of local rewrite steps is as expected, but that of local rules deserves some comments. Essentially, in the first phase of AGREE rewriting, when building the pullback (a) of diagram (4), the shape of $T_K \setminus K$ determines the effect of the rule on the strict complement of L in G, which is mapped by $\overline{m}$ to $T(L) \setminus L$. It can be proved that $T(L) \setminus L$ is isomorphic to $T(K) \setminus K$, therefore if the rule is local we have that $T_K \setminus K$ is isomorphic to $T(L) \setminus L$, and this guarantees that the strict complement of L in G is preserved in the rewrite step. These considerations provide an outline of the proof of the main result of this section, which can be found in [4].

Proposition 1 (locality of AGREE **rewrite steps).** *Let* $\rho = (l, r, t)$ *be a local rule. Then, with the notations as in diagram (4), for each match* $L \overset{m}{\rightarrowtail} G$ *the resulting rewrite step* $G \Rightarrow_{\rho,m} H$ *is local.*

4 Example: 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 set 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 informations are 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 [15] for a discussion about re-identification issues). We are going to show how AGREE rewriting can be used for such anonymization procedure. Of course, due to space limitations we cannot deal with a complete example and will focus in the first task of the anonymization process: the creation of a clone of the social network in which only non-sensitive links are copied. We model the following idealized scenario: the administrator of a social network sells anonymized data sets to third-parties so that they can be analyzed without compromising confidentiality. Our graphs are made of four kinds of nodes: customer (grey nodes), administrator of the social network (white node), user of the social network (black nodes) and square nodes that model the fact that data will suffer post-processing. Links of the social network can be either public (black solid) or private (dashed – this latter denotes sensitive information that should not be disclosed), moreover we use another type of edges (grey), denoting the fact that a node "knows", or has access to another node. The corresponding type graph *Type* is shown in Figure 4.

The rule depicted in Figure 5 shows an example that anonymizes a portion of a social network with 4 nodes (typically portions of a fixed size are sold). Graph

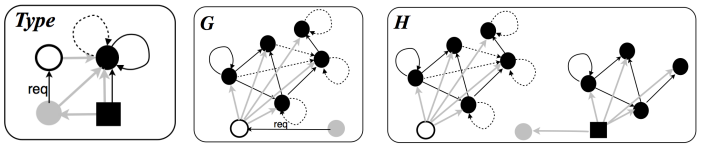


Fig. 4. Type Graph $Type$, Graphs G and H

TK consists of a clique of all copies of matched black nodes (denoted by **c**) with public links, and a graph representing the T construction applied to the rest of K. To enhance readability, we just indicated that the graph inside the dotted square should be completed according to T: a copy of the nodes of the type graph should be added, together with all possible edges that are compatible with the type graph. This allows the cloning of the subgraph defined by the match limited to public edges. In the right hand side R a new square node is added marking the cloned nodes for post-processing. The application of this rule to graph G in Figure 4 with a match not including the top black nodes produces graph H.

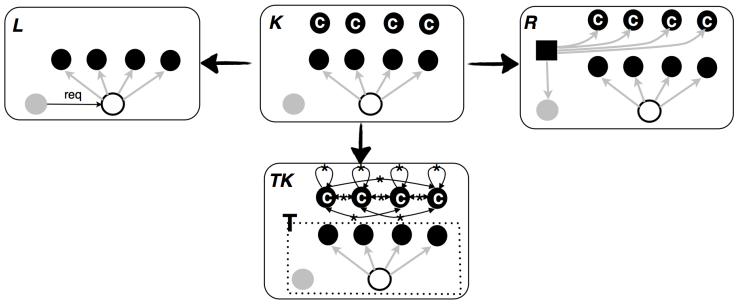


Fig. 5. 4-Anonymize rule

5 AGREE Subsumes SqPO and Polarized Node Cloning

As recalled in the Introduction, in the SqPO approach [5] a rule is a span $L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R$ and a rewriting step for a match $L \stackrel{m}{\rightarrow} G$ is made of a first phase where the *final pullback complement* D is constructed, and next a pushout with the right-hand side is performed.

Definition 7 (final pullback complement). *In diagram (6),* $K \stackrel{n}{\rightarrow} D \stackrel{a}{\rightarrow} G$ *is a* final pullback complement *of* $K \stackrel{l}{\rightarrow} L \stackrel{m}{\rightarrow} G$ *if*

1. *the resulting square is a pullback, and*
2. *for each pullback* $G \stackrel{m}{\leftarrow} L \stackrel{d}{\leftarrow} K' \stackrel{e}{\rightarrow} D' \stackrel{f}{\rightarrow} G$ *and arrow* $K' \stackrel{h}{\rightarrow} K$ *such that* $l \circ h = d$*, there is a unique arrow* $D' \stackrel{g}{\rightarrow} D$ *such that* $a \circ g = f$ *and* $g \circ e = n \circ h$*.*

$$\begin{array}{ccccc} & & \overset{d}{\frown} & & \\ L & \xleftarrow{\ l\ } & K & \xleftarrow{\ h\ } & K' \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle n} & & \downarrow{\scriptstyle e} \\ G & \xleftarrow{\ a\ } & D & \xleftarrow{\ g\ } & D' \\ & & \underset{f}{\smile} & & \end{array} \qquad (6)$$

The next result shows that in a category with a stable system of monos $\mathcal{M}$ and with $\mathcal{M}$-partial map classifiers, the final pullback complement of $m \circ l$, with $m \in \mathcal{M}$, can be obtained by taking the pullback of $T(l)$ along $\overline{m}$. This means that if the embedding morphism of an AGREE rule is the partial map classifier of K, i.e., $K \overset{\eta_K}{\rightarrowtail} T(K)$, then the first phase of the AGREE rewriting algorithm of Definition 4 actually builds the final pullback complement of the left-hand side of the rule and of the match. This will allow us to relate the AGREE approach with others based on the construction of final pullback complements.

Theorem 1 (building final pullback complements). *Let* $\mathbf{C}$ *be a category with pullbacks, with a stable system of monos* $\mathcal{M}$ *and with an* $\mathcal{M}$*-partial map classifier* (T, η)*. Let* $K \overset{l}{\rightarrow} L$ *be an arrow in* $\mathbf{C}$ *and* $L \overset{m}{\rightarrowtail} G$ *be a mono in* $\mathcal{M}$*. Consider the naturality square built over* $K \overset{l}{\rightarrow} L$ *on the left of Figure 6, which is a pullback because* η *is cartesian, and let* $G \overset{a}{\leftarrow} D \overset{n'}{\rightarrow} T(K)$ *be the pullback of* $G \overset{\overline{m}}{\rightarrow} T(L) \overset{T(l)}{\leftarrow} T(K)$*. Then* $K \overset{n}{\rightarrow} D \overset{a}{\rightarrow} G$ *is a final pullback complement of* $K \overset{l}{\rightarrow} L \overset{m}{\rightarrow} G$*, where* $n : K \rightarrowtail D$ *is the only arrow making everything commute.*

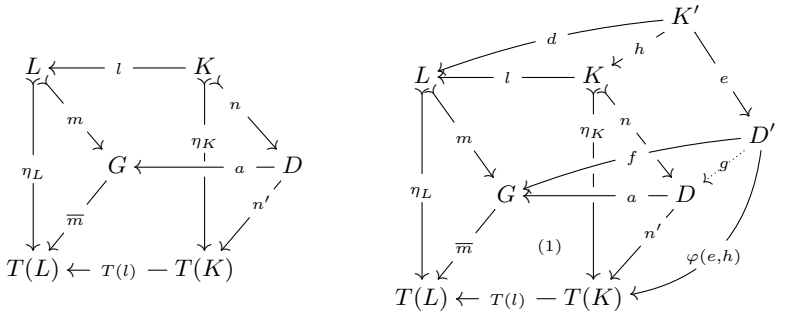


Fig. 6. Constructing the final pullback complement of $m \circ l$ with a pullback.

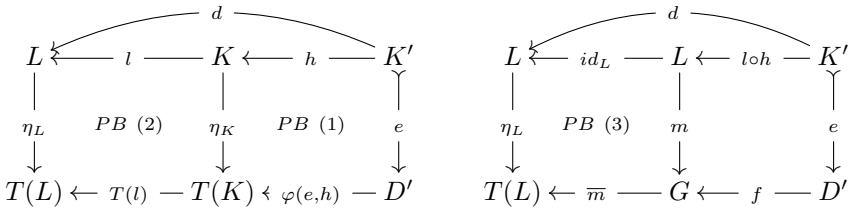
Proof. By the decomposition property we have that $K \overset{n}{\rightarrowtail} D \overset{a}{\rightarrow} G$ is a pullback complement of $K \overset{l}{\rightarrow} L \overset{m}{\rightarrowtail} G$, and $n \in \mathcal{M}$ by stability. We have to show that the pullback complement is final, i.e. that given a pullback $G \overset{m}{\leftarrow} L \overset{d}{\leftarrow} K' \overset{e}{\rightarrow} D' \overset{f}{\rightarrow} G$ and an arrow $K' \overset{h}{\rightarrow} K$ such that $l \circ h = d$, as shown on the right of Figure 6, there is a unique arrow $D' \overset{g}{\rightarrow} D$ such that $n \circ h = g \circ e$ and $a \circ g = f$. We present here the *existence* part, while the proof of *uniqueness* is reported in [4].

Note that $K' \overset{e}{\rightarrowtail} D'$ is in $\mathcal{M}$ by stability. By the properties of the $\mathcal{M}$-partial map classifier T, there is a unique arrow $D' \overset{\varphi(e,h)}{\rightarrow} T(K)$ such that $\eta_K \circ h = \varphi(e,h) \circ e$ and the square is a pullback. We will show below that $\overline{m} \circ f = T(l) \circ \varphi(e,h)$, hence by the universal property of the pullback (1) there is a unique arrow $D' \overset{g}{\rightarrow} D$ such that $n' \circ g = \varphi(e,h)$ and $a \circ g = f$. It remains to show that $n \circ h = g \circ e$: by exploiting again pullback (1), it is sufficient to show

that (i) $a \circ n \circ h = a \circ g \circ e$ and (ii) $n' \circ n \circ h = n' \circ g \circ e$. In fact we have, by simple diagram chasing:

(i) $a \circ n \circ h = m \circ l \circ h = m \circ d = f \circ e = a \circ g \circ e$
(ii) $n' \circ n \circ h = \eta_K \circ h = \varphi(e,h) \circ e = n' \circ g \circ e$

We still have to show that $\overline{m} \circ f = T(l) \circ \varphi(e,h)$. This follows by comparing the following two diagrams, where all squares are pullbacks, either by the statements of Section 2 or (the last to the right) by assumption. Clearly, also the composite squares are pullbacks, but then the bottom arrows must both be equal to $\varphi(e,d)$, as in Equation (1). Therefore we conclude that $\overline{m} \circ f = \varphi(e,d) = T(l) \circ \varphi(e,h)$.



5.1 AGREE subsumes SqPO rewriting with injective matches

Using Theorem 1 it is easy to show that the AGREE approach is a conservative extension of the SqPO approach, because the two coincide if the embedding of the AGREE rule is the arrow injecting K into its partial map classifier.

Theorem 2 (AGREE **subsumes SqPO with monic matches).** *Let* $\mathbf{C}$ *be a category with all pullbacks, with* $\mathcal{M}$*-partial map classifiers* $\eta : Id_{\mathbf{C}} \dot{\to} T$ *for a stable system of monos* $\mathcal{M}$*, and with pushouts along arrows in* $\mathcal{M}$*. Let* $\rho = L \xleftarrow{l} K \xrightarrow{r} R$ *be a rule and* $m : L \rightarrowtail G$ *be a match in* $\mathcal{M}$*. Then*

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

In words, the application of rule ρ *to match* m *using the SqPO approach has exactly the same effect of applying to* m *the same rule enriched with the embedding* $K \overset{\eta_K}{\rightarrowtail} T(K)$ *using the* AGREE *approach.*

Proof. Since the embedding of the rule is arrow $\eta_K : K \rightarrowtail T(K)$, phase (a) of AGREE rewriting (Definition 4) is exactly the construction that is shown, in Theorem 1, to build $K \xrightarrow{n} D \xrightarrow{a} G$ as a final pullback complement of $K \xrightarrow{l} L \xrightarrow{m} G$, therefore it coincides with the construction of the left square of the SqPO approach. The second phase, i.e. the construction of the pushout of $K \xrightarrow{n} D$ and $K \xrightarrow{r} R$ is identical for both approaches by definition.

5.2 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 [9] by using the polarized graphs of Definition 2. Polarization is used in rewriting to control the copies of edges not matched but incident to the matched nodes.

Fact 1 *The* underlying graph *of a polarized graph* $\mathbb{X} = (X, N_X^+, N_X^-)$ *is* X*. This defines a functor* $\mathrm{Depol} : \mathbf{Gr}^{\pm} \to \mathbf{Gr}$ *which has both a right- and a left-adjoint functor denoted* Pol *and* $\mathrm{Pol}^{\pm} : \mathbf{Gr} \to \mathbf{Gr}^{\pm}$*, resp., i.e.* $\mathrm{Pol}^{\pm} \dashv \mathrm{Depol} \dashv \mathrm{Pol}$*.*

Functor Pol *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* $\mathrm{Pol}(f) : \mathrm{Pol}(X) \to \mathrm{Pol}(Y)$ *is a* strict *polarized graph morphism. Furthermore we have that* $\mathrm{Depol} \circ \mathrm{Pol} = Id_{\mathbf{Gr}}$*, and we denote the unit of adjunction* $\mathrm{Depol} \dashv \mathrm{Pol}$ *as* $u : Id_{\mathbf{Gr}^{\pm}} \dot{\to} \mathrm{Pol} \circ \mathrm{Depol}$*, thus* $u_{\mathbb{X}} : \mathbb{X} \to \mathrm{Pol}(\mathrm{Depol}(\mathbb{X}))$*.*

Functor $\mathrm{Pol}^{\pm}$ *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* Depol *has a left adjoint, we have that* Depol *preserves limits and in particular pullbacks.*

The category $\mathbf{Gr}^{\pm}$ has final pullback complements along strict monos: their construction is given in [8, Appendix].

Definition 8 (PSqPO rewriting). *A* PSqPO rewrite rule ρ *is made of a span of graphs* $L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$ *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 mono* $m : L \rightarrowtail G$ *in* $\mathbf{Gr}$*. A* PSqPO rewriting step $G \Rightarrow_{\rho,m}^{PSqPO} H$ *is constructed as follows:*

(a) The left-hand-side l *of the rule* ρ *gives rise to a morphism* $\widehat{l} = \mathrm{Pol}(l) \circ u_{\mathbb{K}} : \mathbb{K} \to \mathrm{Pol}(L)$ *in* $\mathbf{Gr}^{\pm}$*. The match* m *gives rise to a strict mono* $\mathrm{Pol}(m) : \mathrm{Pol}(L) \rightarrowtail \mathrm{Pol}(G)$ *in* $\mathbf{Gr}^{\pm}$*. Then* $\mathbb{K} \overset{n}{\rightarrow} \mathbb{D} \overset{g}{\rightarrow} \mathrm{Pol}(G)$ *is constructed as the final pullback complement of* $\mathbb{K} \overset{\widehat{l}}{\rightarrow} \mathrm{Pol}(L) \overset{\mathrm{Pol}(m)}{\rightarrow} \mathrm{Pol}(G)$ *in category* $\mathbf{Gr}^{\pm}$*.*

(b) Since $\mathrm{Depol}(\mathbb{K}) = K$*, we get* $\mathrm{Depol}(n) : K \to \mathrm{Depol}(\mathbb{D})$ *in* $\mathbf{Gr}$*. Then* $R \overset{p}{\rightarrow} H \overset{h}{\leftarrow} D$ *is built as the pushout of* $R \overset{r}{\leftarrow} K \overset{\mathrm{Depol}(n)}{\rightarrow} \mathrm{Depol}(\mathbb{D})$ *in category* $\mathbf{Gr}$*.*

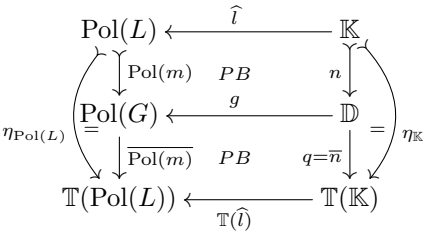
Recall that, as observed in Sect. 2.1, category $\mathbf{Gr}^{\pm}$ has an $\mathcal{S}$-partial map classifier $(\mathbb{T}, \eta)$. This will be exploited in the next result.

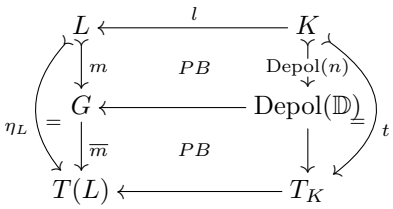
Theorem 3 (AGREE **subsumes polarized node cloning on graphs).** *Let* ρ *be a PSqPO rule made of span* $L \overset{l}{\leftarrow} K \overset{r}{\rightarrow} R$ *and polarized graph* $\mathbb{K} = (K, N_K^+, N_K^-)$*. Consider the component on* $\mathbb{K}$ *of the natural transformation* $\eta : Id_{\mathbf{Gr}^{\pm}} \dot{\to} \mathbb{T}$*, and let* $T_K = \mathrm{Depol}(\mathbb{T}(\mathbb{K}))$ *and* $t = \mathrm{Depol}(\eta_{\mathbb{K}}) : \mathrm{Depol}(\mathbb{K}) \to \mathrm{Depol}(\mathbb{T}(\mathbb{K}))$*, thus* $t : K \to T_K$*. Furthermore, let* $m : L \rightarrowtail G$ *be a mono. Then*

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

Proof. The first phase of PSqPO rewriting consists of building the final pullback complement of $(\mathrm{Pol}(m), \widehat{l})$ in category $\mathbf{Gr}^{\pm}$. According to Theorem 1, since $\mathrm{Pol}(m)$ is strict such final pullback complement can be obtained as the top square in the diagram below to the left, where both squares are pullbacks in $\mathbf{Gr}^{\pm}$. The second phase consists of taking the pushout of morphisms $K \xrightarrow{r} R$ and $\mathrm{Depol}(n) : K \to \mathrm{Depol}(D)$ in $\mathbf{Gr}$.

By applying functor Depol to the left diagram we obtain the diagram below to the right in $\mathbf{Gr}$, where both squares are pullbacks because Depol preserves limits. In fact, recall that $\mathrm{Depol} \circ \mathrm{Pol} = id_{\mathbf{Gr}}$, that $K = \mathrm{Depol}(\mathbb{K})$ and that $t = \mathrm{Depol}(\eta_{\mathbb{K}})$; the fact that $T(L) = \mathrm{Depol}(\mathbb{T}(\mathrm{Pol}(L)))$ can be checked easily by comparing the construction of the ($\mathcal{S}$-)partial map classifiers in $\mathbf{Gr}$ and in $\mathbf{Gr}^{\pm}$.





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 the outer square of the right diagram a pullback. This arrow is precisely $\mathrm{Depol}(\mathbb{T}(\widehat{l}))$, and therefore the pullback is exactly the lower square of the right diagram. The second phase consists of taking the pushout of $K \xrightarrow{r} R$ and of the only arrow $K \to \mathrm{Depol}(D)$ that makes the diagram commute; but $\mathrm{Depol}(n)$ is such an arrow, thus the pushout is the same computed by the PSqPO approach and this concludes the proof.

6 Related Work and Discussion

In this paper we presented the basic definitions of a new approach to algebraic graph rewriting, called AGREE. We showed that this approach subsumes other algebraic approaches like SqPO (Sesqui-pushout) with injective matches (and therefore DPO and SPO under mild restrictions, see [5, Propositions 12 and 14]), as well as its polarised version PSqPO. The main feature provided by this approach is the possibility, in a rule, of specifying which edges shall be copied as a side effect of the copy of a node. This feature offers new facilities to specify applications in which copy of nodes shall be done in an unknown context, and thus it is not possible to describe in the left-hand side of the rule all edges that shall be copied together with the node. As an example, the anonymization of parts of a social network was described in Sect. 4.

The idea of controlling explicitly in the rule how the right-hand side should be 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 [14] 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. The name of our approach is reminiscent of those older ones.

Adaptive star grammars [7] is another framework where node cloning is performed by means of rewrite rules of the form $S ::= R$ where graph S has a shape of a star and R is a graph. Cloning operation, see [7, Definitions 5 and 6], shares the same restrictions as the sesqui-pushout approach: nodes are cloned with all their incident edges.

In [17] a general framework for graph transformations in span-categories, called *contextual graph rewriting*, briefly CR, has been proposed. Using CR, thanks to the notions of rule and of match that are more elaborated than in other approaches, it is possible to specify cloning as in AGREE rewriting, and even more general transformations: e.g., one may create multiple copies of nodes/edges as a side effect, not only when cloning items. The left-hand sides of CR rules allow to specify elements that must exist for the rule to be applicable, called E, and also a context for E, i.e. a part of the graph that will be universally quantified when the rule is applied, called U. A third component plays the role of embedding the context U in the rest of the graph. The rule for copying a web page shown in Figure 3 could be specified using CR as rule $E \rightarrowtail U \rightarrowtail L \leftarrow K \rightarrow R$, where $E = L1, U = L = T(L1)$ and $K = R = TK1$. Finding a match for a rule in a graph G involves finding a smallest subgraph of G that contains E and its complete context. Thus, even if CR is more general, our approach enhances the expressiveness of classical algebraic approaches with a form of controlled cloning using simpler and possibly more natural rules.

Bauderon's pullback approach [1] is also related to our proposal. It was proposed as an algebraic variant of the above mentioned NLC and ed-NLC algorithmic approaches. Bauderon's approach is similar, in part, to the pullback construction used in our first phase of a rewriting step, but a closer analysis is needed and is planned as future work. We also intend to explore if there are relevant applications where AGREE rewriting in its full generality (i.e., with possibly non-local rules) could be useful.

Concerning the applicability of our approach to other structures, in practice the requirement of existence of partial maps classifiers looks quite demanding. AGREE rewriting works in categories of typed/colored graphs, which are used in several applications, because they are slice categories over graphs, and thus toposes. But even more used are the categories of attributed graphs [10], which are not toposes. Under which conditions our approach can be extended or adapted to such structures is an interesting topic that we intend to investigate.

Acknowledgments

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

References

1. Bauderon, M., Jacquet, H.: Pullback as a generic graph rewriting mechanism. Applied Categorical Structures 9(1), 65–82 (2001)
2. Cockett, J., Lack, S.: Restriction categories I: categories of partial maps. Theoretical Computer Science 270(1–2), 223–259 (2002)
3. Cockett, J., Lack, S.: Restriction categories II: partial map classification. Theoretical Computer Science 294(1–2), 61–102 (2003)
4. Corradini, A., Duval, D., Echahed, R., Prost, F., Ribeiro, L.: AGREE - algebraic graph rewriting with controlled embedding. CoRR abs/1411.4597 (2014), `http://arxiv.org/abs/1411.4597`
5. Corradini, A., Heindel, T., Hermann, F., König, B.: Sesqui-pushout rewriting. In: Corradini, A., Ehrig, H., Montanari, U., Ribeiro, L., Rozenberg, G. (eds.) Graph Transformations, ICGT 2006. LNCS, vol. 4178, pp. 30–45. Springer (2006)
6. 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 [19], pp. 163–246
7. Drewes, F., Hoffmann, B., Janssens, D., Minas, M.: Adaptive star grammars and their languages. Theor. Comput. Sci. 411(34-36), 3090–3109 (2010)
8. Duval, D., Echahed, R., Prost, F.: Graph rewriting with polarized cloning. CoRR abs/0911.3786 (2009), `http://arxiv.org/abs/0911.3786`
9. Duval, D., Echahed, R., Prost, F.: Graph transformation with focus on incident edges. In: Ehrig, H., Engels, G., Kreowski, H., Rozenberg, G. (eds.) Graph Transformations, ICGT 2012. LNCS, vol. 7562, pp. 156–171. Springer (2012)
10. Duval, D., Echahed, R., Prost, F., Ribeiro, L.: Transformation of attributed structures with cloning. In: Gnesi, S., Rensink, A. (eds.) Fundamental Approaches to Software Engineering, FASE 2014. LNCS, vol. 8411, pp. 310–324. Springer (2014)
11. Echahed, R.: Inductively sequential term-graph rewrite systems. In: Ehrig, H., Heckel, R., Rozenberg, G., Taentzer, G. (eds.) Graph Transformations, ICGT 2008. LNCS, vol. 5214, pp. 84–98. Springer (2008)
12. 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 [19], pp. 247–312
13. Ehrig, H., Pfender, M., Schneider, H.J.: Graph-grammars: An algebraic approach. In: 14th Annual Symposium on Switching and Automata Theory, Iowa City, Iowa, USA, October 15-17, 1973. pp. 167–180. IEEE Computer Society (1973)
14. Engelfriet, J., Rozenberg, G.: Node replacement graph grammars. In: Rozenberg [19], pp. 1–94
15. 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)
16. Löwe, M.: Algebraic approach to single-pushout graph transformation. Theor. Comput. Sci. 109(1&2), 181–224 (1993)
17. Löwe, M.: Graph rewriting in span-categories. In: Graph Transformations, ICGT 2010. LNCS, vol. 6372, pp. 218–233. Springer (2010)
18. Mitchell, M., Oldham, J., Samuel, A.: Advanced Linux Programming. Landmark Series, New Riders (2001)
19. Rozenberg, G. (ed.): Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations. World Scientific (1997)