Graph rewriting with polarized cloning

Dominique Duval, Rachid Echahed, Frédéric Prost

GRAPH REWRITING WITH POLARIZED CLONING

D. DUVAL, R. ECHAHED, AND F. PROST

D. Duval, UniversitÃ© de Grenoble - Laboratoire LJK, B. P. 53, F-38041 Grenoble, France
e-mail address: dominique.duval@imag.fr

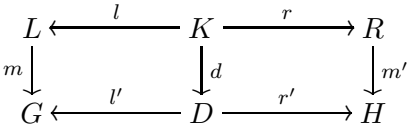
R. Echahed, UniversitÃ© de Grenoble - Laboratoire LIG, B. P. 53, F-38041 Grenoble, France
e-mail address: rachid.echahed@imag.fr

F. Prost, UniversitÃ© de Grenoble - Laboratoire LIG, B. P. 53, F-38041 Grenoble, France
e-mail address: frederic.prost@imag.fr

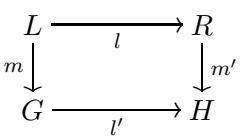
Abstract. We tackle the problem of graph transformation with a particular focus on *node cloning*. We propose a graph rewriting framework where nodes can be cloned zero, one or more times. A node can be cloned together with all its incident edges, with only the outgoing edges, with only the incoming edges or without any of the incident edges. We thus subsume previous works such as the sesqui-pushout, the heterogeneous pushout and the adaptive star grammars approaches. A rule is defined as a span $\mathbb{L} \stackrel{l}{\leftarrow} \mathbb{K} \stackrel{r}{\rightarrow} R$ where the right-hand side R is a multigraph, the left-hand side $\mathbb{L}$ and the interface $\mathbb{K}$ are polarized multigraphs. A polarized multigraph is a multigraph endowed with some cloning annotations on nodes and edges. We introduce the notion of polarized multigraphs and define a rewriting step as pushback followed by a pushout in the same way as in the sesqui-pushout approach.

1. Introduction

Graph transformation [20, 10, 12] extends string rewriting [3] and term rewriting [1] in several respects. In the literature, there are many ways to define graphs and graph rewriting. The proposed approaches can be gathered in two main streams : (i) the algorithmic approaches, which define a graph rewrite step by means of the algorithms involved in the implementation of graph transformation (see e.g. [2, 8]); this stream is out of the scope of the present paper ; (ii) the second stream consists of the algebraic approaches, first proposed in the seminal paper [13], and which uses categorical constructs to define graph transformation in an abstract way. The most popular algebraic approaches are the double pushout (DPO) and the single pushout (SPO) approaches, which can be illustrated as follows :



Double pushout: a rewrite step



Single pushout: a rewrite step

LOGICAL METHODS
IN COMPUTER SCIENCE
DOI:10.2168/LMCS-???

In the DPO approach [13, 5], a rule is defined as a span, i.e., as a pair of graph morphisms $L \leftarrow K \rightarrow R$. A graph G rewrites into a graph H ($G \overset{\text{dpo}}{\Longrightarrow} H$) if and only if there exists a morphism (a matching) $m : L \rightarrow G$, a graph D and graph morphisms d, m', l', r' such that the left and the right squares in the diagram above for a DPO step are pushouts. In general, D is not unique, and sufficient conditions may be given in order to ensure its existence, such as dangling and identification conditions. Since graph morphisms are completely defined, the DPO approach is easy to grasp, but in general this approach fails to specify rules with deletion of nodes, as witnessed by the following example. Let us consider the reduction of the term $f(a)$ by means of the rule $f(x) \rightarrow f(b)$. This rule can be translated into a span $f(x) \leftarrow K \rightarrow f(b)$ for some graph K. When applied to $f(a)$, because of the pushout properties, the constant a must appear in D, hence in H, although $f(b)$ is the only desired result for H.

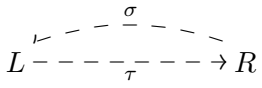
In the SPO approach [19, 14, 15, 11], a rule is a *partial* graph morphism $L \rightarrow R$. When a (total) graph morphism $m : L \rightarrow G$ exists, G rewrites into H ($G \overset{\text{spo}}{\Longrightarrow} H$) if and only if the square in the diagram above for a SPO step is a pushout. This approach is appropriate to specify deletion of nodes thanks to partial morphisms. Deletion of a node causes automatically the deletion of all its incident edges.

In this paper we are interested in graph transformation with an additional feature, namely *cloning* of nodes. Informally, by cloning a node, say n, we mean making zero, one or more copies of the node n ; each copy can be made either with all incident edges of the cloned node n, with only its outgoing edges, with only its incoming edges, or making a clone of n without any of its incident edges.

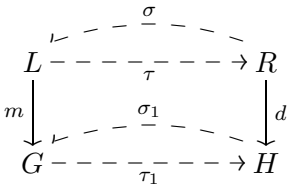
Cloning a substructure is very common in the setting of term rewriting systems. Consider the rule $f(x) \rightarrow g(x, x)$. This rule copies twice the instance of x when applied over first-order terms. In the area of graph transformation, the considered rule can be intuitively written as $f(x) \rightarrow g(p : x, p)$ where x is not copied twice but shared (see e.g., [18, 9]). If this kind of sharing, which is one of the main features of graph transformation, can be of great interest in several areas such as efficient implementations of declarative programming languages [17], cloning of substructures is another important feature which may ease graph transformation in many real-life applications. Unfortunately, this feature has not attracted yet the attention it deserves.

The classical DPO and SPO approaches of graph transformation are clearly not well suited to perform cloning of nodes. As far as we are aware of, there are two algebraic attempts to deal with cloning [4, 7]. In [4], Corradini at al. propose the sesqui-pushout approach where a rewrite rule is a span $L \leftarrow K \rightarrow R$ as in the DPO approach, and a rewrite step $G \overset{\text{sqpo}}{\Longrightarrow} H$ is obtained as in the DPO approach, but the left square is a final pullback complement of the matching m. The sesqui-pushout approach has the ability to clone nodes with all their incident edges.

In this paper, we propose a new algebraic approach to graph transformation with cloning abilities which generalizes the heterogeneous pushout approach (HPO) we have presented in [7]. In the HPO approach, a rewrite rule is defined as a tuple (L, R, τ, σ) such that the left-hand side L and the right-hand side R are termgraphs (a termgraph is a first-order term with possible sharing and cycles), τ is a mapping from the nodes of L to the nodes of R (τ needs not be a graph morphism) and σ is a partial function from the nodes of R to the nodes of L. Roughly speaking, $\tau(p) = n$ indicates that incoming edges of p are to be redirected towards n and $\sigma(n) = p$ indicates that node n should be instantiated as p (parameter passing or cloning). A rewrite rule can be depicted as follows



and a rewrite step in the HPO approach can be illustrated as

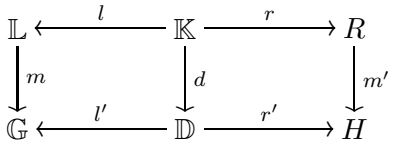


where the graph G is rewritten into H ($G \overset{\text{hpo}}{\Longrightarrow} H$) and H is such that the above diagram is a heterogeneous pushout [7]. In this approach a node may be cloned with all its outgoing edges (and not with all its incident edges as in the sesqui-pushout approach). In the present work, we are generalizing functions τ and σ to spans (relations over the nodes of L and R) and consider multigraph transformation rather than termgraphs. Therefore the rules we are investigating in this paper are of the following shape

$$\mathbb{L} \overset{l}{\leftarrow} \mathbb{K} \overset{r}{\rightarrow} R$$

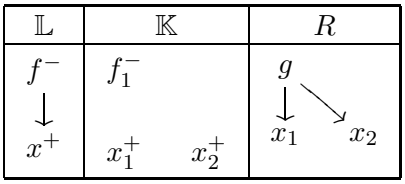
where $\mathbb{L}$ and $\mathbb{K}$ are polarized multigraphs, R is a multigraph, l is a morphism of polarized mutltigraphs and r is a homomorphism of multigraphs. A polarized multigraph $\mathbb{F}$ is a multigraph F such that each node, n, can be annotated or not by the signs $+, -$ such as, n^+, n^-, $n^\pm$ or n. Intuitively, the annotation $+$ in n^+ means that the node n can be cloned together with all its outgoing edges, while n^- means that the node n can be cloned together with all its incoming edges. $n^\pm$ means that the node n can be cloned together with all its incident edges. Finally, the node n without any annotation means that the node n can be cloned without its incident edges. Thus, polarization of nodes provides more flexible ways in cloning nodes if compared to sesqui-pushout and HPO approaches.

A rewrite step in this new approach has the following shape



The right square is a pushout of graphs as in the DPO approach while the left square is a pushback in the category of polarized graphs.

If we consider again the rewrite rule $f(x) \rightarrow g(x, x)$. This rule can be translated to the following span $\mathbb{L} \overset{l}{\leftarrow} \mathbb{K} \overset{r}{\rightarrow} R$ where the morphisms l and r are define as $l(f_1) = f, l(x_1) = x, l(x_2) = x$ and $r(f_1) = g, r(x_1) = x_1, r(x_2) = x_2$. Roughly speaking, the negative (respectively, positive) polarization of f (respectively, of x_1) in $\mathbb{K}$ indicates that all incoming edges of node f should be redirected to point at node g (respectively, all outgoing edges of x, should be copied twice and plugged to x_1 and x_2 in R (see Example 3.5).



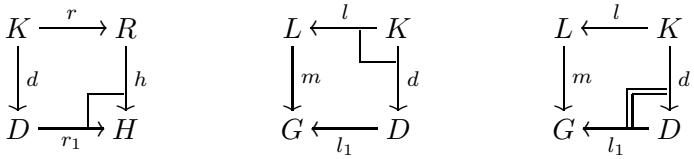
At a first sight, our approach is close to the sesqui-pushout approach [4]. Our work departs from the sesqui-pushout approach in the way the clones can be obtained. We provide different possibilities for cloning thanks to the polarization of nodes. We show later in Section 5 how to simulate the sesqui-pushout approach and emphasize on the abilities which cannot be simulated by the other proposed approaches.

The paper is organized as follows. The next section introduces the notion of polarized graphs together with some properties needed to express graph transformation. Section 3 defines graph rewriting with polarized cloning. Examples illustrating our approach are provided in Section 4. Finally, a comparison with related work and concluding remarks are given in Section 5.

2. Polarized graphs

In this paper, a graph is a directed multigraph. Most results are given up to isomorphism. The propositions are made of explicit constructions, so that their proofs consist "simply" in a verification.

2.1. **Pushouts, pullbacks, pushbacks.** Pushouts and pullbacks are basic notions of category theory [16], mutually dual. A *pushback* is a final pullback complement. They are depicted as follows, respectively:

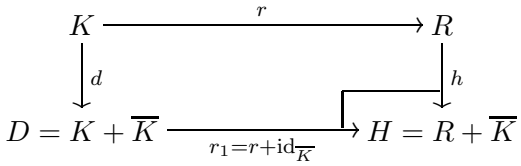


In the first diagram (r_1, h) is the pushout of (d, r) and in the second diagram (l, d) is the pullback of (m, l_1) ("the" pushout and "the" pullback are unique up to isomorphism). Equivalently, in the second diagram (l_1, d) is a pullback complement of (m, l), however pullback complements are not unique up to isomorphism. In the third diagram (l_1, d) is the final pullback complement of (m, l), also called the pushback of (m, l), which is unique up to isomorphism because of its terminality property.

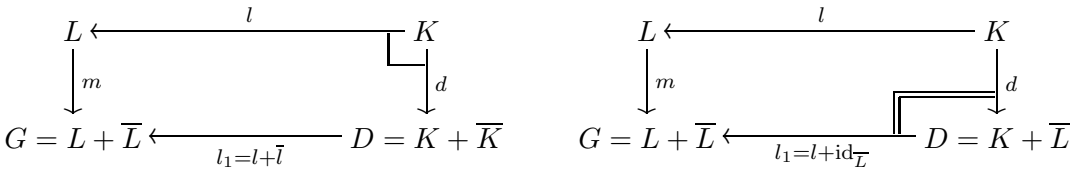
In the category of sets, there are similarities between pushouts and pushbacks. Let **Set** denote the category of sets, let "+" denote the sum (or disjoint union) of sets, and for each set Y with a subset X let $\overline{X}$ denote the complement of X in Y, so that $Y = X + \overline{X}$. The symbol "+" also denotes the sum of functions: when $f_1 : X_1 \to Y_1$ and $f_2 : X_2 \to Y_2$, then $f_1 + f_2 : X_1 + X_2 \to Y_1 + Y_2$ is defined piecewise from f_1 and f_2.

Let $r : K \to R$ be a function and $d : K \to D$ an inclusion, so that $D = K + \overline{K}$. The pushout of d and r in **Set** is the following square, where $h : R \to R + \overline{K}$ is the canonical

inclusion:



Let $l : K \to L$ be a function and $m : L \to G$ an inclusion, so that $G = L + \overline{L}$. The pullback complements of l and m in **Set** are the following squares (on the left), where $d : K \to K + \overline{K}$ is the canonical inclusion and where $\overline{K}$ is any set and $\overline{l} : \overline{K} \to \overline{L}$ any function. The pushback of l and m is the following square (on the right):



Example 2.1. Here is a pullback complement and the pushback of given m and l. Only the sets are represented, the names of their elements describe the functions: m and d are inclusions, while l and l_1 drop the index (every x_i is mapped to x). In this example, for any given $\alpha \in \mathbb{N}$ there is a pullback complement for each $\beta \in \mathbb{N}$ (on the left) and the pushback corresponds to $\beta = 1$ (on the right).

n	$n_1 \dots n_\alpha$
n	$n_1 \dots n_\alpha$
p	$p_1 \dots p_\beta$

n	$n_1 \dots n_\alpha$
n	$n_1 \dots n_\alpha$
p	p_1

2.2. **Graphs.** In this section, some pushouts and pushbacks of graphs are built explicitly.

Definition 2.2. A *graph* X is made of a set of *nodes* $|X|$, a set of *edges* $X_\to$ and two functions *source* and *target* from $X_\to$ to $|X|$. An edge e with source n and target p is denoted $n \xrightarrow{e} p$. The set of edges from n to p in X is denoted $X_{n \to p}$. A *morphism* of graphs $f : X \to Y$ is made of two functions (both denoted f) $f : |X| \to |Y|$ and $f : X_\to \to Y_\to$, such that $f(n) \xrightarrow{f(e)} f(p)$ for each edge $n \xrightarrow{e} p$. This provides the category **Gr** of graphs.

The construction of the sum of two graphs is explicited below, as well as the construction of the *edge-sum* for adding edges to a graph. Given two graphs X_1 and X_2, the *sum* $X_1 + X_2$ is the graph such that $|X_1 + X_2| = |X_1| + |X_2|$ and $(X_1 + X_2)_\to = X_{1\to} + X_{2\to}$ and the source and target functions for $X_1 + X_2$ are induced by the source and target functions for X_1 and for X_2. Given two graphs X and E such that $|E| \subseteq |X|$, the *edge-sum* $X +_e E$ is the graph such that $|X +_e E| = |X|$ and $(X +_e E)_\to = X_\to + E_\to$ and the source and target functions for $X +_e E$ are induced by the source and target functions for X and for E. Clearly, the precise set of nodes of E does not matter in this construction, as long as it contains the source and target of every edge of E and is contained in $|X|$. This notation is extended to morphisms: let $f_1 : X_1 \to Y_1$ and $f_2 : X_2 \to Y_2$, then $f_1 + f_2 : X_1 + X_2 \to Y_1 + Y_2$ is defined piecewise from f_1 and f_2. Similarly, let $f : X \to Y$ and $g : E \to F$ with $|E| \subseteq |X|$ and $|F| \subseteq |Y|$, then $f +_e g : X +_e E \to Y +_e F$ is defined as f on the nodes and piecewise from f and g on the edges.

Remark 2.3. Each graph Y with a subgraph X can be expressed as

$$Y = (X + \overline{X}) +_e \widetilde{X}$$

where $\overline{X}$ is the subgraph of Y generated by the complement of $|X|$ in $|Y|$ and $\widetilde{X}$ is the subgraph of Y generated by the edges in $Y_\rightarrow$ not in $X_\rightarrow + \overline{X}_\rightarrow$.

Definition 2.4. A *matching* of graphs is a monomorphism of graphs (this is often called an "injective matching").

A morphism of graphs is a monomorphism if and only if both underlying functions (on nodes and on edges) are injections. So, up to isomorphism, every monomorphism of graphs can be seen as an inclusion.

Proposition 2.5. *Let $r : K \to R$ be a morphism and $d : K \to D$ a matching of graphs, so that $D = (K + \overline{K}) +_e \widetilde{K}$ as in remark 2.3. Let $r_1 = r + \mathrm{id}_{\overline{K}} : K + \overline{K} \to R + \overline{K}$, let $\widetilde{R}$ be made of one edge $(e, n_K, p_K) : n_H \to p_H$ for each nodes n_H and p_H in $R + \overline{K}$, and each edge $e : n_K \to p_K$ in $\widetilde{K}$ such that $r_1(n_K) = n_H$ and $r_1(p_K) = p_H$, and let $\widetilde{r} : \widetilde{K} \to \widetilde{R}$ map e to (e, n_K, p_K). Then the pushout of d and r in* **Gr** *is the following square, where h is the canonical inclusion.*

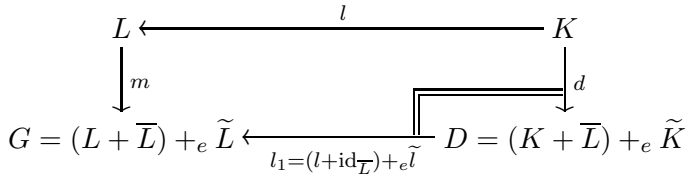
$$\begin{array}{ccc} K & \xrightarrow{\quad r \quad} & R \\ \downarrow{\scriptstyle d} & & \downarrow{\scriptstyle h} \\ D = (K + \overline{K}) +_e \widetilde{K} & \xrightarrow[r_1 = (r + \mathrm{id}_{\overline{K}}) +_e \widetilde{r}]{} & H = (R + \overline{K}) +_e \widetilde{R} \end{array}$$

$$\text{with } \widetilde{R}_{n_H \to p_H} \cong \sum_{n_D \in r_1^{-1}(n_H), p_D \in r_1^{-1}(p_H)} \widetilde{K}_{n_D \to p_D} \text{ for all } n_H, p_H \in |H| \,.$$

Proposition 2.6. *Let $l : K \to L$ be a morphism and $m : L \to G$ a matching of graphs, so that $G = (L + \overline{L}) +_e \widetilde{L}$. The pullback complements of l and m in* **Gr** *are the following squares, where d is the canonical inclusion and where $\overline{K}$ is any graph, $\overline{l} : \overline{K} \to \overline{L}$ is any morphism of graphs, $\widetilde{K}$ is any graph such that $|\widetilde{K}| \subseteq |K| + |\overline{K}|$, $\widetilde{l} : \widetilde{K} \to \widetilde{L}$ is any morphism of graphs which coincide with $l + \overline{l}$ on the nodes.*

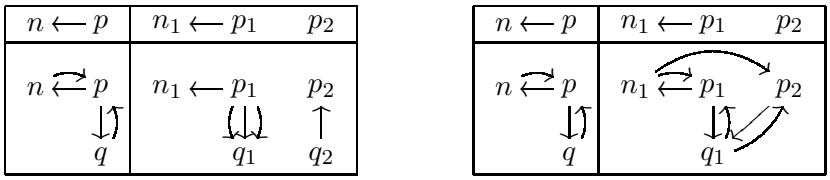
$$\begin{array}{ccc} L & \xleftarrow{\quad l \quad} & K \\ \downarrow{\scriptstyle m} & & \downarrow{\scriptstyle d} \\ G = (L + \overline{L}) +_e \widetilde{L} & \xleftarrow[l_1 = (l + \overline{l}) +_e \widetilde{l}]{} & D = (K + \overline{K}) +_e \widetilde{K} \end{array}$$

Proposition 2.7. *Let $l : K \to L$ be a morphism and $m : L \to G$ a matching of graphs, so that $G = (L + \overline{L}) +_e \widetilde{L}$. The pushback of l and m is the following square, where $\widetilde{K}$ is made of one edge $(e, n, p) : n \to p$ for each $e : l_1(n) \to l_1(p)$ in $\widetilde{L}$ and $\widetilde{l}$ maps (e, n, p) to e.*



with $\widetilde{K}_{n_D \to p_D} \cong \widetilde{L}_{l_1(n_D) \to l_1(p_D)}$ for all $n_D, p_D \in |D|$.

Example 2.8. With the same conventions as in exemple 2.1, here is a pullback complement (on the left) and the pushback (on the right) of given m and l in **Gr**.



2.3. **Polarized graphs.** In this paper, a polarized graph is a graph with two distinguished subsets of nodes and a distinguished subset of edges. In our rewriting process, the polarization will be used for cloning. In this section the category of polarized graphs is defined and it is proved that it has the required pushbacks, by building them explicitly.

Definition 2.9. A *polarization* X° of a graph X is a triple $(|X|^+, |X|^-, X^\star_\rightarrow)$ such that $X^\circ \subseteq X$, in the sense that $|X|^+ \subseteq |X|$, $|X|^- \subseteq |X|$ and $X^\star_\rightarrow \subseteq X_\rightarrow$, such that each $n \xrightarrow{e} p$ in $X^\star_\rightarrow$ has its source $n \in |X|^+$ and its target $p \in |X|^-$. A *polarized graph* $\mathbb{X} = (X, X^\circ)$ is a graph X together with a polarization X° of X. A *morphism* of polarized graphs $f : \mathbb{X} \to \mathbb{Y}$, where $\mathbb{X} = (X, X^\circ)$ and $\mathbb{Y} = (Y, Y^\circ)$, is a morphism of graphs $f : X \to Y$ such that $f(X^\circ) \subseteq Y^\circ$, in the sense that $f(|X|^+) \subseteq |Y|^+$, $f(|X|^-) \subseteq |Y|^-$ and $f(X^\star_\rightarrow) \subseteq Y^\star_\rightarrow$. This provides the category $\mathbf{Gr}^\circ$ of polarized graphs.

Given two polarized graphs $\mathbb{X}_1$ and $\mathbb{X}_2$, their *sum* is the polarized graph $\mathbb{X}_1 + \mathbb{X}_2$ made of the graph $X_1 + X_2$ with the polarization $|X_1 + X_2|^+ = |X_1|^+ + |X_2|^+$, $|X_1 + X_2|^- = |X_1|^- + |X_2|^-$ and $(X_1 + X_2)^\star_\rightarrow = X_1{}^\star_\rightarrow + X_2{}^\star_\rightarrow$. Given two polarized graphs $\mathbb{X}$ and $\mathbb{E}$ such that $|E| \subseteq |X|$, $|E|^+ \subseteq |X|^+$ and $|E|^- \subseteq |X|^-$, their *edge-sum* is the polarized graph $\mathbb{X} +_e \mathbb{E}$ made of the graph $X +_e E$ with the polarization $|X +_e E|^+ = |X|^+$, $|X +_e E|^- = |X|^-$ and $(X +_e E)^\star_\rightarrow = X^\star_\rightarrow + E^\star_\rightarrow$. This notation is extended to morphisms.

As in definition 2.9, we use the same notations for a polarization as for a subgraph, with obvious meaning: each notation involving a polarization X° is a shorthand for a triple of notations involving respectively $|X|^+$, $|X|^-$ and $X^\star_\rightarrow$.

Definition 2.10. A *matching* of polarized graphs is a monomorphism $f : \mathbb{X} \to \mathbb{Y}$ such that $f(X^\circ) = f(X) \cap Y^\circ$ (f strictly preserves the polarization).

A morphism of polarized graphs is a monomorphism if and only if, as a morphism of graphs, it is a monomorphism. So, up to isomorphism, it can be seen as an inclusion.

Remark 2.11. Let $\mathbb{X} \subseteq \mathbb{Y}$ be a matching of polarized graphs. Let $\mathbb{X} = (X, X^\circ)$ and $\mathbb{Y} = (Y, Y^\circ)$, so that $X^\circ = X \cap Y^\circ$. Let $Y = (X + \overline{X}) +_e \widetilde{X}$ as in remark 2.3. Let $\overline{X}^\circ = \overline{X} \cap Y^\circ$ and $\overline{\mathbb{X}} = (\overline{X}, \overline{X}^\circ)$. Let $\widetilde{X}^\circ = \widetilde{X} \cap Y^\circ$ and $\widetilde{\mathbb{X}} = (\widetilde{X}, \widetilde{X}^\circ)$. Then $\mathbb{X}$, $\overline{\mathbb{X}}$ and

$\widetilde{\mathbb{X}}$ are polarized graphs included in $\mathbb{Y}$. In addition, since the inclusion is a matching, the subpolarization $(X^\circ + \overline{X}^\circ) +_e \widetilde{X}^\circ$ of Y° is equal to Y°. It follows that:

$$\mathbb{Y} = (\mathbb{X} + \overline{\mathbb{X}}) +_e \widetilde{\mathbb{X}} \,.$$

Proposition 2.12. *Let $l : \mathbb{K} \to \mathbb{L}$ be a morphism and $m : \mathbb{L} \to \mathbb{G}$ a matching of polarized graphs, so that $\mathbb{G} = (\mathbb{L} + \overline{\mathbb{L}}) +_e \widetilde{\mathbb{L}}$ as in remark 2.11. The pullback complements of l and m in $\mathbf{Gr}^\circ$ are the following squares, where d is the canonical inclusion and where the polarized graph $\overline{\mathbb{K}}$ and the morphism $\overline{l} : \overline{\mathbb{K}} \to \overline{\mathbb{L}}$ are arbitrary, $\widetilde{\mathbb{K}}$ is any polarized graph such that $|\widetilde{\mathbb{K}}| \subseteq |\mathbb{K} + \overline{\mathbb{K}}|$ as polarized graphs, and $\widetilde{l} : \widetilde{\mathbb{K}} \to \widetilde{\mathbb{L}}$ is any morphism of polarized graphs which coincides with $l + \overline{l}$ on nodes.*

$$\begin{array}{ccc} \mathbb{L} & \xleftarrow{\quad l \quad} & \mathbb{K} \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle d} \\ \mathbb{G} = (\mathbb{L} + \overline{\mathbb{L}}) +_e \widetilde{\mathbb{L}} & \xleftarrow[l_1 = (l+\overline{l}) +_e \widetilde{l}]{} & \mathbb{D} = (\mathbb{K} + \overline{\mathbb{K}}) +_e \widetilde{\mathbb{K}} \end{array}$$

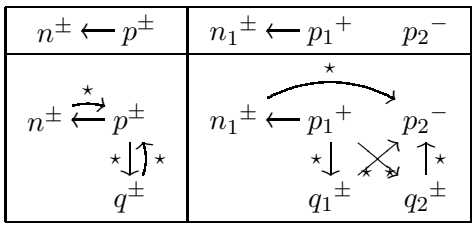
Proposition 2.13. *Let $l : \mathbb{K} \to \mathbb{L}$ be a morphism and $m : \mathbb{L} \to \mathbb{G}$ a matching of polarized graphs, so that $\mathbb{G} = (\mathbb{L} + \overline{\mathbb{L}}) +_e \widetilde{\mathbb{L}}$ as in remark 2.11. In addition, let us assume that $l(K^\star_\to) = l(K_\to) \cap L^\star_\to$ (l strictly preserves the polarization of edges) and that $\widetilde{\mathbb{L}}^\star_\to = \widetilde{\mathbb{L}}_\to$ (every edge in $\widetilde{\mathbb{L}}$ is polarized). Then the pushback of l and m is the following square, where $\widetilde{\mathbb{K}}$ and $\widetilde{l} : \widetilde{\mathbb{K}} \to \widetilde{\mathbb{L}}$ are defined as follows, using $l_1 = l + \mathrm{id}_{\overline{\mathbb{L}}}$ on nodes: $\widetilde{K}^\star_\to$ is made of one edge $(e, n_D, p_D) : n \to p$ for each $n_D \in |K|^+ + |\overline{L}|^+$, $p_D \in |K|^- + |\overline{L}|^-$ and $e : l_1(n_D) \to l_1(p_D)$ in $\widetilde{L}^\star_\to$, $\widetilde{K}_\to = \widetilde{K}^\star_\to$, and $\widetilde{l}$ maps (e, n_D, p_D) to e.*

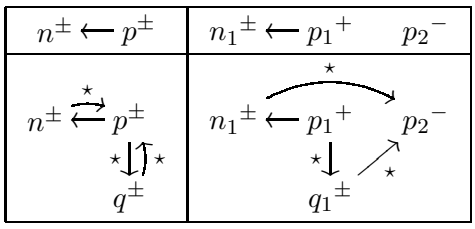
$$\begin{array}{ccc} \mathbb{L} & \xleftarrow{\quad l \quad} & \mathbb{K} \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle d} \\ \mathbb{G} = (\mathbb{L} + \overline{\mathbb{L}}) +_e \widetilde{\mathbb{L}} & \xleftarrow[l_1 = (l+\mathrm{id}_{\overline{\mathbb{L}}}) +_e \widetilde{l}]{} & \mathbb{D} = (\mathbb{K} + \overline{\mathbb{L}}) +_e \widetilde{\mathbb{K}} \end{array}$$

with $\widetilde{K}_{n_D \to p_D} = \widetilde{K}^\star_{n_D \to p_D} \cong \widetilde{L}^\star_{l_1(n_D) \to l_1(p_D)}$ for all $n_D \in |D|^+$, $p_D \in |D|^-$

Remark 2.14. This result means that the polarized part of $\mathbb{G}$ is cloned in $\mathbb{D}$, according to the cloning instructions provided by the polarized part of $\mathbb{K}$.

Example 2.15. Here is a pullback complement (on the left) and the pushback (on the right) of given m and l in $\mathbf{Gr}^\circ$. A node $x \in |X|^+$ is denoted x^+, symmetrically $x \in |X|^-$ is denoted x^-, and an edge $e : x \to y$ in $X^\star_\to$ is denoted $e : x \overset{\star}{\to} y$.



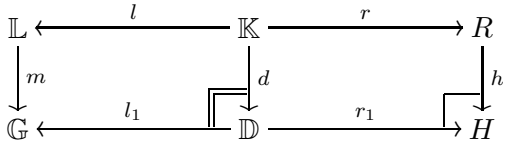


3. Graph rewriting with polarized cloning

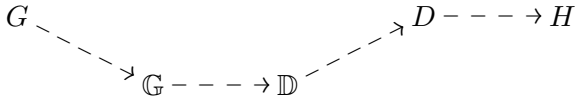
Definition 3.1. A *rewrite rule*, or *production*, is made of a morphism of polarized graphs $l : \mathbb{K} \to \mathbb{L}$ such that $l(K^\star_\rightarrow) = l(K_\rightarrow) \cap L^\star_\rightarrow$ (l strictly preserves the polarization of edges) and a morphism of graphs $r : K \to R$ where K is the graph underlying $\mathbb{K}$. This is denoted $\mathbb{L} \overset{l}{\leftarrow} \mathbb{K} \overset{r}{\rightarrow} R$.

As in definitions 2.4 and 2.10, a *matching* of graphs is a monomorphism $m : X \to Y$ in **Gr** and a *matching* of polarized graphs is a monomorphism $m : \mathbb{X} \to \mathbb{Y}$ in $\mathbf{Gr}^\circ$ such that $f(X^\circ) = f(X) \cap Y^\circ$. Now, let us consider a rule $p = \mathbb{L} \overset{l}{\leftarrow} \mathbb{K} \overset{r}{\rightarrow} R$ and a matching of graphs $m : L \to G$.

Roughly speaking, in graph rewriting with polarized cloning, the aim of a rewrite step with respect to p and m is to build a matching of graphs $h : R \to H$, by replacing L in G by R and taking into account the cloning instructions provided by $\mathbb{K}$. More precisely, a rewrite step with respect to p and m builds a rule $p_1 = \mathbb{G} \overset{l_1}{\leftarrow} \mathbb{D} \overset{r_1}{\rightarrow} H$, a matching of polarized graphs $d : \mathbb{K} \to \mathbb{D}$ and a matching of graphs $h : R \to H$, in such a way that the square of polarized graphs on the left-hand side takes care of the cloning, thanks to a pushback, and the square of graphs on the right-hand side takes care of the "pure" rewriting, thanks to a pushout.



Graph rewriting with polarized cloning is made of four consecutive substeps, their succession can be illustrated as follows; the top line is about graphs, and the bottom line about polarized graphs. The first and third parts are quite easy, the second and fourth parts are made of a pushback of polarized graphs and a pushout of graphs, respectively: we recognize the main features of sesqui-pushout rewriting [4].



To begin with, the matching $m : L \to G$ and the polarization L° of L give rise to a polarization G° of G.

Definition 3.2. Let $\mathbb{L}$ be a polarized graph, $m : L \to G$ a matching of graphs, and let $G = (L+\overline{L})+_e \widetilde{L}$ as in remark 2.3. The *maximal polarization* of G preserving the polarization L° of L (with respect to m) is $G^\circ = (L^\circ + \overline{L}) +_e \widetilde{L}$.

This means that G° coincides with L° on the image of m and that it is maximal outside the image of m. The resulting polarized graph is $\mathbb{G} = (G, G^\circ) = (\mathbb{L} + \overline{\mathbb{L}}) +_e \widetilde{\mathbb{L}}$ where $\overline{\mathbb{L}} = (\overline{L}, \overline{L})$ and $\widetilde{\mathbb{L}} = (\widetilde{L}, \widetilde{L})$, hence every edge in $\widetilde{\mathbb{L}}$ is polarized. Then m defines a matching of polarized graphs $m : \mathbb{L} \to \mathbb{G}$.

Definition 3.3. Given a rule $p = \mathbb{L} \overset{l}{\leftarrow} \mathbb{K} \overset{r}{\rightarrow} R$ and a matching of graphs $m : L \to G$, the *rewrite step* with respect to p and m is the succession of the following substeps.

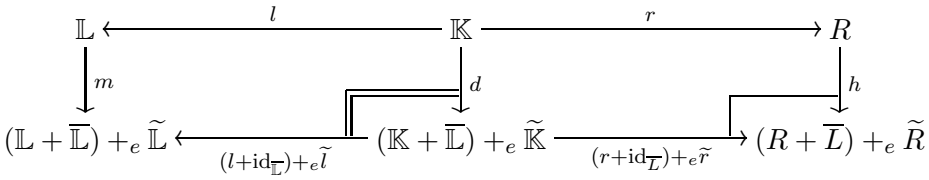
- **From G to $\mathbb{G}$.** The polarized graph $\mathbb{G}$ is (G, G°) where G° is the maximal polarization of G preserving the polarization L° of L.

- **From $\mathbb{G}$ to $\mathbb{D}$.** The pushback of m and l in $\mathbf{Gr}^\circ$ (proposition 2.13) gives rise to a polarized graph $\mathbb{D}$, a morphism $l_1 : \mathbb{D} \to \mathbb{G}$ and a matching $d : \mathbb{K} \to \mathbb{D}$ in $\mathbf{Gr}^\circ$.
- **From $\mathbb{D}$ to D.** The graph D and the matching $d : K \to D$ are obtained simply by forgetting the polarizations.
- **From D to H.** The pushout of m and l in $\mathbf{Gr}$ (proposition 2.5) gives rise to a graph H, a morphism $r_1 : D \to H$ and a matching $h : R \to H$.

This is written $G \overset{pc}{\Longrightarrow} H$.

Theorem 3.4. With respect to a rule $\mathbb{L} \overset{l}{\leftarrow} \mathbb{K} \overset{r}{\rightarrow} R$ and a matching $m : L \to G$, the graph G rewrites into H in the sense of polarized cloning rewriting ($G \overset{pc}{\Longrightarrow} H$) if and only if there is a matching $h : R \to H$ such that, with the notations $G = (L + \overline{L}) +_e \widetilde{L}$ (with respect to m) and $H = (R + \overline{R}) +_e \widetilde{R}$ (with respect to h) as in remark 2.3,

- $\overline{R} = \overline{L}$
- and for every pair of nodes $n_H, p_H \in |H|$ (where $|H| = |R| + |\overline{L}|$), if $n_H \in |H|^+$ and $p_H \in |H|^-$ then $R_{n_H \to p_H}$ is the disjoint union of the sets $L^\star_{l_1(n_K) \to l_1(p_K)}$ for every $n_K \in |K|^+$ such that $r_1(n_K) = n_H$ and every $p_K \in |K|^-$ such that $r_1(p_K) = p_H$, otherwise $R_{n_H \to p_H}$ is empty.

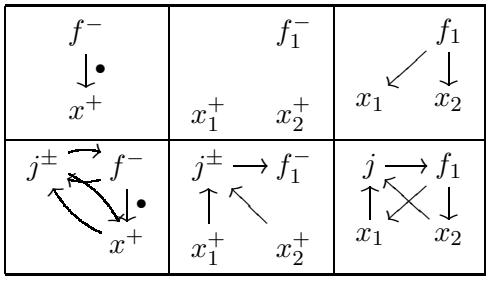


Proof. The proof follows the construction of the polarized cloning rewrite step.

- By definition 3.2, $\mathbb{G} = (\mathbb{L} + \overline{\mathbb{L}}) +_e \widetilde{\mathbb{L}}$ where $\overline{\mathbb{L}} = (\overline{L}, \overline{L})$ and $\widetilde{\mathbb{L}} = (\widetilde{L}, \widetilde{L})$.
- Then by proposition 2.13, $\mathbb{D} = (\mathbb{K} + \overline{\mathbb{L}}) +_e \widetilde{\mathbb{K}}$ and $l_1 = (l + \mathrm{id}_{\overline{\mathbb{L}}}) +_e \widetilde{l}$ where $\widetilde{K}_{n_D \to p_D} \cong \widetilde{L}_{l_1(n_D) \to l_1(p_D)}$ for all $n_D, p_D \in |D|$ and $\widetilde{K}^\star = \widetilde{K}$, and $\widetilde{l}(e, n_D, p_D) = e$ for every $(e, n_D, p_D) \in \widetilde{K}_\to$.
- It follows that $D = (K + \overline{L}) +_e \widetilde{K}$
- Then by proposition 2.5, $H = (R + \overline{L}) +_e \widetilde{R}$ and $r_1 = (r + \mathrm{id}_{\overline{K}}) +_e \widetilde{r}$ where $\widetilde{R}_{n_H \to p_H} \cong \sum_{n_D \in r_1^{-1}(n_H), p_D \in r_1^{-1}(p_H)} \widetilde{K}_{n_D \to p_D}$ for all $n_H, p_H \in |H|$ and $\widetilde{r}(e) = (e, n_D, p_D)$ for every $e : n_D \to p_D \in \widetilde{K}_\to$.

□

Example 3.5. Here is a rewrite step that implements the basic cloning rule $f(x) \to f_1(x, x)$ discussed in the introduction. On the left-hand side we use the same convention on node names as in previous examples: $n_i \mapsto n$. On the right-hand side in this simple example the names are preserved: $n_i \mapsto n_i$. Instead of annotating most edges with $\star$, we use the symbol $\bullet$ for annotating the other edges: in other words, $\bullet$ is the negation of $\star$.

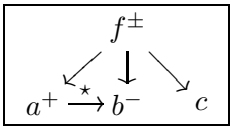


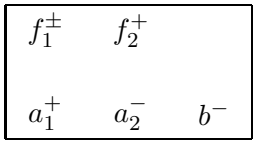
Since the clones x_1 and x_2 of x are annotated by $+$, x_1 and x_2 inherit solely the outgoing edges of x. As for f_1 which is annotated by $-$, it inherits only the incoming edges of f.

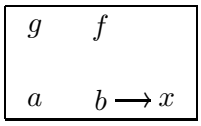
4. Examples

In this section we illustrate through some examples the proposed graph transformation with polarized cloning.

Example 4.1. Let us consider the production $p = \mathbb{L} \xleftarrow{l} \mathbb{K} \xrightarrow{r} R$ where

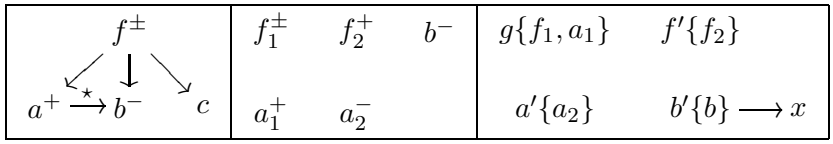
$\mathbb{L} =$ 

$\mathbb{K} =$ 

$\mathbb{R} =$ 

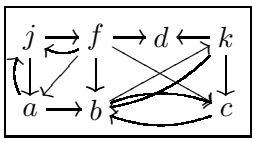
and l is defined by $\{f_1 \mapsto f, f_2 \mapsto f, b \mapsto b, a_1 \mapsto a, a_2 \mapsto a\}$ and r is defined by $\{f_1 \mapsto g, f_2 \mapsto f, b \mapsto b, a_1 \mapsto g, a_2 \mapsto a\}$

The production p is graphically represented as follows:

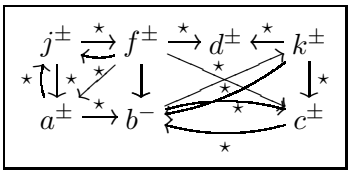


We use name sharing in order to represent morphisms, i.e. $r(f_i) = f = l(f_i)$. We do not put annotations on nodes of $\mathbb{L}$. We assume in the rest of this section that the annotations of the nodes of $\mathbb{L}$ are defined from those of the nodes of $\mathbb{K}$ and the morphism l as follows : a node n in $\mathbb{L}$ inherits the annotations of its l antecedents. In R we follow the convention that $n\{e_1, ..., e_n\}$ is a node n such that e_i is in $\mathbb{K}$, and $r(e_i) = n$. Therefore, a node without a list of antecedents in R is not an image of a node in $\mathbb{K}$ and has to be considered as a new node.

Now let us consider the following graph G:

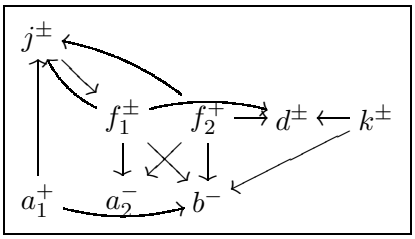


with matching m such that $m(x) = x$ for $x \in \{f, a, b, c\}$. In the following we use this name sharing convention between L and G in order to represent the matching. Then $\mathbb{G}$ is:

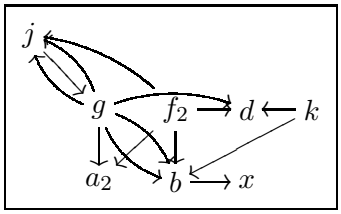


The idea is that each node in $|G| - |m(L)|$ is polarized $\pm$, and each node in $|m(L)|$ is polarized as in $\mathbb{L}$. The same method is followed to annotate edges. Thus, in $\mathbb{G}$ all edges are annotated with $\star$ except the ones that are matched and not annotated in L. In our example only edges $f \to a, f \to b, f \to c$ are not annotated. Notice that the edge $a \to b$ is annotated even if it is matched since it is annotated in $\mathbb{L}$.

For $\mathbb{D}$ we use the same naming convention in order to represent morphism h between $\mathbb{K}$ and $\mathbb{D}$ as the one used to represent morphisms r and l. Thus we have the following graphic representation in which we omitted annotations on edges for the sake of readability (all edges should be annotated with $\star$):

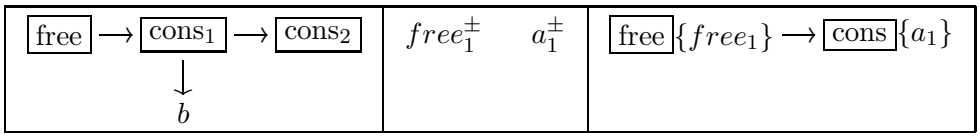


and finally R is the graph:

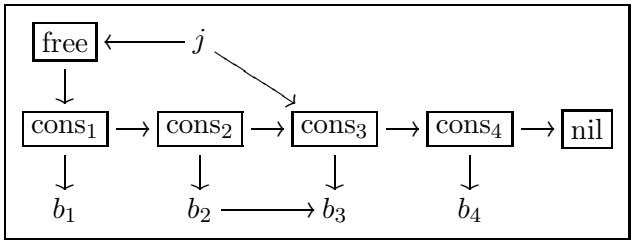


Example 4.2 (Memory freeing)**.** We consider unlabeled graphs, but for the sake of readability we will informally write $\boxed{\text{cons}}$ for a node of name *cons*. Such a node is supposed to match nodes written $\boxed{\text{cons}_i}$.

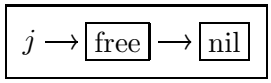
Following this convention, we can write the rule that frees the memory used by a list as follows:



One can check that if we match $\boxed{\text{cons}_1}$ to $\boxed{\text{cons}_i}$s, a to $\boxed{\text{cons}_{i+1}}$s (and a to $\boxed{\text{nil}}$ for $i = 4$) and b to b_is, then successive applications of the previous rewriting rule starting from the following graph G:



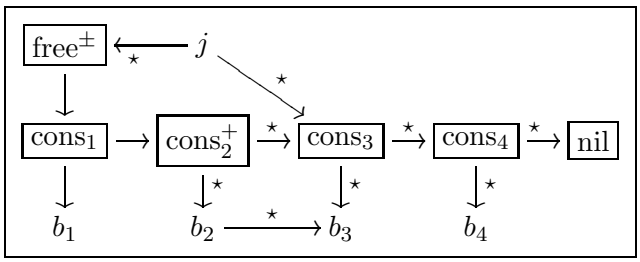
lead to the graph



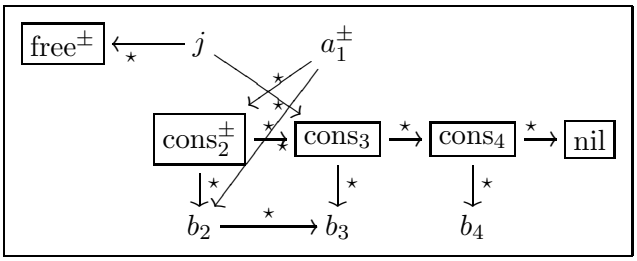
Notice that edges adjacent to a deleted nodes are deleted as well (namely in this example edges $j \to b_2$ and $b_2 \to b_3$).

The first rewrite step reducing G is defined as follows.

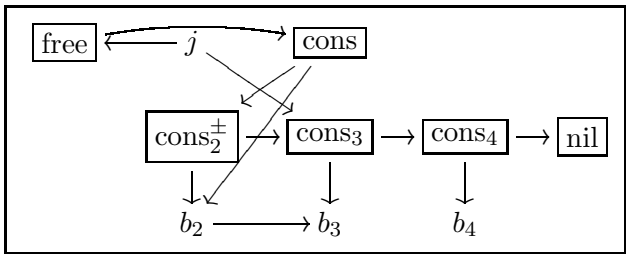
The annotated graph $\mathbb{G}$ is :



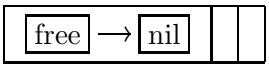
$\mathbb{D}$ is :



and finally R is:

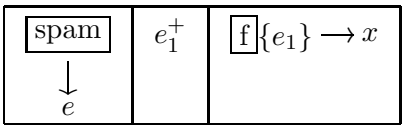


The halt case can be treated by the following simple rule

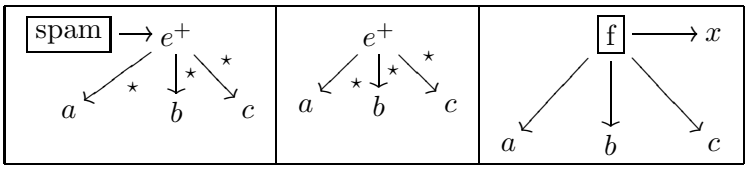


in which $\mathbb{K}$ and R are empty graphs.

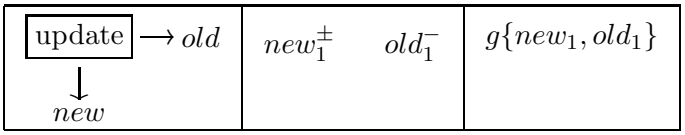
Example 4.3. Now we present a rule scheme that illustrates a use of the cloning of outgoing edges. Function spam applies function f to all nodes pointed by the argument of spam.



This rule can produce the following rewrite step:

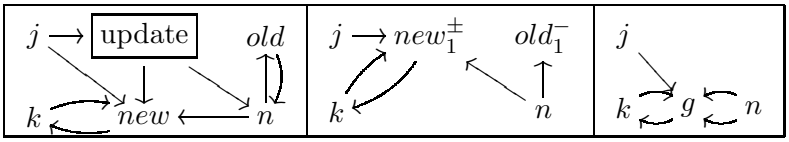


Example 4.4 (Global update)**.** Let us consider now the use of incoming edges cloning. This feature is useful when one wants to update a shared information. Consider the following rule:



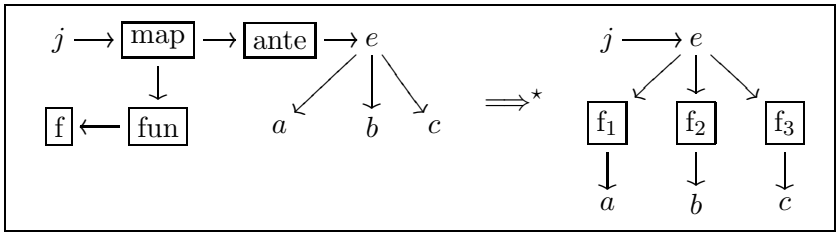
In $\mathbb{K}$, $new^{\pm}$ is used to make a clone of the node *new* (this is a clone of both incoming and outgoing edges) and old^{-} is used to collect all incoming edges of the node *old*. Then, using r all those incoming edges are redirected to the clone of *new*.

This rule produces the follwing rewrite step:



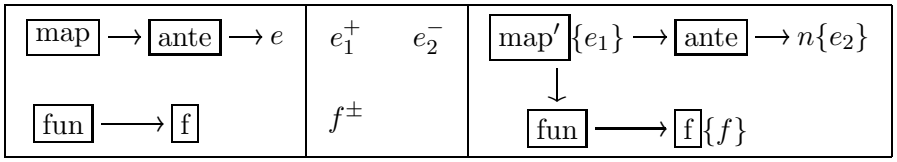
Thus *old* is collapsed to *new* in g, and edges pointing to *old*, namely $n \to old$ are redirected to point g.

Example 4.5. Being able to clone separately incoming and outgoing edges makes possible the writing of rules that behave like the higher order function *map*, which applies a given function to each descendant of a node (instead of a list as in the usual map function). We want to have the following rewrite derivation:

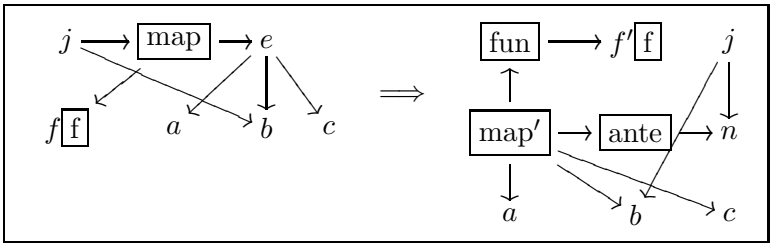


Nodes *ante* and *fun* are used in order to avoid any matching confusion since in our graphs, descendant of a node are not ordered . *ante* indicates the antecedent node (e in our example) to which the result is going to be linked. *fun* specifies which function has to be copied for each descendant of the node pointed by the one specified by *ante*.

It is done by the combination of two rewrite rules and the introduction of an intermediate function map′. The idea is to clone all outgoing edges from e and for each of these edges to make a copy of function f that is introduced between e and its successors. The initialization of map′ is done by the following rule:



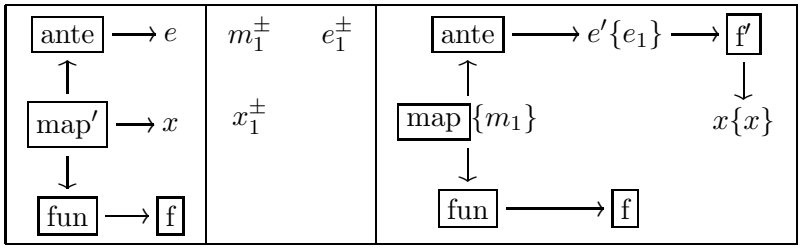
Using this rule, we get the following rewrite step



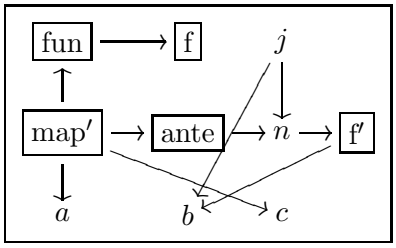
Let us call G_1 the right-hand side of this rewrite step.

This first step uses two clones of e, one is used to collect all the nodes to which f is applied (namely e_1^+) and the other one is going to be used to gather the successive applications of f.

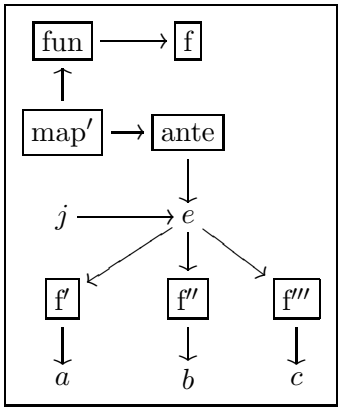
We now give the rule for map′. Here the idea is to make a copy of the function f pointed by *fun* and to introduce an edge from this new copy to the third argument of map' and from the node pointed by *ante* to the copy of the function f. Thus this rule is going to apply the function f to all descendants of map' that are not labeled by ante nor fun.



G_1 matched by the left-hand side above through the matching $\{e \mapsto n, x \mapsto b, ante \mapsto ante, fun \mapsto fun, f \mapsto f'\}$ rewrites to the following graph G_2:



Now, the successive applications of this rule lead to the graph:



The subgraph reachable from node map′ is no longer reachable from the rest of the graph and has to be deleted by garbage collection.

5. Conclusion and Related Work

We proposed a new way to perfom graph transformation which offers different possiblities to clone nodes and their incident edges, in addition to classical graph transformation (addition and deletion of nodes and edges). This work has been obtained as a generalization of [7] where an algebraic approach, based on heterogeneous pushouts (HPO), of termgraph transformation has been proposed. There, a rule is defined as a tuple (L, R, τ, σ) such that

L and R are termgraphs representing the left-hand and the right-hand sides of the rule, τ is a mapping from the nodes of L to those of R and σ is a partial function from nodes of R to nodes of L. The mapping τ describes how incoming edges of the nodes in L are connected in R (i.e., global redirection of incoming pointers), τ is not required to be a graph morphism as in classical algebraic approaches of graph transformation. The role of σ is to indicate the parts of L to be cloned. These two functions τ and σ have been generalized in our present framework to a span $\mathbb{L} \stackrel{l}{\leftarrow} \mathbb{K} \stackrel{r}{\rightarrow} R$ where graphs $\mathbb{L}$ and $\mathbb{K}$ are annotated with cloning indications. Handling termgraphs as in [7] requires some care to ensure the preservation of the arity (the number of outgoing edges) of a node during a transformation process. This requirement prevents from deletion of nodes and their incident edges in general. To ensure preservation of node arities, the function τ is required to be total. The problem of arity preserving does not appear in multigraphs. Thus, in our context, a node, n in $\mathbb{L}$, can actually be deleted (zero clone) with all its incident edges if, for instance, n has no antecedent in $\mathbb{K}$. With respect to cloning abilities, the HPO approach offers the possibility to make one or more copies of a node together with its outgoing edges. Therefore, this way of cloning nodes is limited to the outgoing edges only and contrasts with the flexible possibilities of cloning edges proposed in the present paper. In fact, whenever a graph G rewrites into H according to the HPO approach by using a rule (L, R, τ, σ), the graph G can also be rewritten into H according to a rule $\mathbb{L} \stackrel{l}{\leftarrow} \mathbb{K} \stackrel{r}{\rightarrow} R$ where morphisms l and r encode the functions τ and σ as described below.

Proposition 5.1. *Let ρ be an HPO rule (L, R, τ, σ). Then L and R are multigraphs, $\tau : |L| \rightarrow |R|$ is a total function and $\sigma : |R| \rightarrow |L|$ is partial function. Let C denote the domain of σ, seen as a graph without edges. Let us assume that every node in C has no outgoing edges (they are kind of variables). Let p be the rule $\mathbb{L} \stackrel{l}{\leftarrow} \mathbb{K} \stackrel{r}{\rightarrow} R$ defined as follows*

- *K is a graph without edges and $|K| = |L| + |C|$,*
- *let n in $|L|$, then $l(n) = n$ and $r(n) = \tau(n)$,*
- *let n in $|C|$, then $l(c) = \sigma(c)$ and $r(c) = c$,*
- *$|K|^+ = |C|$ and $|L|^+ = l(|C|) = \sigma(|C|)$: nodes in C are dedicated to make clones with outgoing edges only.*
- *$|K|^- = |L|$ and $|L|^+ = l(|C|) = \sigma(|C|)$: this reflects the fact that the nodes of L undergo global redirection of incoming pointers.*
- *$L^\star$ is empty.*

Then, for an injective matching $m : L \rightarrow G$, $G \stackrel{hpo}{\Longrightarrow} H$ implies $G \stackrel{pc}{\Longrightarrow} H$.

Cloning is also one of the features of the sesqui-pushout approach to graph transformation [4]. In this approach, a rule is a span $L \leftarrow K \rightarrow R$ of multigraphs and the application of a rule to a graph G can be illustrated by the same figure as for a DPO step (as in the introduction), where the right-hand side is a pushout as in the DPO approach but the left-hand side is a pullback, and moreover it is a final pullback complement. The sesqui-pushout approach and ours mainly differ in the way of handling cloning. In [4], the cloning of a node is performed by copying all incident edges (incoming and outgoing edges) of the cloned node. This is a particular case of our way of cloning nodes. The use of polarized multigraphs helped us to specify for every clone, the way incident edges can be copied. Therefore, a sesqui-pushout rewriting step can be simulated by a rewriting step with polarized rules, but the converse does not hold in general.

Proposition 5.2. *Let ρ be the rewrite rule $L \leftarrow K \rightarrow R$. Let ρ' be the rule $\mathbb{L} \stackrel{l}{\leftarrow} \mathbb{K} \stackrel{r}{\rightarrow} R$ such that*

- $|K|^+ = |K|^- = |K|$, $K^\star = \emptyset$,
- $|L|^+ = |L|^- = |L|$, $L^\star = \emptyset$,

Then, for an injective matching $m : L \rightarrow G$, $G \stackrel{sqpo}{\Longrightarrow}_\rho H$ implies $G \stackrel{pc}{\Longrightarrow}_{\rho'} H$.

In [4], the sesqui-pushout approach has been compared to the classical double pushout and single pushout approaches. Corradini et al. showed that the sesqui-pushout and the DPO approaches coincide under some conditions (see [4, Proposition 12]). They also showed how the sesqui-pushout approach can be simulated by the SPO approach and gave conditions under which a SPO derivation can be simulated by a sesqui-pushout (see [4, Proposition 13]). So, according to Proposition 5.2, which shows how to simulate a sesqui-pushout step in our setting, we can infer the same comparisons with respect to DPO and SPO for our graph rewriting definition.

Cloning has also been subject of interest in [6]. The authors considered rewrite rules of the form $S := R$ where S is a star, i.e., S is a (nonterminal) node surrounded by its adjacent nodes together with the edges that connect them. Rewrite rules which perform the cloning of a node have been given in [6, Def. 6]. These rules show how a star can be removed, kept identical to itself or copied (cloned) more than once. Here again, unlike our framework, the cloning does not care about the arity of the nodes and, as in the case of the sesqui-pushout approach, a node is copied together with all its incoming and outgoing edges.

References

[1] F. Baader and T. Nipkow. *Term rewriting and all that.* Cambridge University Press, 1998.

[2] H. Barendregt, M. van Eekelen, J. Glauert, R. Kenneway, M. J. Plasmeijer, and M. Sleep. Term graph rewriting. In *PARLE'87*, pages 141–158. Springer Verlag LNCS 259, 1987.

[3] R. V. Book and F. Otto. *String-rewriting systems.* Springer-Verlag, 1993.

[4] A. Corradini, T. Heindel, F. Hermann, and B. König. Sesqui-pushout rewriting. In *Third International Conference on Graph Transformations (ICGT 06)*, volume 4178 of *Lecture Notes in Computer Science*, pages 30–45. Springer, 2006.

[5] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe. Algebraic approaches to graph transformation - part I: Basic concepts and double pushout approach. In *Handbook of Graph Grammars*, pages 163–246, 1997.

[6] F. Drewes, B. Hoffmann, D. Janssens, M. Minas, and N. V. Eetvelde. Adaptive star grammars. In *Third International Conference on Graph Transformations (ICGT 06)*, volume 4178 of *Lecture Notes in Computer Science*, pages 77–91. Springer, 2006.

[7] D. Duval, R. Echahed, and F. Prost. A heterogeneous pushout approach to term-graph transformation. In *20th International Conference on Rewriting Techniques and Applications, RTA 2009*, pages 194–208, 2009.

[8] R. Echahed. Inductively sequential term-graph rewrite systems. In *4th International Conference on Graph Transformations, ICGT*, volume 5214 of *Lecture Notes in Computer Science*, pages 84–98. Springer, 2008.

[9] R. Echahed and J. C. Janodet. Admissible graph rewriting and narrowing. In *Proc. of Joint International Conference and Symposium on Logic Programming (JICSLP'98)*, pages 325–340. MIT Press, June 1998.

[10] H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 2: Applications, Languages and Tools.* World Scientific, 1999.

[11] H. Ehrig, R. Heckel, M. Korff, M. Löwe, L. Ribeiro, A. Wagner, and A. Corradini. Algebraic approaches to graph transformation - part ii: Single pushout approach and comparison with double pushout approach. In *Handbook of Graph Grammars*, pages 247–312, 1997.

[12] H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 3: Concurrency, Parallelism and Distribution*. World Scientific, 1999.

[13] H. Ehrig, M. Pfender, and H. J. Schneider. Graph-grammars: An algebraic approach. In *14th Annual Symposium on Foundations of Computer Science (FOCS), 15-17 October 1973, The University of Iowa, USA*, pages 167–180. IEEE, 1973.

[14] R. Kennaway. On "on graph rewritings". *Theor. Comput. Sci.*, 52:37–58, 1987.

[15] M. Löwe. Algebraic approach to single-pushout graph transformation. *Theor. Comput. Sci.*, 109(1&2):181–224, 1993.

[16] Saunders Mac Lane. *Categories for the Working Mathematician. 2nd edition. Graduate Texts in Mathematics 5, Springer-Verlag (1997).*

[17] R. Plasmeijer and M. van Eekelen. Functional Programming and Parallel Graph Rewriting. *Addison-Wesley, 1993.*

[18] D. Plump. Term graph rewriting. In H. Ehrig, G. Engels, H. J. Kreowski, and G. Rozenberg, editors, Handbook of Graph Grammars and Computing by Graph Transformation*, volume 2, pages 3–61. World Scientific, 1999.*

[19] J. C. Raoult. On graph rewriting. Theoretical Computer Science*, 32:1–24, 1984.*

[20] G. Rozenberg, editor. Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations. *World Scientific, 1997.*