Reversibility and composition of rewriting in hierarchies

Russ Harmer, Eugenia Oshurko

Reversibility and composition of rewriting in hierarchies

Russ Harmer[0000−0002−0817−1029] and Eugenia Oshurko[0000−0003−1218−8170]

Univ Lyon, EnsL, UCBL, CNRS, LIP, F-69342 LYON Cedex 07, France
{russell.harmer,ievgeniia.oshurko}@ens-lyon.fr

Abstract. In this paper we study how SqPO rewriting of individual objects and hierarchies of objects can be reversed and how the composition of rewrites can be constructed. We introduce the notion of a rule hierarchy, study how such rule hierarchies can be applied to object hierarchies and analyse the conditions under which this application is reversible. We then present a theory for constructing the composition of consecutive hierarchy rewrites. We further illustrate how the reversibility and composition of rewriting can be used to design an audit trail for both individual graphs and graph hierarchies. This provides us a compact way to maintain the history of updates of an object including its multiple versions. The main application of the designed framework is an audit of updates to knowledge represented with hierarchies of graphs. The prototype system for transformations in hierarchies of simple graphs with attributes is implemented as a part of the ReGraph Python library.

1 Introduction

In this work we present a mathematical framework for building an *audit trail*—a traceable history of sesqui-pushout (SqPO) rewrites [3] operating on individual objects and hierarchies of objects from appropriately structured categories. A transparent audit trail provides insight into the history of object transformations and allows us to revert to an arbitrary point in this history. The latter feature is extremely useful, when, for example, trying to fix an erroneous transformation. Moreover, such an audit trail provides means for efficient accommodation of multiple versions of the same object arising, for example, as the result of inconsistent transformations.

An audit trail system with the desired capabilities heavily relies on: (1) the existence of an *efficient semantic representation of object transformations* (or deltas), which frees us from the necessity to store the state of the object at each point of its transformation history; (2) the *reversibility of transformations*, which guarantees that any sequence of tranformations can be 'undone'; (3) the existence of sound means for *composing a pair of successive transformations*, which allows us to efficiently store and switch between different versions of the same object. In this work we formulate the three above-mentioned ingredients with respect to both individual objects (Section 2) and hierarchies of objects (Section 3), and

discuss how these ingredients can be used to construct an efficient audit trail (Section 4).

The main application of interest to us is an audit trail system for the knowledge representation (KR) framework based on hierarchies of graphs presented in [7]. We would like to use this system to record the history of predominantly small localized updates of a large knowledge corpus represented with graphs and graph hierarchies, where storing the corpus at each point in the history is not feasible. Moreover, we would like to design means for maintaining multiple versions of the corpus, crucial when accommodating different versions of knowledge that, for example, correspond to different view-points of knowledge curators or some intrinsic knowledge conflicts (e.g. contradicting experimental results, alternative hypotheses). The main use cases of audit trails for KRs based on hierarchies of graphs include version control for updates in schema-aware graph databases presented in [2] and updates to the knowledge corpora provided by the bio-curation framework KAMI [6].

Related work

While the reversibility of SqPO rewriting is a well-studied problem [4], the composition of consecutive (and not necessarily sequentially independent) SqPO rewrites is a far less studied subject. The construction of such composition was described for the double pushout (DPO) approach (see D-concurrent production in [5]), the SqPO approach, where rules are linear [1] and where the right-hand side of the first applied rule is exactly the left-hand side of the second rule [9]. In this work we present such a construction for two consecutive applications of general SqPO rules, where the first rewriting is required to be reversible. This construction is inspired by [5] and adapted to SqPO rewriting. For the composition to be well-defined we require the underlying category to be adhesive [8]. The notion of a rule hierarchy and the questions of the reversibility and composition of hierarchy rewrites represent a novel direction that generalizes SqPO rewriting of individual objects to hierarchies of objects [7].

Finally, the main application of interest to us, a transformation audit trail, is closely related to the version control systems (VCSs) used in software development. While such systems typically provide control over different versions of software source code, our audit trail provides such control for different versions of a graph or a graph hierarchy. Similarly to VCSs, the transformation audit trail avoids maintaining the state of an object at the time of every transformation by keeping only its current state together with a compact representation of a history of transformations. Moreover, by using a mechanism similar to *delta compression* in VCSs, such audit trails allow the maintaining of multiple versions of the same object and switching between these versions.

2 Preliminaries

In this section we briefly present some useful notions that serve as preliminaries for the rest of this paper and allow us to constuct the desired audit trail for SqPO

rewiriting of individual objects. We introduce SqPO rewriting, its reversible version and add the third audit trail ingredient by presenting in Section 2.2 how the composition of two consecutive SqPO rewrites can be constructed when the first rewrite is reversible. Finally, we conclude this section by presenting the principal KR model of interest, a hierarchy of objects.

2.1 SqPO rewriting

SqPO rewriting is an approach for abstract deterministic rewriting in any category with pushouts (POs) and (final) pullback complements (PBCs) over monos [3]. SqPO rewriting allows us to perform the operations of addition, deletion, cloning and merging of elements, where by element we mean any concrete constituent of an object in a category of interest (such as nodes and edges in categories of graphs). Rewriting of an object G is defined by a *rule* $r : L \leftarrow r^- - P - r^+ \rightarrow R$ and its *instance* given by a mono $m : L \rightarrowtail G$. Application of r is performed in two phases as in Diagram 1: (a) an object G^- is constructed as the final PBC of r^- and m and (b) the final result of rewriting G^+ is constructed as the PO of m^- and r^+. An arbitrary rewrite of an object in a category of interest can be decoupled into two phases: the *restrictive* rewrite (Diagram 1a) performing deletion and cloning of elements and the *expansive* rewrite (Diagram 1b) performing merging and addition of elements.

$$\begin{array}{ccccc} L & \xleftarrow{r^-} & P & \xrightarrow{r^+} & R \\ \downarrow{\scriptstyle m}\ (a) & & \downarrow{\scriptstyle m^-}\ (b) & & \downarrow{\scriptstyle m^+} \\ G & \xleftarrow[g^-]{} & G^- & \xrightarrow[g^+]{} & G^+ \end{array} \quad (1)$$

Transformations of individual objects through SqPO rewriting can be efficiently represented with corresponding rewriting rules and their instances. However, such rewriting may introduce some *side-effects*, i.e. graph transformations not explicitly specified by the underlying rules and instances. The nature of these side-effects depends on the category in which we are working. For example, in both simple and non-simple graphs, some edges not matched by the left-hand side of the rule can be removed as a side-effect of a node removal. Due to such side-effects, having applied a rewriting rule to an object, we can no longer restore this object by simply looking at the applied rule and its instance. The reversible variant of SqPO rewriting that does not introduce side-effects was presented in [4]. It corresponds to the scenario where the SqPO rewriting diagram can be read both forwards and backwards. More formally:

Definition 1. *An SqPO rewriting corresponding to the application of a rule* $r : L \leftarrow r^- - P - r^+ \rightarrow R$ *through a matching* $m : L \rightarrowtail G$ *as in Diagram 1 is* reversible*, if the square (a) is also a PO and the square (b) is also a PBC, i.e.* $P \rightarrowtail m^- \rightarrow G^- - g^+ \rightarrow G^+$ *is the final PBC of* r^+ *and* m^+*. We call* $r^{-1} : R \leftarrow r^+ - P - r^- \rightarrow L$ *the* reverse *of* r*.*

2.2 Composition of SqPO rewriting

Let $r_1 : L_1 \leftarrow P_1 \rightarrow R_1$ be a rewriting rule applied to an object G_1 through an instance $m_1 : L_1 \rightarrowtail G_1$ and let G_2 be the result of application of this rule

(corresponding to Diagram 2). Let $r_2 : L_2 \leftarrow P_2 \rightarrow R_2$ be a rule applied to the resulting object G_2 through an instance $m_2 : L_2 \rightarrowtail G_2$ (as in Diagram 3).

$$\begin{array}{ccc} L_1 & \xleftarrow{r_1^-} & P_1 \xrightarrow{r_1^+} R_1 \\ \downarrow{\scriptstyle m_1} & & \downarrow{\scriptstyle m_1^-} \quad \downarrow{\scriptstyle m_1^+} \\ G_1 & \xleftarrow[g_1^-]{} & G_1^- \xrightarrow[g_1^+]{} G_2 \end{array} \qquad (2)$$

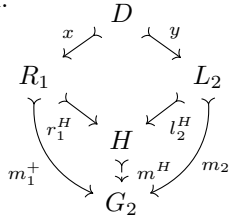
$$\begin{array}{ccc} L_2 & \xleftarrow{r_2^-} & P_2 \xrightarrow{r_2^+} R_2 \\ \downarrow{\scriptstyle m_2} & & \downarrow{\scriptstyle m_2^-} \quad \downarrow{\scriptstyle m_2^+} \\ G_2 & \xleftarrow[g_2^-]{} & G_2^- \xrightarrow[g_2^+]{} G_3 \end{array} \qquad (3)$$

Given these two consecutive rule applications, we would like to find a rule $L \leftarrow r^- - P - r^+ \rightarrow R$ and an instance $m : L \rightarrowtail G_1$ that, when applied to G_1, directly produces the object G_3, i.e. such that Diagram 4 is an SqPO diagram. Apart from being well-structured for SqPO rewriting, construction of such a composed rule will require from the category in which we are working to to be *adhesive* [8].

$$\begin{array}{ccc} L & \xleftarrow{r^+} & P \xrightarrow{r^-} R \\ \downarrow{\scriptstyle m} & & \downarrow{\scriptstyle m^-} \quad \downarrow{\scriptstyle m^+} \\ G_1 & \xleftarrow[g^-]{} & G_1^{\ominus} \xrightarrow[g^+]{} G_3 \end{array} \qquad (4)$$

Let us first proceed by constructing the pullback (PB) $R_1 \leftarrow x \prec D \succ y \rightarrow L_2$ from m_1^+ and m_2. Note that, because PBs preserve monos, arrows x and y are monos. We will call the span given by this PB the *overlap* of R_1 and L_2 given their matching inside G_2, and we will denote it with o. Intuitively D indicates whether the two rule applications are *sequentially independent*, i.e. the two rules operate on disjoint parts of G_2 [4]. When D is non-empty, the first rule can produce elements that are 'consumed' by the second rule, i.e. for the second rule to be applied the first one should have been applied.

The PO $R_1 \succ l_1^H \rightarrow H \leftarrow l_2^H \prec L_2$ from x and y as in Diagram 5 constructs the object H that can be seen as the union of two patterns R_1 and L_2 given their overlap. By the universal property (UP) of POs, there exists a unique homomorphism $m^H : H \rightarrow G_2$ that renders the diagram commutative.



(5)

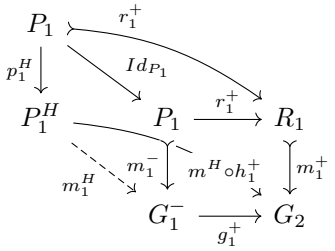
This homomorphism gives us the PO factorization of the PB of m_1^+ and m_2. Because m_1^+ and m_2 are monos, by adhesivity, m^H is also a mono (see *Theorem 5.1.* in [8]). Using the object H we now construct two objects P_1^H and P_2^H given by the final PBC $P_1 \succ p_1^H \rightarrow P_1^H - h_1^+ \rightarrow H$ to r_1^+ and r_1^H and $P_2 \succ p_2^H \rightarrow P_2^H - h_2^- \rightarrow H$ to r_2^- and l_2^H as in Diagrams 6 and 7.

$$\begin{array}{ccc} P_1 & \xrightarrow{r_1^+} & R_1 \\ {\scriptstyle p_1^H}\downarrow & & \downarrow{\scriptstyle r_1^H} \\ P_1^H & \xrightarrow[h_1^+]{} & H \end{array} \qquad (6)$$

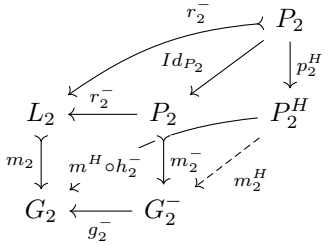
$$\begin{array}{ccc} L_2 & \xleftarrow{r_2^-} & P_2 \\ {\scriptstyle l_2^H}\downarrow & & \downarrow{\scriptstyle p_2^H} \\ H & \xleftarrow[h_2^-]{} & P_2^H \end{array} \qquad (7)$$

For the first PBC to be 'meaningful', we need to make an important assumption that the application of r_1 is *reversible*. Having made this assumption, the object P_1^H can be interpreted as the result of reverting the rewrite specified by r_1^+ on H. On the other hand, the second PBC simply applies the rewrite spec-

ified by the arrow r_2^- to H. It is easy to demonstrate that, by the UP of final PBCs, there exist unique arrows $m_1^H : P_1^H \to G_1^-$ and $m_2^H : P_2^H \to G_2^-$ that render Diagrams 8 and 9 commutative. Moreover, m_1^H and m_2^H are monos.



(8)



(9)

To understand how the non-reversibility of the first rule prevents us from finding a 'meaningful' P_1^H consider the following example.

Example 1. Let $P_1 \to R_1$ in Figure 1a represent an expansive phase of the first rewrite applied to G_1 (as in the right-most square in Diagram 2). It is easy to verify that this rewrite is not reversible, i.e. we cannot restore G_1^- by applying r_1^+ through m_1^+ (the depicted square does not form a PBC). Let L_2 in Figure 1b be the left-hand side of the second rule, applied to the resulting object G_2, then this figure corresponds to Diagram 5, where H represents a union of R_1 and L_2 given their overlap. Reverting the rewrite specified by r_1^+ on H gives us the object P_1^H depicted in Figure 1c, which splits the merged circle and triangle. This splitting reconnects the black square to both circle and triangle which prevents us from constructing a match $P_1^H \rightarrowtail G_1^-$ necessary to obtain the desired composition.

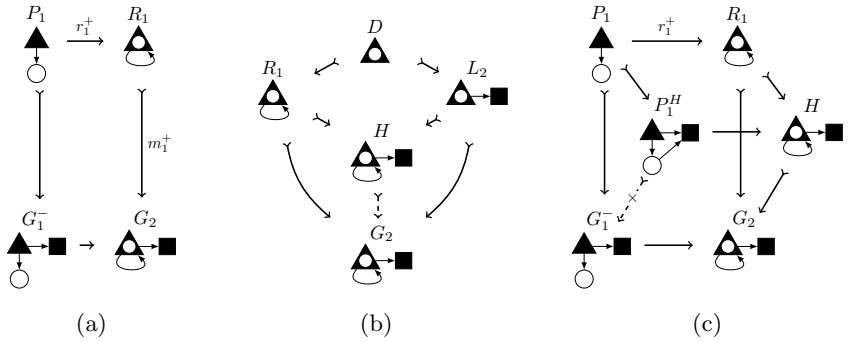


Fig. 1: Composition with an irreversible rule.

Next, let us construct two POs: $L_1 \rightarrowtail^{l_1^H} L \leftarrow^{h_1^-} P_1^H$ from r_1^- and p_1^H and $R_2 \rightarrowtail^{r_2^H} R \leftarrow^{h_2^+} P_2^H$ from r_2^+ and p_2^H. The first PO reverts the rewrite of P_1^H specified by r_1^-, and the second performs the rewrite of P_2^H using r_2^+ and p_2^H. The constructed object L represents the result of reversing the transformation of the pattern P_1^H specified by r_1^- and the instance p_1^H, precisely because the application of r_1 is reversible. By the UP of these POs we can construct unique matches $m : L \to G_1$ and $m^+ : R \to G_3$.

Finally, to construct the rule composition, we find the PB $P_1^H \leftarrow p' - P - p'' \rightarrow P_2^H$ from h_1^+ and h_2^-. The resulting rule corresponds to the span $r : L \leftarrow h_1^- \circ p' - P - h_2^+ \circ p'' \rightarrow R$. We will refer to it as the *composed rule* given the overlap o and write $r = \otimes(r_1, o, r_2)$.

Theorem 1 (Synthesis). *In adhesive categories, if rewriting given by r_1 is reversible, application of the rule r given by $\otimes(r_1, o, r_2)$ with the instance $m : L \rightarrowtail G_1$ produces the object G_3, i.e. Diagram 4 with $r^- = h_1^- \circ p'$ and $r^+ = h_2^+ \circ p''$ is an SqPO diagram.*

Proposition 1. *In adhesive categories, the composition of two reversible rewrites is a reversible rewrite.*

2.3 Hierarchies and SqPO rewriting in hierarchies

A *hierarchy* of objects in a category $\mathbf{C}$ is a directed acyclic graph (DAG) whose nodes are objects and whose edges are arrows from $\mathbf{C}$ such that all paths between each pair of objects are equal [7]. We refer to the latter condition as the *commutativity* condition. In the rest of this paper we assume that we are working in a fixed category $\mathbf{C}$ that has an appropriate structure for SqPO rewriting. For the commutativity of a hierarchy to be maintained, an SqPO rewrite of an object situated inside the hierarchy may require updates to other objects and arrows called *propagation*.

The main model of interest to us operates on hierarchies of (simple) graphs and uses both hierarchy objects and arrows to represent knowledge. Hierarchies provide a powerful formalism for representation and update of fragmented knowledge on different interrelated abstraction levels. In this model, an edge of a hierarchy associated to an arrow $h : G \rightarrow T$ is often interpreted as *typing*, i.e. the graph T defines the kinds of nodes and edges that can exist in G. In this context, the commutativity condition guarantees that the representation of knowledge on different abstraction levels is consistent. The propagation framework presented in [7] allows us to transform individual graphs inside a hierarchy and perform the *co-evolution* of its different layers, which guarantees the consistency of knowledge at all times.

When the knowledge represented in a hierarchy is frequently updated by potentially different curators, it is often desirable to maintain the history of updates and be able to store multiple versions of knowledge at the same time. The design of a mathematical system providing such features, thus, constitutes the principal motivation for this work, and this system relies on the existence of a compact representation of object transformations. The semantics of SqPO rewriting allows us to efficiently represent object transformations using rules. To be able to build and study such a representation for transformations in hierarchies let us briefly formulate the semantics of rewriting and propagation in hierarchies (formal details can be found in [7]).

Rewriting an individual object situated at a node of a hierarchy affects the objects associated at its ancestor and the descendant nodes, while the rest of

the objects stay unchanged. The restrictive phase is propagated *backwards* to all the objects typed by the target of rewriting. For instance, let G be an object corresponding to an ancestor of an object T in a hierarchy with an associated arrow $h : G \to T$. A restrictive rewrite performing deletion and cloning of elements in T induces propagation to instances of these elements in G. The expansive phase is propagated *forward* to all the objects typing the target. For instance, for $h : G \to T$ as before, an expansive rewrite performing addition and merging of elements in G induces propagation to T affecting the types of these elements. The commutativity of the updated hierarchy is guaranteed by the *composability conditions* imposed on the performed propagations.

3 Rule hierarchies

In this section we formulate the notion of a *rule hierarchy* that serves us as a compact representation of coupled transformations of objects in a hierarchy. We describe how a rule hierarchy can be applied to the corresponding hierarchy of objects through specified instances and study the side-effects introduced by the application of a rule hierarchy. Such side-effects, apart from the side-effects introduced by SqPO rewriting on objects, may include some implicit changes to the homomorphisms in the hierarchy. Thus, we formulate the conditions under which a given application of a rule hierarchy is reversible. Finally, we present how consecutive rewrites of a hierarchy can be composed. In this section we consider SqPO rewriting rules operating on objects from $\mathbf{C}$. Such rules are spans formed by objects and arrows from $\mathbf{C}$.

Definition 2. *A* rule homomorphism *f from $r_1 : L_1 \leftarrow r_1^- - P_1 - r_1^+ \rightarrow R_1$ to $r_2 : L_2 \leftarrow r_2^- - P_2 - r_2^+ \rightarrow R_2$ is given by three arrows $\lambda : L_1 \to L_2$, $\pi : P_1 \to P_2$ and $\rho : R_1 \to R_2$ from $\mathbf{C}$ such that $\lambda \circ r_1^- = r_2^- \circ \pi$ and $r_2^+ \circ \pi = \rho \circ r_1^+$.*

Using rules as objects and rule homomorphisms as arrows, we obtain the category of rules $\mathbf{Rule}[\mathbf{C}]$ over the category $\mathbf{C}$.

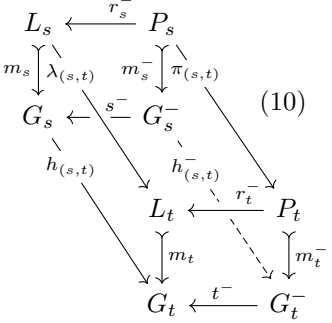
Definition 3. *A* rule hierarchy *is a hierarchy of objects in the category of rules.*

Let $\mathcal{H}$ be a hierarchy of objects in $\mathbf{C}$ and $\mathcal{R}$ be a hierarchy of rules operating on objects in $\mathbf{C}$ both defined over the same DAG $\mathcal{G} = (V, E \subseteq V \times V)$. We refer to such $\mathcal{G}$ as the *skeleton* of $\mathcal{H}$ and $\mathcal{R}$. For the sake of simplicity, in the rest of this section we will assume that we are working on a fixed pair $(\mathcal{H}, \mathcal{R})$ defined over the same skeleton. As a short-hand, for every node $v \in V$ we will denote the object associated to v in $\mathcal{H}$ with G_v and the rule associated to v in $\mathcal{R}$ with $r_v : L_v \leftarrow r_v^- - P_v - r_v^+ \rightarrow R_v$. For every edge $(s,t) \in E$ we will denote the associated homomorphism in $\mathcal{H}$ as $h_{(s,t)}$ and the arrows constituting the rule homomorphism in $\mathcal{R}$ as $\lambda_{(s,t)}$, $\pi_{(s,t)}$ and $\rho_{(s,t)}$.

Definition 4. *An* instance *of $\mathcal{R}$ in $\mathcal{H}$ is given by a function $\mathcal{I} : V \to Monos(\mathbf{C})$ that associates every node of the skeleton to an instance of the corresponding rule from $\mathcal{R}$ in the corresponding object from $\mathcal{H}$, i.e. $\mathcal{I}(v) : L_v \rightarrowtail G_v$ for all $v \in V$. For every node $v \in V$ we will denote the instance $\mathcal{I}(v)$ as m_v.*

Definition 5. *$\mathcal{R}$ is* applicable *to $\mathcal{H}$ through an instance $\mathcal{I}$, if for any pair of nodes $s, t \in V$ such that $(s,t) \in E$:*

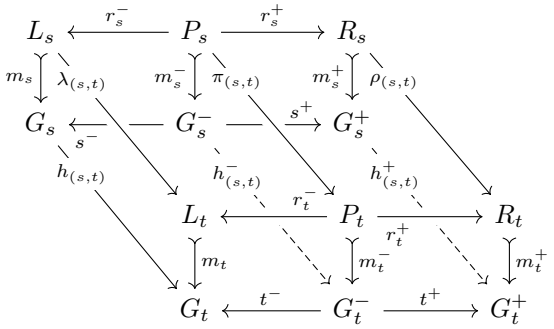
- *$h_{(s,t)} \circ m_s = m_t \circ \lambda_{(s,t)}$, i.e. their instances commute;*
- *if G_s^- and G_t^- are the results of the restrictive phase of rewriting given by the final PBC of r_s^- and m_s, and the final PBC of r_t^- and m_t respectively, then there exists a unique $h^-_{(s,t)} : G_s^- \to G_t^-$ that renders Diagram 10 commutative.*



(10)

Remark 1. Observe that, if the left face in Diagram 10 is a PB, $\mathcal{R}$ is always applicable to $\mathcal{H}$ as the unique arrow $h^-_{(s,t)}$ exists by the UP of final PBCs.

To rewrite $\mathcal{H}$ using the rule hierarchy $\mathcal{R}$, applicable through an instance $\mathcal{I}$, for every node of the skeleton we simply apply the associated rule to the associated object through the instance specified by $\mathcal{I}$. To restore the arrows of $\mathcal{H}$, for every edge $(s,t) \in E$, we use the applicability condition and the UP of POs as follows. Let the back and the front faces of the cube in Diagram 11 be two SqPO diagrams corresponding to the above-mentioned rewriting of the objects G_s and G_t respectively. First of all, by the applicability of $\mathcal{R}$ given $\mathcal{I}$, there exists a unique arrow $h^-_{(s,t)}$ such that $h_{(s,t)} \circ s^- = t^- \circ h^-_{(s,t)}$ and $h^-_{(s,t)} \circ m_s^- = m_t^- \circ \pi_{(s,t)}$. This enables us to use the UP of the PO G_s^+ and show that there exists a unique arrow $h^+_{(s,t)}$ that renders Diagram 11 commutative.



(11)

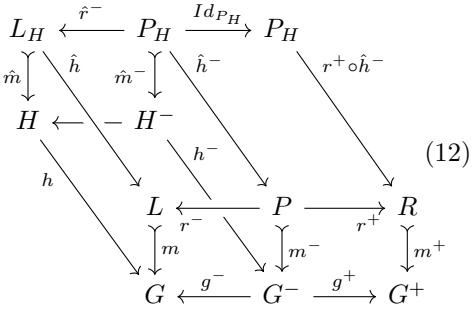
Therefore, the introduced notion of a rule hierarchy can be used as a compact representation of coupled updates in hierarchies of objects. Rules describe tranformations of hierarchy objects and rule homomorphisms allow us to restore homomoprhisms between them. The commutativity condition imposed on the rule homomorphisms in a hierarchy guarantees that their application results in a valid hierarchy of objects.

3.1 Expressing rewriting and propagation in hierarchies

In this subsection we will briefly discuss how the transformations of objects and arrows in a hierarchy $\mathcal{H}$ induced by a rewrite of an object G with a rule

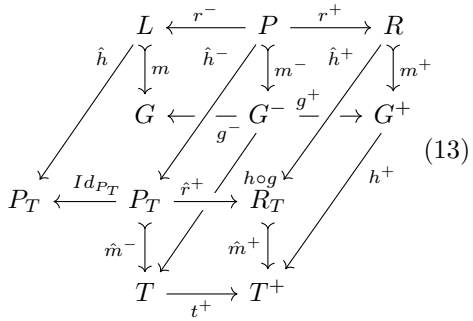
$r : L \leftarrow r^- - P - r^+ \rightarrow R$ through $m : L \rightarrowtail G$ can be represented as a rule hierarchy and its instance. Recall that, upon rewriting of an object in a hierarchy, the objects associated to ancestors and descendants of the origin of rewriting are updated according to the framework of backward and forward propagation [7], while the rest of the objects stay unaffected. We would like to construct a rule hierarchy that is defined over the skeleton of $\mathcal{H}$ and, therefore, contains rules for both affected and unaffected objects. For the sake of conciseness, here we will focus only on non-trivial updates to objects, i.e. on the construction of a rule *subhierarchy* corresponding to the objects updated as the result of backward or forward propagation. Moreover, we will omit the technical details of the constructions involved, but rather give a high-level idea behind them.

Backward propagation rules. Let H be an object corresponding to an ancestor of an object G in a hierarchy with an associated arrow $h : H \rightarrow G$. Backward propagation of r^- to H can be expressed as a rule $\hat{r} : L_H \leftarrow \hat{r}^- - P_H - Id_{P_H} \rightarrow P_H$ with an instance $\hat{m} : L_H \rightarrowtail H$ that, when applied to H, results into an object H^- homomorphic to the result of the original rewriting of G in a way that makes Diagram 12 commutes. Such a rule, called the *lifting* of r^-, is constructed given a specification for backward propagation (i.e. a rule factorization and a clean-up arrow [7]), and contains only restrictive updates (given by deletion and cloning). Informally, such specification indicates which changes to G should be propagated to H and how the updated H can be 'retyped' by the updated G. Such typing is obtained as a composition $g^+ \circ h^-$ in the diagram. Together with the propagation rule we obtain the arrows $\hat{h} : L_H \rightarrow L$ and $\hat{h}^- : P_H \rightarrow P$ defining the rule homomorphism given by $(\hat{h}, \hat{h}^-, r^+ \circ \hat{h}^-)$ as in the diagram. Therefore, the backward propagation framework allows us to extract a rule representing the specified propagation for every ancestor of the rewritten object together with a homomorphisms to the original rule r. Moreover, under appropriate conditions (see *backward composability* in [7]), we can construct the homomorphisms between backward propagation rules required by the skeleton of the hierarchy.



(12)

Forward propagation rules. Let T be an object corresponding to a descendant of an object G in a given hierarchy with an associated homomorphism $h : G \rightarrow T$. Forward propagation of r^+ to T can be expressed as a rule $\hat{r} : P_T \leftarrow P_T \rightarrow R_T$ with an instance $\hat{m}^- : P_T \rightarrowtail T$ that, when applied to T, results into an object T^+



(13)

to which the result of the original rewriting of G is homomorphic in a way renders Diagram 13 commutative. Such a rule, called the *projection* of r^+, is constructed given a specification for forward propagation (i.e. a rule factorization and a clean-up arrow [7]), and contains only expansive updates (given by addition and merging). Informally, such specification indicates which changes to G should be propagated to T and how the updated G can be 'retyped' by the updated T. Together with the propagation rule we obtain three arrows $\hat{h} : L \to P_T$, $\hat{h}^- : P \to P_T$ and $\hat{h}^+ : R \to R_T$ defining the rule homomorphism given by $(\hat{h}, \hat{h}^-, \hat{h}^+)$. Therefore, the forward propagation framework allows us to excerpt a rule representing the specified propagation for every descendant of the rewritten object together with homomorphisms from the original rule r. Moreover, under appropriate conditions (see *forward composability* in [7]), we can construct the homomorphisms between forward propagation rules required by the skeleton of the hierarchy.

Example 2. Consider the hierarchy $\mathcal{H}$ depicted in Figure 2. Let G be the target of rewriting with the rule highlighted in the gray area: this rule clones the circle into two semi-circles and merges one of these circles with the square node. Let the rules delimited with dashed arrows be the objects of the rule hierarchy representing respective rewriting and propagation. The rule $L_H \leftarrow P_H \to R_H$ is a backward propagation rule that specifies how the cloning in G is propagated to different instances of the circle. The rule $L_T \leftarrow P_T \to R_T$ is a forward propagation rule that describes the merging of the nodes typing the semi-circle and the square in G. Finally, the hierarchy $\mathcal{H}'$ on the right represents the result of the rule hierarchy application.

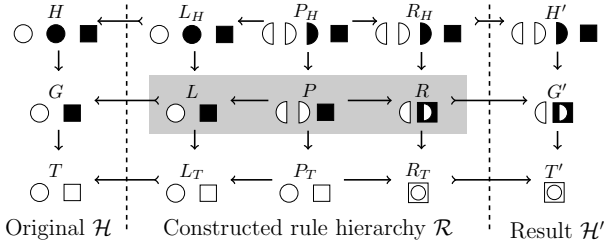


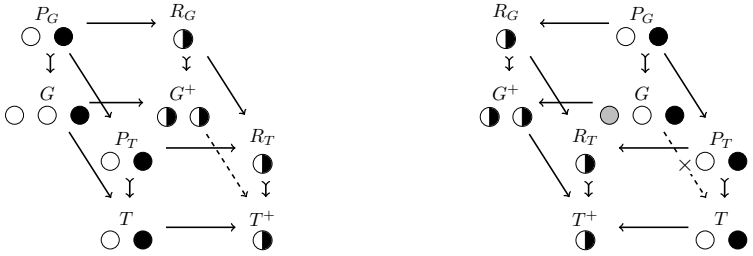
Fig. 2: Example of rewriting in a hierarchy represented with a rule hierarchy.

3.2 Reversible rewriting in hierarchies

In this subsection we study the side-effects introduced by the application of a rule hierarchy. These side-effects may induce some implicit changes to the homomorphisms representing hierarchy edges, which may prevent us from restoring the original hierarchy by simply looking at the applied rule hierarchy. In general, such side-effects make the rewriting produced by reversing the original rule hierarchy not *applicable*. Let us first consider the following example.

Example 3. Let $G \to T$ in Figure 3a be two homomorphic objects and let $P_G \to R_G$ and $P_T \to R_T$ specify expansive phases of rules applied to these objects. G

has two instances of the white circle and the rule $P_G \to R_G$ merges one of them with an instance of the black circle. The rule $P_T \to R_T$ merges the white and the black circle. The unique arrow $G^+ \to T^+$ is given by the UP of the PO that gives G^+. As a side-effect, the first instance of the white circle is also typed by the merged node in T^+. As a consequence, we 'forget' that it was an instance of the white circle in T. In Figure 3b we reverse the rules and apply them to the results G^+ and T^+: the merged node in T^+ is cloned into two circle nodes and one instance of the merged node in G^+ is cloned. As the result, we recover the object G, but we are not able to type it by T, which happens precisely because we 'forgot' how the circle denoted with gray was typed in T.



(a) Application of the original rules (b) Application of the reversed rules

Fig. 3: Example of side-effects affecting hierarchy homomorphisms

Definition 6. *The* reverse $\mathcal{R}^{-1}$ *of* $\mathcal{R}$ *is the rule hierarchy whose nodes correspond to the rules* r_v^{-1} *for all* $v \in V$*, and whose edges correspond to the rule homomorphisms* $(\rho_{(s,t)}, \pi_{(s,t)}, \lambda_{(s,t)})$ *for all edges* $(s,t) \in E$*.*

Definition 7. *Rewriting of* $\mathcal{H}$ *with* $\mathcal{R}$*, applicable through an instance* $\mathcal{I}$*, is* reversible*, if rewriting of every individual object is reversible and the reverse* $\mathcal{R}^{-1}$ *is applicable, i.e. for any pair of nodes* $s, t \in V$ *such that* $(s,t) \in E$ *corresponding to objects and rules as in Diagram 11, if* G_s^- *is given as the final PBC of* r_s^+ *and* m_s^+ *and* G_t^- *as the final PBC of* r_t^+ *and* m_t^+*, there exists a unique homomorphism* $h_{(s,t)}^- : G_s^- \to G_t^-$ *that makes the right cube in Diagram 11 commute.*

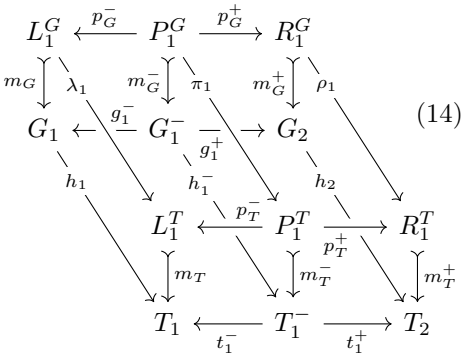
Even though the latter definition imposes rather abstract requirements, intuitive sufficient conditions can be formulated as follows. If for every hierarchy edge the left-most face in Diagram 11 is a PB, the rewriting is reversible (the proof is outside the scope of this paper). Informally, this guarantees that all the instances of the elements selected by m_t are also selected by m_s.

3.3 Composition of rewriting in hierarchies

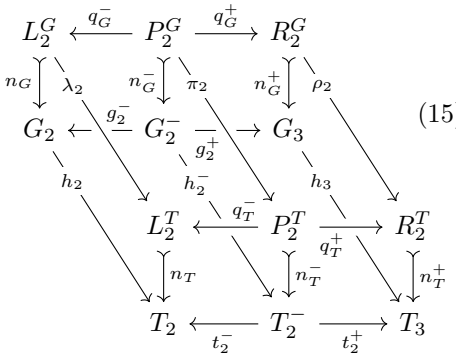
To study composition of consecutive rewriting in a hierarchy, we will focus on a simple hierarchy with two nodes and one edge, corresponding to objects G_1, T_1 and a homomorphism $h_1 : G_1 \to T_1$. Composition of rewriting in general

hierarchies can be trivially obtained by applying the presented technique to every pair of hierarchy nodes connected with an edge.

Let $\mathcal{H}$ be a hierarchy corresponding to two homomorphic objects $G_1 \rightarrow T_1$ and let $\mathcal{R}_1$ be a rule hierarchy corresponding to rules $p_G : L_1^G \leftarrow P_1^G \rightarrow R_1^G$ and $p_T : L_1^T \leftarrow P_1^T \rightarrow R_1^T$, whose homomorphism $f^p : p_G \rightarrow p_T$ is given by arrows λ_1, π_1 and ρ_1 as in Diagram 14. Let $G_2 \rightarrow T_2$ correspond to the result of the application of $\mathcal{R}_1$ through the instances m_G and m_T (we assume that $\mathcal{R}_1$ is applicable given m_G and m_T). Let $\mathcal{R}_2$ be another rule hierarchy given by a homomorphic pair of rules $q_G : L_2^G \leftarrow P_2^G \rightarrow R_2^G$ and $q_T : L_2^T \leftarrow P_2^T \rightarrow R_2^T$ as in Diagram 15. Their homomorphism $f^q : q_G \rightarrow q_T$ is given by arrows λ_2, π_2 and ρ_2. Let $G_3 \rightarrow T_3$ be the result of the application of $\mathcal{R}_2$ through the instances n_G and n_T as in the diagram (similarly, we assume that $\mathcal{R}_2$ is applicable given the instances).

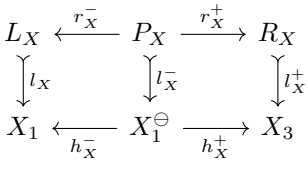


(14)



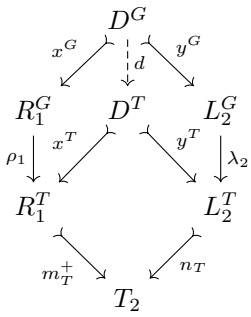
(15)

We can compose these pairs of rewrites using the constructions presented in Subsection 2.2. Namely, if the rules p_G and p_T are reversible, we can find a pair of rules $r_G : L_G \leftarrow P_G \rightarrow R_G$ and $r_T : L_T \leftarrow P_T \rightarrow R_T$, together with a pair of instances $l_G : L_G \rightarrowtail G_1$ and $l_T : L_T \rightarrowtail T_1$, such that, applying r_G to G_1 and r_T to T_1 through l_G and l_T respectively (as in Diagram 16, where X stands for G or T), we directly obtain G_3 and T_3 from Diagram 15.



(16)

To be able to construct a rule homomorphism $f : r_G \rightarrow r_T$, we need to make an assumption that rewriting specified by $\mathcal{R}_1$ given m_G and m_T is reversible, i.e. for G_1^- being the final PCB to p_G^+ and m_G^+, and T_1^- to p_T^+ and m_T^+, there always exists a unique arrow h_1^- that renders the rightmost cube in Diagram 14 commutative. Let D^G, x^G, y^G, D^T, x^T and y^T from Diagram 17 be the two overlaps of respectively R_1^G with L_2^G and R_1^T with L_2^T, constructed as described in Subsection 2.2 and denoted with o^G and o^T. By the UP of PBs, there exists a unique arrow



(17)

$d : D^G \to D^T$ using which we can construct a hierarchy of such overlaps defined over the same skeleton as $\mathcal{H}$, and together with arrows x^G, y^G, x^T and y^T, such a hierarchy gives us the *hierarchy overlap* $\mathcal{O}$.

Lemma 1. *If the rewriting of $\mathcal{H}$ given by $\mathcal{R}_1$ through m_G and m_T is reversible, then, given the hierarchy overlap $\mathcal{O}$, there exist a unique rule homomorphism $f : r_G \to r_T$.*

Therefore, we can construct the rule hierarchy $\mathcal{R}$ corresponding to $r_G - f \to r_T$. We will refer to it as the *composed rule hierarchy* given the hierarchy overlap $\mathcal{O}$ and write $\mathcal{R} = \otimes(\mathcal{R}_1, \mathcal{O}, \mathcal{R}_2)$.

Theorem 2. *In adhesive categories, if rewriting given by $\mathcal{R}_1$ is reversible, $\mathcal{R}$ is applicable given l_G and l_T, and its application results into $G_3 - h_3 \to T_3$.*

Proposition 2. *In adhesive categories, the composition of two reversible hierarchy rewrites is a reversible rewrite.*

4 Transformation audit trail

In this section we describe how reversibility and composition of rewriting can be used to construct the audit trail for transformations of individual objects and hierarchies of objects. The proposed framework is implemented as a part of the `ReGraph`[1] Python library for building hierarchical knowledge representations based on simple graphs with attributes.

4.1 Audit trail for object transformations

Let G^0 be the starting object whose history of transformations we would like to maintain and let $\langle r_i : L^i \leftarrow r_i^- - P^i - r_i^+ \to R^i \mid i \in [1 \dots n] \rangle$ be a sequence of rules consecutively applied to G^0 through the instances $m_i : L^i \rightarrowtail G^{i-1}$,

$$\begin{array}{ccccc} L^i & \xleftarrow{r_i^-} & P^i & \xrightarrow{r_i^+} & R^i \\ \downarrow{\scriptstyle m_i} & & \downarrow{\scriptstyle m_i^-} & & \downarrow{\scriptstyle m_i^+} \\ G^{i-1} & \xleftarrow[\bar{g}_i^-]{} & \bar{G}^{i-1} & \xrightarrow[\bar{g}_i^+]{} & G^i \end{array} \tag{18}$$

resulting in a sequence of objects $\langle G^i \mid i \in [1 \dots n] \rangle$ with $m_i^+ : R^i \rightarrowtail G^i$ for $1 \leq i \leq n$, i.e. such that for every $1 \leq i \leq n$, Diagram 18 is a SqPO diagram. To be able to build a sound audit trail, we additionally require such a sequence of rewrites to be reversible.

Definition 8. *The* audit trail *for the object G^n consists of the sequence of rules $\langle r_i \mid i \in [1 \dots n] \rangle$ and the right-hand side instances $m_i^+ : R^i \rightarrowtail G^i$ for $1 \leq i \leq n$.*

Figure 4 shows an example of an audit trail: grey circles represent the states of a potentially large graph whose history of transformations we record.

Rollback. Using such an audit trail we can *rollback* to any point in the history of transformations corresponding to some intermediate object G^i for $0 \leq i \leq n-1$ by applying the sequence rules $\langle r_j^{-1} \mid j \in [n \dots i+1] \rangle$ with the corresponding instances $m_j^+ : R^j \rightarrowtail G^j$ for $j \in [n \dots i+1]$.

[1] https://github.com/Kappa-Dev/ReGraph

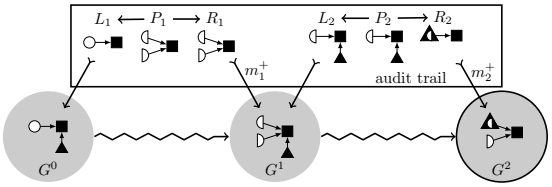


Fig. 4: Audit trail: the elements stored by the system are highlighted with solid lines.

Maintain diverged versions. To maintain multiple versions of an object in the audit trail, we use a technique known from VCSs as *delta compression*, i.e. only the current version of the object is stored, while the other versions are encoded in a *delta*, a representation of the 'difference' between the versions. Let v_1 and v_2 be two versions of the starting object G^0 with v_1 being the current version. Initial delta Δ from v_1 to v_2 is set to the identity rule (the rule that does not perform any transformations) $\emptyset \leftarrow Id_\emptyset - \emptyset - Id_\emptyset \rightarrow \emptyset$ and the instance $u : \emptyset \rightarrowtail G^0$, where $\emptyset$ stands for the initial object in $\mathbf{C}$ and u is the unique arrow from the initial object to G^0. Every rewrite of the current version of the object induces an update of the delta that consists in the composition of the previous delta and the reverse of the applied rule (recall that we assume that every rewriting in the audit trail is reversible). As before, let v_1 be the current version corresponding to some object G (e.g. obtained by transforming of G^0) and let $r_\Delta : L^\Delta \leftarrow r_\Delta^- - P^\Delta - r_\Delta^+ \rightarrow R_\Delta$ and $m_\Delta : L^\Delta \rightarrowtail G$ be respectively the rule and the instance given by Δ. Let $r : L \leftarrow r^- - P - r^+ \rightarrow R$ be a rule applied to G through the instance $m : L \rightarrowtail G$ and G' be the result of application of r given m. To update the delta, we compute the composition $\otimes(r^{-1}, o, r^\Delta)$ with o being a span $L \leftarrow x \prec D - y \rightarrow L^\Delta$ obtained as a PB from m and m_Δ. The new delta is, thus, set to the rule and the instance given by the composition $\otimes(r^{-1}, o, r_\Delta)$ (see an example in Figure 5).

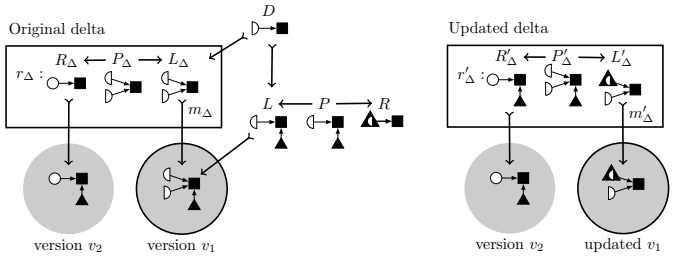


Fig. 5: Update of a delta representing different versions of an object.

Switch versions. Switching between different versions of the object can be done by simply applying the rule through the instance given by the delta. Namely, if v_1 is the current version corresponding to an object G with the delta to v_2 given by $\Delta = (r_\Delta, m_\Delta)$, switching to v_2 is performed by applying r_Δ to G through the instance m_Δ. If G' is the result of the above-mentioned rewriting and $m_\Delta^+ : R^\Delta \rightarrowtail G'$ is its right-hand side instance, then v_2 becomes the current version of the object and the new delta Δ is set to $(r_\Delta^{-1}, m_\Delta^+)$.

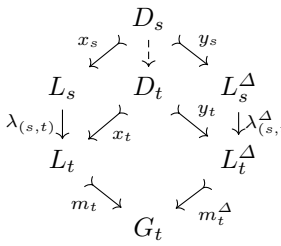
4.2 Audit trail for hierarchy transformations

Let $\mathcal{H}^0$ be the starting hierarchy of objects, defined over a skeleton $\mathcal{G} = (V, E)$, whose history of transformations we would like to maintain. Let $\langle \mathcal{R}_i \mid i \in [1 \dots n] \rangle$ be a sequence of rule hierarchies consecutively applied to $\mathcal{H}^0$ through the instances $\mathcal{I}_i$, resulting in a sequence of hierarchies $\langle \mathcal{H}^i \mid i \in [1 \dots n] \rangle$ with the right-hand side instances given by $\mathcal{I}_i^+$ for $1 \leq i \leq n$, i.e. for every $v \in V$, $\mathcal{I}_i^+(v) : R_v^i \rightarrowtail G_v^i$. As in the case of individual objects, to be able to build an audit trail, we require all the rewrites to be reversible.

Definition 9. *The* audit trail *for $\mathcal{H}^n$ consists of the sequence of rule hierarchies $\langle \mathcal{R}_i \mid i \in [1 \dots n] \rangle$ and the right-hand side instances $\mathcal{I}_i^+$ for $1 \leq i \leq n$.*

Rollback. Using the audit trail we can *rollback* to any point in the history of transformations corresponding to some intermediate hierarchy $\mathcal{H}^i$ for $0 \leq i \leq n-1$. This can be done by applying the rule hierarchies $\langle \mathcal{R}_j^{-1} \mid j \in [n \dots i+1] \rangle$ with the corresponding instances $\mathcal{I}_j^+$, where $\mathcal{I}_j^+(v) : R_v^j \rightarrowtail G_v^j$ for every $v \in V$ and $j \in [n \dots i+1]$.

Maintain diverged versions. To be able to accommodate multiple versions of a hierarchy, we use delta compression. Let v_1 and v_2 be two versions of the starting hierarchy $\mathcal{H}^0$ with v_1 being the current version. Initial delta Δ from v_1 to v_2 is set to the identity rule hierarchy with the rule $\emptyset \leftarrow Id_\emptyset - \emptyset - Id_\emptyset \rightarrow \emptyset$ at every node $v \in V$. We set the instance $\mathcal{I}(v)$ for every $v \in V$ to be the unique homomorphism $u_v : \emptyset \rightarrowtail G_v^0$. Every rewrite of the current version of the hierarchy induces an update of the delta that consists in the composition of the previous delta and the reverse of the applied rule hierarchy. Let v_1 be the current version corresponding to some hierarchy $\mathcal{H}$ (e.g. obtained as the result of transformation of $\mathcal{H}^0$). Let $\mathcal{R}_\Delta$ and $\mathcal{I}_\Delta$ be respectively the rule hierarchy and the instance given by Δ, where $r_v^\Delta : L_v^\Delta \leftarrow r_{v,\Delta}^- - P_v^\Delta - r_{v,\Delta}^+ \rightarrow R_v^\Delta$ and $m_v^\Delta : L_v^\Delta \rightarrowtail G_v$ are the rule and the instance corresponding to a node $v \in V$. Let $\mathcal{R}$ be a rule hierarchy applied to $\mathcal{H}$ through the instance $\mathcal{I}$ and $\mathcal{H}'$ be the result of the corresponding rewriting. The new delta is given by the rule hierarchy and the instance obtained as the composition $\otimes(\mathcal{R}^{-1}, \mathcal{O}, \mathcal{R}^\Delta)$ with $\mathcal{O}$ being the hierarchy overlap computed by finding the overlaps between L_v and L_v^Δ for every node $v \in V$ and the homomorphism $D_s \rightarrow D_t$ between overlaps given by the UP of PBs as in Diagram 19 for every edge $(s, t) \in E$.



(19)

Switch versions. Switching between different versions of the hierarchy is performed by applying the rule hierarchy though the instance given by the delta. If v_1 is the current version corresponding to a hierarchy $\mathcal{H}$ with the delta given by $\Delta = (\mathcal{R}_\Delta, \mathcal{I}_\Delta)$, switching to v_2 is performed by applying $\mathcal{R}_\Delta$ to $\mathcal{H}$ through $\mathcal{I}_\Delta$. For $\mathcal{H}'$ being the result of rewriting and $\mathcal{I}_\Delta^+$ being its right-hand side instance (where for every $v \in V$, $\mathcal{I}_\Delta^+(v) : R_v^\Delta \rightarrowtail G'_v$), v_2 becomes the current version of the object and the new delta Δ is set to $(\mathcal{R}_\Delta^{-1}, \mathcal{I}_\Delta^+)$.

5 Conclusions

In this paper we have described how the reversibility and composition of SqPO rewriting can be used to design an audit trail framework for individual objects and hierarchies of objects.

In particular, we have presented the construction that allows composing consecutive SqPO rewrites, where the first rewrite is reversible. We have also presented the notion of a rule hierarchy that generalizes SqPO rewriting to hierarchies of objects and allows for an efficient representation of rewriting and propagation in such hierarchies previously presented in [7]. We have studied the conditions under which an arbitrary rule hierarchy can be applied to the corresponding hierarchy of objects and described the conditions for such application to be reversible. We then briefly discussed the construction that can be used to compose consecutive applications of two rule hierarchies. Finally, we have described how an audit trail for individual objects and hierarchies of objects can be defined. Such an audit trail allows maintaining the history of transformations and provides means for reverting sequences of such transformations. Moreover, it enables accommodation of multiple versions of the same object diverged as the result of conflicting rewrites.

As a future work, we would like to study how an arbitrary transformation from a sequence of rewrites can be undone for individual objects and hierarchies of objects. This question is directly related to the theory of *causality* for SqPO rewriting and requires a generalization for such rewriting in hierarchies.

References

1. Behr, N.: Sesqui-pushout rewriting: Concurrency, associativity and rule algebra framework. arXiv preprint arXiv:1904.08357 (2019)
2. Bonifati, A., Furniss, P., Green, A., Harmer, R., Oshurko, E., Voigt, H.: Schema validation and evolution for graph databases. arXiv preprint arXiv:1902.06427 (2019)
3. Corradini, A., Heindel, T., Hermann, F., König, B.: Sesqui-pushout rewriting. In: Graph Transformations, pp. 30–45. Springer (2006)
4. Danos, V., Heindel, T., Honorato-Zimmer, R., Stucki, S.: Reversible sesqui-pushout rewriting. In: International Conference on Graph Transformation. pp. 161–176. Springer (2014)
5. Ehrig, H., Habel, A., Kreowski, H.J., Parisi-Presicce, F.: Parallelism and concurrency in high-level replacement systems. Mathematical Structures in Computer Science **1**(3), 361–404 (1991)
6. Harmer, R., Le Cornec, Y.S., Légaré, S., Oshurko, E.: Bio-curation for cellular signalling: the kami project. IEEE/ACM transactions on computational biology and bioinformatics **16**(5), 1562–1573 (2019)
7. Harmer, R., Oshurko, E.: Knowledge representation and update in hierarchies of graphs. In: International Conference on Graph Transformation. pp. 141–158. Springer (2019)
8. Lack, S., Sobociński, P.: Adhesive and quasiadhesive categories. RAIRO-Theoretical Informatics and Applications **39**(3), 511–545 (2005)
9. Löwe, M.: Polymorphic sesqui-pushout graph rewriting. In: International Conference on Graph Transformation. pp. 3–18. Springer (2015)