Unranked Second-Order Anti-Unification

Alexander Baumgartner, Temur Kutsia

Research Institute for Symbolic Computation
Johannes Kepler University Linz, Austria

Abstract

In this work we study anti-unification for unranked terms and hedges, permitting context and hedge variables. Hedges are sequences of unranked terms. The anti-unification problem of two hedges $\tilde{s}$ and $\tilde{q}$ is concerned with finding their generalization, a hedge $\tilde{g}$ such that both $\tilde{s}$ and $\tilde{q}$ are substitution instances of $\tilde{g}$. Second-order power is gained by using context variables to generalize vertical differences at the input hedges. Hedge variables are used to generalize horizontal differences. An anti-unification algorithm is presented, which computes a generalization of input hedges and records all the differences.

The algorithm is parametric by a skeleton computation function. For instance, we can compute a generalization of a skeleton which represents a constrained longest common subforest, or an agreement subhedge/subtree of the input hedges. The computation of the generalization is done in quadratic time.

Keywords: Higher-Order Anti-Unification for Unranked Terms and Hedges, Anti-Unification using Context and Hedge Variables, Generalization of Forests
2000 MSC: 68W05

1. Introduction

The anti-unification problem for two terms t_1 and t_2 requires finding their generalization: A term such that both t_1 and t_2 are instances of it under some substitutions. The interesting generalizations are least general ones (lggs). Anti-unification algorithms are supposed to compute lggs.

In 1970, Plotkin [24] and Reynolds [25] independently came up with essentially the same anti-unification algorithm. It was designed for first-order ranked terms (i.e., where function symbols have a fixed arity) in the syntactic case. Since then, a number of algorithms and their modifications have been developed, addressing the problem in various theories (e.g., [1, 4, 5, 9, 13, 23]) and from the point of view of different applications (e.g., [3, 8, 10, 16, 20, 19, 27]).

In this paper, we consider anti-unification for hedges, which are finite sequences of unranked terms. Such terms are constructed from function symbols

Email addresses: abaumgar@risc.jku.at (Alexander Baumgartner), kutsia@risc.jku.at (Temur Kutsia)

that do not have a fixed arity. We permit two kinds of variables: first-order, for hedges, and second-order, for contexts. Contexts that we consider here are hedges with a single occurrence of the distinguished symbol "hole". They are functions which can apply to another context or to a hedge, which are then "plugged" in the place of the hole.

Some applications of anti-unification indeed require higher-order features. For instance, reuse of proofs in program verification needs anti-unification with higher-order variables [20]. A restricted use of higher-order variables in generalizations turned out to be helpful for analogy making with Heuristic-Driven Theory Projection [16]. Anti-unification with combinator terms plays a role in replaying program derivations [12].

First-order anti-unification for ranked terms has been used to detect software code clones [8, 19]. It helps to achieve high-precision for clones obtained, essentially, by renaming and reformatting, but ranked anti-unification is not strong enough to detect clones obtained by omitting/inserting pieces of statements in the code. Unranked anti-unification can detect similarities not only between renamed parts of a hedge, but also between parts which differ from each other by inserting or omitting subparts, as was indicated in [18]. These features can be useful also for comparison of XML documents, which can be abstracted by unranked trees.

However, one important restriction of existing hedge anti-unification algorithms, such as, e.g., [18, 7, 28], is that the languages used in these algorithms do not permit higher-order variables. This imposes a natural restriction on solutions: The computed lggs do not reflect similarities between input hedges, if those similar pieces are located under distinct heads or at different depths. For instance, $f(a, b)$ and $g(h(a, b))$ are generalized by a single variable, although both terms contain a and b and a more natural generalization could be, e.g. $X(a, b)$, where X is a higher-order variable. In applications, it is often desirable to detect these similarities.

This is the problem we address here, permitting the use of context variables to abstract vertical differences between trees, and hedge variables used to abstract horizontal differences. The algorithm described in this paper first constructs a "skeleton" of a generalization of the input hedges, which corresponds to a hedge embedded into each of the input hedges. Next, it inserts context and/or hedge variables into the skeleton, which are supposed to uniformly generalize (vertical and horizontal) differences between input hedges, to obtain an lgg (with respect to the given skeleton). The skeleton computation function is the parameter of the algorithm: One can compute an lgg which contains, for instance, a constrained longest common subforest [2], or an agreement subhedge/subtree [14] of the input hedges.

In this paper we focus on the step of computing an lgg of two hedges, when the skeleton is already constructed. We assume that the latter is given in the form of an admissible alignment, which is a certain sequence of symbols occurring in both hedges, together with the positions these symbols occur in. We need to restrict variable occurrences in the generalization to guarantee that for each admissible alignment a unique lgg is computed. We use the idea of rigid

generalization from [18] about forbidding consecutive hedge variables, e.g., a sequence like (x, y) is not allowed if x and y are hedge variables. Since in this work second-order power is provided by using context variables, the above restriction is naturally extended to also forbid (vertical) chains of variables, e.g., terms like $X(Y(a))$ or $X(x)$ are not allowed, but $X(a, Y(a))$ is allowed as it can be seen as a branching in the tree representation of the term and therefore we do not consider it a chain (X, Y are context variables, x is a hedge variable, and a is a constant).

We develop an algorithm which takes two hedges and an admissible alignment and computes a rigid lgg of the hedges with respect to that alignment. The computed lgg is unique modulo variable renaming. Moreover, we can return not only the generalization, but also the differences between the input hedges, which tells us how one can obtain the original hedges from the generalization. The algorithm runs in quadratic time and requires linear space with respect to the size of the input. This result means that, for instance, if the skeleton is a constrained longest common subhedge of the input hedges in the sense of [29], then both skeleton and generalization computation can be done in quadratic time, because the time complexity of computing a constrained lcs is quadratic.

In some cases, the skeleton can be constructed in multiple ways, giving rise to several admissible alignments. It requires that the generalizations computed for each alignment should be compared to each other, to make the obtained set minimal. This problem requires matching with context and hedge variables in the minimization step and goes beyond the scope of this paper.

Example 1. The hedge $(X(a), f(X(g(a, x), c), x))$ is a generalization of two hedges $(h(a), f(h(g(a, b, b), c), b, b))$ and $(a, f(g(a, d), c, d))$. Dotted and dashed nodes indicate differences, while the solid ones form the admissible alignment. The first hedge can be obtained from the generalization by replacing the context variable X with the context $h(\circ)$ and the hedge variable x with the hedge (b, b). To obtain the second hedge, we need to replace X with the hole (i.e., to eliminate X) and to replace x by d.

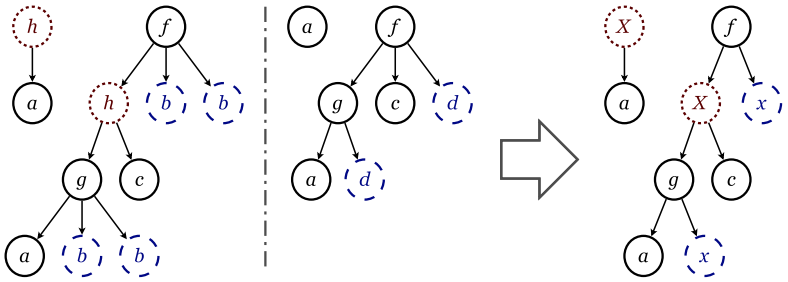

Figure 1: The hedges from Example 1 and their generalization.

A prototype implementation of the algorithm is available at `http://www.risc.jku.at/projects/stout/software/urauc.php`.

2. Preliminaries

Definition 1 (Terms, hedges, contexts). Given pairwise disjoint countable sets of unranked function symbols $\mathcal{F}$ (symbols without fixed arity), hedge variables $\mathcal{V}_{\mathsf{H}}$, unranked context variables $\mathcal{V}_{\mathsf{C}}$, and a special symbol $\circ$ (the hole), we define *terms*, *hedges*, and *contexts* by the following grammar:

$$\begin{array}{lll} t & := & x \mid f(\tilde{s}) \mid X(\tilde{s}) \qquad \text{(terms)} \\ \tilde{s} & := & t_1, \ldots, t_n \qquad \text{(hedges)} \\ \tilde{c} & := & \tilde{s}_1, \circ, \tilde{s}_2 \mid \tilde{s}_1, f(\tilde{c}), \tilde{s}_2 \mid \tilde{s}_1, X(\tilde{c}), \tilde{s}_2 \qquad \text{(contexts)} \end{array}$$

where $x \in \mathcal{V}_{\mathsf{H}}$, $f \in \mathcal{F}$, $X \in \mathcal{V}_{\mathsf{C}}$, and $n \geqslant 0$.

Hedges are finite sequences of terms, constructed over $\mathcal{F}$ and $\mathcal{V}_{\mathsf{H}} \cup \mathcal{V}_{\mathsf{C}}$. A term can be seen as a singleton hedge. A context can be seen as a hedge over $\mathcal{F} \cup \{\circ\}$ and $\mathcal{V}_{\mathsf{H}} \cup \mathcal{V}_{\mathsf{C}}$, where the hole occurs exactly once. A singleton context is then a term over $\mathcal{F} \cup \{\circ\}$ and $\mathcal{V}_{\mathsf{H}} \cup \mathcal{V}_{\mathsf{C}}$ with a single hole in it. To improve readability, we put non-singleton hedges and contexts between parenthesis.

We use the letters x, y, z for hedge variables and X, Y, Z for context variables. By f, g, h, a, b, c, d, e we denote function symbols, by $\tilde{s}, \tilde{q}, \tilde{r}, \tilde{g}, \tilde{h}$ hedges, by $\tilde{c}, \tilde{d}$ arbitrary contexts and by $\dot{c}, \dot{d}$ singleton contexts. We use ϕ, ψ for a context variable or a function symbol. The empty hedge is denoted by ϵ. Terms of the form $a(\epsilon)$ are written as just a.

A context $\tilde{c}$ can apply to a hedge $\tilde{s}$, denoted by $\tilde{c}[\tilde{s}]$, obtaining a hedge by replacing the hole in $\tilde{c}$ with $\tilde{s}$. Application of a context to a context is defined similarly.

Example 2. Examples of a term, a hedge, and a context are, respectively, $f(f(a), b)$, $(x, X(a, x), f(f(a), b))$, and $(x, X(a, x), f(f(\circ), b))$. The latter can be applied to a hedge $(a, X(a))$, resulting in $(x, X(a, x), f(f(\circ), b))[a, X(a)] = (x, X(a, x), f(f(a, X(a)), b))$.

The cardinality of a set A is denoted by $|A|$ and similarly $|\tilde{s}|$ denotes the length, i.e. number of elements, of a hedge $\tilde{s}$. The size of a hedge $\tilde{s}$ is the number of all symbols in it and it is written as $\|\tilde{s}\|$. We denote by $\tilde{s}|_i$ the ith element of $\tilde{s}$ and by $\tilde{s}|_i^j$ the subhedge $(\tilde{s}|_i, \ldots, \tilde{s}|_j)$. If $i > j$ then $\tilde{s}|_i^j$ is the empty hedge. The set of all function symbols which appear in a hedge $\tilde{s}$ (resp., in a context $\tilde{c}$) is denoted by $\mathcal{F}(\tilde{s})$ (resp., by $\mathcal{F}(\tilde{c})$). We overload the notation $\mathcal{F}(A)$ for the set of all function symbols which appear in a set of hedges and contexts A. Similarly we use $\mathcal{V}(\ldots)$, $\mathcal{V}_{\mathsf{H}}(\ldots)$, and $\mathcal{V}_{\mathsf{C}}(\ldots)$ to obtain the set of all variables, all the hedge variables, and all the context variables, respectively.

Definition 2 (Substitution). A *substitution* is a mapping from hedge variables to hedges and from context variables to contexts, which is identity almost everywhere. When substituting a context variable X by a context, the context will be applied to the argument hedge of X.

The symbols σ, ϑ are used to denote a substitution. Substitutions can be applied to hedges and contexts in the usual way. We use postfix notation for application, writing, e.g., $\tilde{s}\sigma$ for the application of σ to $\tilde{s}$. The notions *domain* and *range* of a substitution σ are standard and denoted by $\mathrm{Dom}(\sigma)$ and $\mathrm{Ran}(\sigma)$, respectively.

Example 3. Let $\sigma = \{x \mapsto \epsilon, y \mapsto (a, x), X \mapsto g(\circ)\}$ be a substitution, then $(X(x), y, f(X(y), c))\sigma = (g, a, x, f(g(a, x), c))$.

Definition 3 (Instantiation). A hedge $\tilde{s}$ is the *instance* of a hedge $\tilde{q}$ if there exists a substitution σ with $\tilde{q}\sigma = \tilde{s}$. We say that $\tilde{q}$ is more general than $\tilde{s}$ if $\tilde{s}$ is an instance of $\tilde{q}$ and denote this by $\tilde{q} \preceq \tilde{s}$. If $\tilde{q} \preceq \tilde{s}$ and $\tilde{s} \preceq \tilde{q}$, then we write $\tilde{q} \simeq \tilde{s}$. If $\tilde{q} \preceq \tilde{s}$ and $\tilde{q} \not\simeq \tilde{s}$, then we say that $\tilde{q}$ is strictly more general than $\tilde{s}$ and write $\tilde{q} \prec \tilde{s}$.

Definition 4 (Generalization). A hedge $\tilde{g}$ is a *generalization* of the hedges $\tilde{s}$ and $\tilde{q}$ if $\tilde{s}$ and $\tilde{q}$ are instances of $\tilde{g}$.

The word representation of a hedge is defined by the concatenation of the depth-first pre-order traversal of the constituent terms. For instance, $afgagbbc$ is the word representation of $(a, f(g(a, g(b, b)), c))$. Generalizations contain a common subsequence of the word representation of the input hedges. We will use this property in the formulation of our anti-unification algorithm. Observe, e.g., the hedges from Example 1:

$$\begin{array}{l} (\ h(\mathbf{a}), \mathbf{f}(\ h(\mathbf{g}(\mathbf{a}, b, b), \mathbf{c}), b, b)) \\ \underline{(\quad \mathbf{a}\ , \mathbf{f}(\quad \mathbf{g}(\mathbf{a}, d\quad), \mathbf{c}\ , d\quad))} \\ (X(\mathbf{a}), \mathbf{f}(X(\mathbf{g}(\mathbf{a}, x\quad), \mathbf{c}), x\quad)) \end{array}$$

Definition 5 (Position). The set of *positions* of a hedge $\tilde{s} = (t_1, \ldots t_n)$, denoted $pos(\tilde{s})$, is a set of strings of positive integers. It is defined as $pos(\tilde{s}) := \bigcup_{i=1}^{n} \{i \cdot p \mid p \in pos_{\mathsf{T}}(t_i)\}$, where $\cdot$ stands for concatenation. $pos_{\mathsf{T}}(t)$ is defined as $pos_{\mathsf{T}}(x) := \{\lambda\}$ and $pos_{\mathsf{T}}(\phi(\tilde{q})) := \{\lambda\} \cup pos(\tilde{q})$, where λ is the empty string.

Example 4. For instance, $pos(f(a, g(b, c))) = \{1, 1{\cdot}1, 1{\cdot}2, 1{\cdot}2{\cdot}1, 1{\cdot}2{\cdot}2\}$ and $pos(a, f(b, g(c)), d) = \{1, 2, 2{\cdot}1, 2{\cdot}2, 2{\cdot}2{\cdot}1, 3\}$. In the latter hedge, the symbol g stands at the position $2{\cdot}2$ and c occurs at the position $2{\cdot}2{\cdot}1$.

We write $\tilde{s}\|_I$ for the symbol which stands at position I in the hedge $\tilde{s}$. For instance $(a, f(b, g(c)), d)\|_{2\cdot 2} = g$. Given a symbol $\mathsf{s} \in \mathcal{F} \cup \mathcal{V}_{\mathsf{H}} \cup \mathcal{V}_{\mathsf{C}}$ and a hedge $\tilde{s}$ we write $pos_{\mathsf{s}}(\tilde{s})$ for the set $\{I \mid \tilde{s}\|_I = \mathsf{s}, I \in pos(\tilde{s})\}$ of all occurrences of s in $\tilde{s}$.

Definition 6 (Horizontal consecutive). Two occurrences of symbols $\mathsf{s}_1, \mathsf{s}_2 \in \mathcal{F} \cup \mathcal{V}_{\mathsf{H}} \cup \mathcal{V}_{\mathsf{C}}$ of a hedge are *horizontal consecutive* if the corresponding positions $I_{\mathsf{s}_1} \cdot i_{\mathsf{s}_1}$ and $I_{\mathsf{s}_2} \cdot i_{\mathsf{s}_2}$ are in the relation $I_{\mathsf{s}_1} = I_{\mathsf{s}_2}$ and $i_{\mathsf{s}_1} + 1 = i_{\mathsf{s}_2}$.

Definition 7 (Vertical chain). Two occurrences of symbols $\mathsf{s}_1, \mathsf{s}_2 \in \mathcal{F} \cup \mathcal{V}_{\mathsf{H}} \cup \mathcal{V}_{\mathsf{C}}$ of a hedge $\tilde{s}$ are in a *vertical chain* if their positions I_{s_1} and I_{s_2} are in the relation $I_{\mathsf{s}_1} \cdot 1 = I_{\mathsf{s}_2}$ and $I_{\mathsf{s}_1} \cdot 2 \notin pos(\tilde{s})$.

Example 5. For example, in $\tilde{s} = (a, f(X(a, b)))$, the occurrence of a at position 1 and the occurrence of f at 2 are horizontal consecutive, as well as a at $2{\cdot}1{\cdot}1$ and b at $2{\cdot}1{\cdot}2$. The occurrence of f at 2 and the occurrence of X at $2{\cdot}1$ are in vertical chain because $2{\cdot}2 \notin pos(\tilde{s}) = \{1, 2, 2{\cdot}1, 2{\cdot}1{\cdot}1, 2{\cdot}1{\cdot}2\}$, while the occurrence of X at $2{\cdot}1$ and the occurrence of a at $2{\cdot}1{\cdot}1$ are *not* because $2{\cdot}1{\cdot}2 \in pos(\tilde{s})$.

Intuitively, two occurrences of symbols $\mathsf{s}_1, \mathsf{s}_2 \in \mathcal{F} \cup \mathcal{V}_\mathsf{H} \cup \mathcal{V}_\mathsf{C}$ are in a vertical chain, if s_1 is applied to a *term* $\mathsf{s}_2(\tilde{q})$ where $\tilde{q}$ is an arbitrary hedge. If the symbol s_1 is applied to a hedge like $(\tilde{q}_1, \mathsf{s}_2(\tilde{q}), \tilde{q}_2)$ with $|\tilde{q}_1, \mathsf{s}_2(\tilde{q}), \tilde{q}_2| > 1$, then it is *not* considered a vertical chain, since it can be seen as a branching in the tree representation of the term.

With $<$ we denote the (strict) *lexicographic ordering* and with $\sqsubset$ the (strict) *ancestor relation* on positions, e.g., $1{\cdot}2{\cdot}1 < 1{\cdot}2{\cdot}2$, $1{\cdot}2{\cdot}1 < 1{\cdot}2{\cdot}1{\cdot}2$, and $1{\cdot}2{\cdot}1 \sqsubset 1{\cdot}2{\cdot}1{\cdot}2$. The relation $\sqsubseteq$ is defined as $\sqsubset \cup =$.

Definition 8 (Triangle relation). Given three positions I_1, I_2 and I_3, the ternary relation $\bowtie$ is defined as

$$I_1 \bowtie_{I_3} I_2 :\Longleftrightarrow \text{there is } I_4 \neq \lambda \text{ such that } I_4 \sqsubset I_1 \text{ and } I_4 \sqsubset I_2 \text{ and } I_4 \not\sqsubseteq I_3 \text{ and } I_1, I_2, I_3 \text{ are pairwise not in } \sqsubseteq .$$

This relation tests whether I_1 and I_2 have a common ancestor which is not an ancestor of I_3. None of these positions should be an ancestor of another.

Example 6. For instance, $1{\cdot}1 \bowtie_2 1{\cdot}2$, but *neither* $1 \bowtie_3 2$, nor $1{\cdot}1 \bowtie_2 1{\cdot}1{\cdot}2$, nor $1{\cdot}1 \bowtie_2 1{\cdot}1$, nor $1{\cdot}1 \bowtie_{1{\cdot}3} 1{\cdot}2$. A real world example of this relation would be two sisters and one of their uncles.

3. The Skeletons: Admissible Alignments

In this section we introduce the concept of admissible alignments, which are used later as skeletons to compute corresponding generalizations. For simplicity, we formulate all the notions and the algorithm for two hedges. The extension to more hedges is straightforward. Hedges to be generalized are assumed to be variable disjoint.

Given two input strings of symbols, an alignment is defined in [18] as a common subsequence of the input strings which is coupled with the information of the respective positions of the symbol occurrences. Afterwards, they define a rigidity function as a function that returns a set of alignments for two given strings of symbols. It is suggested to compute a generalization by recursively applying the rigidity function to two strings of top function symbols for two hedges, while decomposing and going deeper into the terms. This approach does not work for higher-order generalization. Therefore we suggest to extend the notion of an alignment such that it can contain common function symbols at different levels of two given input hedges. Furthermore we suggest to decouple the skeleton computation from the computation of the generalization.

In [18], the skeleton is computed during the computation of the generalization by recursively applying a given rigidity function.

Definition 9 (Alignment). Given two hedges $\tilde{s}$ and $\tilde{q}$, an *alignment* is a sequence of the form $a_1\langle I_1, J_1\rangle \dots a_m\langle I_m, J_m\rangle$ such that:

- $I_1 < \dots < I_m$ and $J_1 < \dots < J_m$, and
- for all $1 \leqslant k \leqslant m$ holds $a_k = \tilde{s}\|_{I_k} = \tilde{q}\|_{J_k}$.

An alignment represents common function symbols inside of two hedges with the corresponding positions, respecting the ordering $<$. It is a common subsequence of the word representation of those hedges with some additional information about the positions. The length of an alignment $\mathfrak{a}$ is the number of elements in it and we write $|\mathfrak{a}|$. The empty alignment is denoted by $\mathfrak{e}$.

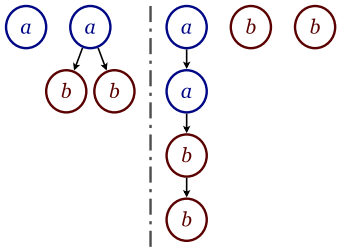

Figure 2: $\tilde{s} = (a, a(b, b))$ and $\tilde{q} = (a(a(b(b))), b, b)$.

Example 7. The two hedges $\tilde{s}$ and $\tilde{q}$ from Figure 2 have many different alignments. $a\langle 1,1\rangle a\langle 2,1{\cdot}1\rangle b\langle 2{\cdot}1, 1{\cdot}1{\cdot}1\rangle b\langle 2{\cdot}2, 3\rangle$ and $a\langle 1, 1{\cdot}1\rangle b\langle 2{\cdot}1, 2\rangle b\langle 2{\cdot}2, 3\rangle$ and $a\langle 2, 1\rangle$ are three of them, while $b\langle 2{\cdot}2, 1{\cdot}1{\cdot}1\rangle a\langle 1, 1\rangle$ is not an alignment.

Definition 10 (Collision). *Collisions* in an alignment $\mathfrak{a}$ of two hedges are defined as follows:

- A collision appears at two elements $a_k\langle I_k, J_k\rangle$, $a_l\langle I_l, J_l\rangle$ of $\mathfrak{a}$ if either ($I_k \sqsubset I_l$ and $J_k \sqsupset J_l$) or ($I_k \sqsupset I_l$ and $J_k \sqsubset J_l$).
- A collision appears at three elements $a_k\langle I_k, J_k\rangle$, $a_l\langle I_l, J_l\rangle$, $a_n\langle I_n, J_n\rangle$ of $\mathfrak{a}$ if $I_k \bowtie_{I_n} I_l$ and $J_l \bowtie_{J_k} J_n$.

Example 8. For instance, the alignment $f\langle 2, 1\rangle b\langle 2{\cdot}1, 1{\cdot}2\rangle c\langle 2{\cdot}2, 2\rangle$ of the hedges from Figure 3 contains a collision at the two elements $f\langle 2, 1\rangle$ and $c\langle 2{\cdot}2, 2\rangle$. The alignment $a\langle 1, 1{\cdot}1\rangle b\langle 2{\cdot}1, 1{\cdot}2\rangle c\langle 2{\cdot}2, 2\rangle$ of the same hedges has a collision at its three elements.

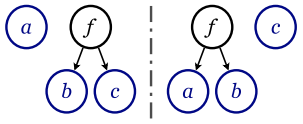

Figure 3: $(a, f(b, c))$ and $(f(a, b), c)$.

Definition 11 (Admissible alignment). An alignment of two hedges is called *admissible* if there are no collisions in it.

Note that for any two elements $a_k\langle I_k, J_k\rangle$ and $a_l\langle I_l, J_l\rangle$ of an admissible alignment, $I_k < I_l$ iff $J_k < J_l$ and $I_k \sqsubset I_l$ iff $J_k \sqsubset J_l$.

Example 9. For instance $a\langle 1, 1\rangle a\langle 2, 1{\cdot}1\rangle b\langle 2{\cdot}1, 1{\cdot}1{\cdot}1\rangle b\langle 2{\cdot}2, 3\rangle$ is a non-admissible alignment of the hedges from Figure 2, while the alignments $a\langle 1, 1{\cdot}1\rangle b\langle 2{\cdot}1, 2\rangle b\langle 2{\cdot}2, 3\rangle$ and $a\langle 2, 1\rangle b\langle 2{\cdot}2, 1{\cdot}1{\cdot}1{\cdot}1\rangle$ are admissible.

Definition 12 (Distinct alignment renaming). Given an alignment $\mathfrak{a} = a_1\langle I_1, J_1\rangle \dots a_m\langle I_m, J_m\rangle$ of two hedges $\tilde{s}$ and $\tilde{q}$, and a sequence of pairwise distinct (fresh) symbols $\acute{a}_1, \dots, \acute{a}_m$ which occur neither in $\tilde{s}$ nor in $\tilde{q}$. A *distinct alignment renaming* is a renaming of all the symbols in $\mathfrak{a}$ by the fresh ones $\acute{a}_1\langle I_1, J_1\rangle \dots \acute{a}_m\langle I_m, J_m\rangle$. Furthermore, the occurrences $\tilde{s}\|_{I_k}$ and $\tilde{q}\|_{J_k}$ of the symbol a_k are replaced by $\acute{a}_k$, for all $1 \leqslant k \leqslant m$.

Notice that a distinct alignment renaming does not impose a loss of generality. One can simply maintain the mapping $\{\acute{a}_1 \mapsto a_1, \dots, \acute{a}_m \mapsto a_m\}$ and restore the original function symbols at any time. For a given hedge $\tilde{s}$, we denote this symbol renaming by $\tilde{s}\{\acute{a}_1 \mapsto a_1, \dots, \acute{a}_m \mapsto a_m\}$.

Admissible alignments are related to generalization by the following theorem:

Theorem 1. *Consider an alignment $\mathfrak{a}$ of two hedges $\tilde{s}_1$ and $\tilde{q}_1$. Let $\acute{a}_1\langle I_1, J_1\rangle \dots \acute{a}_m\langle I_m, J_m\rangle$, $\tilde{s}_2$, and $\tilde{q}_2$ be a distinct alignment renaming of $\mathfrak{a}$, $\tilde{s}_1$, and $\tilde{q}_1$, by the fresh symbols $\acute{a}_1, \dots, \acute{a}_m$. The given alignment $\mathfrak{a}$ is admissible iff there exists a generalization $\tilde{g}$ of $\tilde{s}_2$ and $\tilde{q}_2$ with $\mathcal{F}(\tilde{g}) = \{\acute{a}_1, \dots, \acute{a}_m\}$.*

Notice that in Theorem 1 the alignment $\mathfrak{a}$ is admissible iff $\acute{a}_1\langle I_1, J_1\rangle \dots \acute{a}_m\langle I_m, J_m\rangle$ is admissible, because collisions only depend on the positions and not on the function symbols.

Proof. Let $\mathfrak{a} = a_1\langle I_1, J_1\rangle \dots a_m\langle I_m, J_m\rangle$ be an alignment of $\tilde{s}$ and $\tilde{q}$ such that for all $1 \leqslant k \leqslant m$ the function symbol a_k is unique in $\tilde{s}$ and unique in $\tilde{q}$.

($\Leftarrow$) Assume $\tilde{g}$ is a generalization of $\tilde{s}$ and $\tilde{q}$ with $\mathcal{F}(\tilde{g}) = \{a_1, \dots, a_m\}$. We will prove by contradiction that there are no collisions in $\mathfrak{a}$ (see definition of admissible alignment). Furthermore, we assume that there are at least two elements in $\mathfrak{a}$ because the other cases are trivial by definition.

Case 1: Assume there is a collision at two elements of $\mathfrak{a}$. Then there exist $a_i, a_j \in \{a_1, \dots, a_m\}$ such that a_i is an ancestor of a_j in $\tilde{s}$, while it is not an ancestor of a_j in $\tilde{q}$. We know that $\tilde{g}$ contains both symbols a_i and a_j.

Case 1.1: a_i is an ancestor of a_j in $\tilde{g}$. Then we have $a_i(\tilde{r}_1, t, \tilde{r}_2)$ being a subterm of $\tilde{g}$, where t is the term which contains a_j, and $\tilde{r}_1, \tilde{r}_2$ are arbitrary hedges. By assumption, there exists a substitution σ with $a_i, a_j \notin \mathcal{F}(\mathrm{Ran}(\sigma))$ such that a_i is not an ancestor of a_j in $\tilde{g}\sigma$. However, by the rule of substitution application $a_i(\tilde{r}_1, t, \tilde{r}_2)\sigma = a_i(\tilde{r}_1\sigma, t\sigma, \tilde{r}_2\sigma)$ the ancestor-descendant relation is preserved, which is a contradiction.

Case 1.2: a_i is not an ancestor of a_j in $\tilde{g}$. Then we have $(\tilde{r}_1, t_1, \tilde{r}_2, t_2, \tilde{r}_3)$ being a subhedge of $\tilde{g}$, where t_1 is the term which contains a_i, t_2 is the term which contains a_j, and $\tilde{r}_1, \tilde{r}_2, \tilde{r}_3$ are arbitrary hedges. By assumption, there exists a substitution σ with $a_i, a_j \notin \mathcal{F}(\mathrm{Ran}(\sigma))$ such that a_i is an ancestor of a_j in $\tilde{g}\sigma$, but this contradicts the rule of substitution application $(\tilde{r}_1, t_1, \tilde{r}_2, t_2, \tilde{r}_3)\sigma = (\tilde{r}_1\sigma, t_1\sigma, \tilde{r}_2\sigma, t_2\sigma, \tilde{r}_3\sigma)$ again.

Case 2: A collision appears at three elements. Let a_i, a_j, a_k be those elements. Without loss of generality, assume that a_i, a_j have a common ancestor ϕ that is not an ancestor of a_k in $\tilde{s}$ and let a_j, a_k have a common ancestor ψ that is not an ancestor of a_i in $\tilde{q}$. By assumption, $\tilde{g}$ contains all

three symbols exactly once. It follows that there are substitutions σ_1, σ_2 with $a_i, a_j, a_k \notin \mathcal{F}(\mathrm{Ran}(\sigma_1) \cup \mathrm{Ran}(\sigma_2))$, where $\tilde{g}\sigma_1 = \tilde{s}$ and $\tilde{g}\sigma_2 = \tilde{q}$. By assumption, we know that $\tilde{g}\sigma_1$ contains a subhedge $(t_{ij}, \tilde{s}_k)$, with t_{ij} being the term that contains the symbols ϕ, a_i, a_j, and $\tilde{s}_k$ being a hedge that contains the symbol a_k. This implies that $\tilde{g}$ contains either ϕ or a context variable that can be instantiated to introduce ϕ. It follows that $\tilde{g}$ also contains a subhedge $(t'_{ij}, \tilde{s}'_k)$, with t'_{ij} being the term that contains the symbols a_i, a_j, and $\tilde{s}'_k$ being a hedge that contains the symbol a_k. Similarly, $\tilde{g}\sigma_2$ contains a subhedge $(\tilde{q}_i, t_{jk})$, with $\tilde{q}_i$ being a hedge that contains the symbol a_i, and t_{jk} being the term that contains the symbols ψ, a_j, a_k. Further on, $\tilde{g}$ either contains ψ or a context variable, say X, which can be instantiated to introduce ψ. Let us call this metavariable χ. As ψ is an ancestor of both, a_j and a_k in $\tilde{q}$, χ has to be above t'_{ij}. This is a contradiction to the assumption that ψ is not an ancestor of a_i in $\tilde{q}$.

($\Rightarrow$) Proof by construction of an algorithm which computes such a generalization for a given admissible alignment of two hedges. In section 4 we describe this algorithm and prove its properties. □

Notice that in Theorem 1, by restoring the original symbol names in $\tilde{g}$, one obtains a generalization $\tilde{h}$ of the given input hedges.

Definition 13 (Supporting generalization). Consider an alignment $\mathfrak{a} = a_1\langle I_1, J_1\rangle \ldots a_m\langle I_m, J_m\rangle$ of two hedges $\tilde{s}_1$ and $\tilde{q}_1$. Let $\acute{a}_1\langle I_1, J_1\rangle \ldots \acute{a}_m\langle I_m, J_m\rangle$, $\tilde{s}_2$, and $\tilde{q}_2$ be a distinct alignment renaming of $\mathfrak{a}$, $\tilde{s}_1$, and $\tilde{q}_1$, by the fresh symbols $\acute{a}_1, \ldots, \acute{a}_m$. Then, for any generalization $\tilde{g}$ of $\tilde{s}_2$ and $\tilde{q}_2$ with $\mathcal{F}(\tilde{g}) = \{\acute{a}_1, \ldots, \acute{a}_m\}$, the generalization $\tilde{g}\{\acute{a}_1 \mapsto a_1, \ldots, \acute{a}_m \mapsto a_m\}$ of $\tilde{s}_1$ and $\tilde{q}_1$ is called a *supporting generalization* of $\tilde{s}_1$ and $\tilde{q}_1$ with respect to $\mathfrak{a}$.

Example 10. Let $\tilde{s}$ and $\tilde{q}$ be the hedges from Figure 2. Then $(x, a(y, Y(b)), z)$ is a supporting generalization of $\tilde{s}$ and $\tilde{q}$, with respect to $a\langle 2, 1\rangle\, b\langle 2{\cdot}2, 1{\cdot}1{\cdot}1{\cdot}1\rangle$, while it is *not* a supporting generalization of $\tilde{s}$ and $\tilde{q}$ with respect to $a\langle 1, 1\rangle$ $b\langle 2{\cdot}2, 1{\cdot}1{\cdot}1{\cdot}1\rangle$. The hedge $(X(a(x)), Y(b, b))$ is a supporting generalization of $\tilde{s}$ and $\tilde{q}$ with respect to $a\langle 1, 1{\cdot}1\rangle\, b\langle 2{\cdot}1, 2\rangle\, b\langle 2{\cdot}2, 3\rangle$.

Corollary 2. *For any admissible alignment of two hedges there exists a supporting generalization of those hedges with respect to the given alignment.*

Corollary 3. *For any generalization of two hedges there exists an admissible alignment of those hedges containing all the function symbols which appear in the generalization.*

4. Computing Least General Rigid Generalizations

We aim at solving the following problem:

Given: Two hedges $\tilde{s}$ and $\tilde{q}$ and their admissible alignment $\mathfrak{a}$.
Find: A least general supporting generalization $\tilde{g}$ of $\tilde{s}$ and $\tilde{q}$ with respect to $\mathfrak{a}$.

However, least general supporting generalizations might not be unique. Therefore, we are interested in a special class of supporting generalizations, which we call rigid generalizations.

Example 11. For instance, for (a, b, a) and (b, c) with the admissible alignment $b\langle 2, 1\rangle$, we have two supporting lggs (x, b, x, y) and (x, b, y, x). For $a(b(a))$ and $b(c)$ with the admissible alignment $b\langle 1{\cdot}1, 1\rangle$, we also have two supporting lggs $X(b(X(y)))$ and $X(b(Y(X())))$.

A solution for the first case of Example 11 has been suggested in [18] where the notion of rigid generalization was introduced for the first-order case. The idea is to capture the common structure of both input hedges but forbid consecutive hedge variables. By this restriction for the mentioned example we get one *rigid* lgg (x, b, y) which is a unique supporting generalization. Since we introduce context variables in addition to the first-order variables from [18], the restriction has to be extended for the "vertical direction". The following definition introduces this restrictions.

Definition 14 (Rigid generalization). Given two hedges $\tilde{s}, \tilde{q}$ and their admissible alignment $\mathfrak{a}$, a hedge $\tilde{g}$ is called a *rigid generalization* of $\tilde{s}$ and $\tilde{q}$ with respect to $\mathfrak{a}$, if $\tilde{g}$ is a supporting generalization of $\tilde{s}$ and $\tilde{q}$ with respect to $\mathfrak{a}$ such that the following conditions hold:

- There exist substitutions σ, ϑ with $\tilde{g}\sigma = \tilde{s}$ and $\tilde{g}\vartheta = \tilde{q}$ such that all the contexts in σ and ϑ are singleton contexts.
- No context variable in $\tilde{g}$ applies to the empty hedge.
- $\tilde{g}$ doesn't contain horizontal consecutive hedge variables.
- $\tilde{g}$ doesn't contain vertical chains of variables.
- $\tilde{g}$ doesn't contain context variables with a hedge variable as the first or the last argument (i.e., no subterms of the form $X(x, \dots)$ and $X(\dots, x)$).

Intuitively, the first two restrictions forbid that context variables capture horizontal disagreements of the input hedges. We want to use hedge variables for generalizing horizontal disagreements. For instance, consider the hedges $(f(b), a)$ and (b, c) with the admissible alignment $b\langle 1{\cdot}1, 1\rangle$. The three supporting generalizations $X(b)$, $(X(b), Y())$ and $(X(b), y)$ are pairwise distinct in the relation $\simeq$. Nevertheless, the latter one tells us more about the common structure. Those two restrictions are solely for the purpose of picking one (the most natural) case out of some equi-general solutions, and we provide the user some additional information about the solution we compute.

The other restrictions are needed to compute a unique supporting generalization, as discussed above.

Example 12. For instance, $X(a, b)$ is a rigid generalization of $f(g(a, b, c))$ and (a, b) with respect to $a\langle 1{\cdot}1{\cdot}1, 1\rangle b\langle 1{\cdot}1{\cdot}2, 2\rangle$, while $X(a, b, x)$ and $X(Y(a, b))$ are not rigid generalizations.

Definition 15 (Rigid lgg). A rigid generalization $\tilde{g}$ of $\tilde{s}$ and $\tilde{q}$ with respect to $\mathfrak{a}$ is called a *least general rigid generalization* (rigid lgg) of $\tilde{s}$ and $\tilde{q}$ with respect to $\mathfrak{a}$, if there is no rigid generalization $\tilde{h}$ of $\tilde{s}$ and $\tilde{q}$ with respect to $\mathfrak{a}$ which satisfies $\tilde{g} \prec \tilde{h}$.

Note that two hedges might have a supporting generalization which is less general than their rigid lgg with respect to the same admissible alignment.

Example 13. For instance, $X(a) \prec X(X(a))$ and both of them are supporting generalizations of $f(f(a))$ and $g(g(g(g(a))))$ with respect to $a\langle 1{\cdot}1{\cdot}1, 1{\cdot}1{\cdot}1{\cdot}1{\cdot}1\rangle$, but only $X(a)$ is a rigid generalization.

From now on, we concentrate on computing least general rigid generalizations of two variable-disjoint hedges with respect to an admissible alignment.

Definition 16 (AUP). An *anti-unification problem* (AUP) is a triple of the form $x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d};\ \mathfrak{a}$, where

- x is a hedge variable and $\tilde{s}, \tilde{q}$ are hedges,
- X is a context variable and $\tilde{c}, \tilde{d}$ are contexts,
- $\mathfrak{a}$ is an admissible alignment of $\tilde{s}$ and $\tilde{q}$.

We present our anti-unification algorithm as a rule-based algorithm that works on triples $P;\ S;\ \sigma$, where

- the problem set P is a set of AUPs,
- the store S is a set of AUPs with empty alignments,
- σ is a substitution which holds the generalization computed so far,
- for all pairs of AUPs $\{x\colon \tilde{s}_1 \triangleq \tilde{q}_1;\ X\colon \tilde{c}_1 \triangleq \tilde{d}_1;\ \mathfrak{a}_1,\ y\colon \tilde{s}_2 \triangleq \tilde{q}_2;\ Y\colon \tilde{c}_2 \triangleq \tilde{d}_2;\ \mathfrak{a}_2\} \subseteq P \cup S$ holds $x \neq y$ and $X \neq Y$.

As all the AUPs in S have the empty alignment, we write $x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d}$ instead of $x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d};\ \mathfrak{e}$ for an AUP of S. In the rules below, we use the symbols Y, Z for fresh context variables and y, z for fresh hedge variables. The brackets $[\,]$, as before, are used for context application. The symbol $\cup$ stands for disjoint union. Furthermore, i^{--} denotes $i-1$ and i^{++} denotes $i+1$.

Spl-H: **Split Hedge**

$$\begin{aligned}
&\{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d};\ a_1\langle i_1{\cdot}I_1, j_1{\cdot}J_1\rangle \ldots a_k\langle i_k{\cdot}I_k, j_k{\cdot}J_k\rangle\\
&\quad a_{k+1}\langle i_{k+1}{\cdot}I_{k+1}, j_{k+1}{\cdot}J_{k+1}\rangle \ldots a_m\langle i_m{\cdot}I_m, j_m{\cdot}J_m\rangle\} \cup P;\ S;\ \sigma \implies\\
&\quad \{y\colon \tilde{s}|_{i_1}^{i_k} \triangleq \tilde{q}|_{j_1}^{j_k};\ Y\colon \circ \triangleq \circ;\ a_1\langle (i_1 - i_1^{--}){\cdot}I_1, (j_1 - j_1^{--}){\cdot}J_1\rangle \ldots\\
&\qquad a_k\langle (i_k - i_1^{--}){\cdot}I_k, (j_k - j_1^{--}){\cdot}J_k\rangle\} \cup\\
&\quad \{z\colon \tilde{s}|_{i_k^{++}}^{i_m} \triangleq \tilde{q}|_{j_k^{++}}^{j_m};\ Z\colon \circ \triangleq \circ;\ a_{k+1}\langle (i_{k+1} - i_k){\cdot}I_{k+1}, (j_{k+1} - j_k){\cdot}J_{k+1}\rangle \ldots\\
&\qquad a_m\langle (i_m - i_k){\cdot}I_m, (j_m - j_k){\cdot}J_m\rangle\} \cup P;\\
&\quad \{x\colon \epsilon \triangleq \epsilon;\ X\colon \tilde{c}[\tilde{s}|_1^{i_1^{--}}, \circ, \tilde{s}|_{i_m^{++}}^{|\tilde{s}|}] \triangleq \tilde{d}[\tilde{q}|_1^{j_1^{--}}, \circ, \tilde{q}|_{j_m^{++}}^{|\tilde{q}|}]\} \cup S;\ \sigma\{x \mapsto (Y(y), Z(z))\},
\end{aligned}$$

If $i_1 \neq i_{k+1}$ and $j_1 \neq j_{k+1}$, and, moreover, $i_1 = i_k$ or $j_1 = j_k$, for $1 \leqslant k < m$.

Abs-L: **Abstract Left Context**

$\{x\colon (\tilde{s}_l, \phi(\tilde{s}), \tilde{s}_r) \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d};\ a_1\langle i{\cdot}I_1, J_1\rangle \dots a_m\langle i{\cdot}I_m, J_m\rangle\} \cup P;\ S;\ \sigma \Longrightarrow$
$\{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c}[\tilde{s}_l, \phi(\circ), \tilde{s}_r] \triangleq \tilde{d};\ a_1\langle I_1, J_1\rangle \dots a_m\langle I_m, J_m\rangle\} \cup P;\ S;\ \sigma,$

where $I_1 \neq \lambda$, $\phi(\tilde{s}) = (\tilde{s}_l, \phi(\tilde{s}), \tilde{s}_r)|_i$, and $\tilde{s}_l, \tilde{s}_r$ are hedges.

Abs-R: **Abstract Right Context**

$\{x\colon \tilde{s} \triangleq (\tilde{q}_l, \phi(\tilde{q}), \tilde{q}_r);\ X\colon \tilde{c} \triangleq \tilde{d};\ a_1\langle I_1, j{\cdot}J_1\rangle \dots a_m\langle I_m, j{\cdot}J_m\rangle\} \cup P;\ S;\ \sigma \Longrightarrow$
$\{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d}[\tilde{q}_l, \phi(\circ), \tilde{q}_r];\ a_1\langle I_1, J_1\rangle \dots a_m\langle I_m, J_m\rangle\} \cup P;\ S;\ \sigma,$

where $J_1 \neq \lambda$, $\phi(\tilde{q}) = (\tilde{q}_l, \phi(\tilde{q}), \tilde{q}_r)|_j$, and $\tilde{q}_l, \tilde{q}_r$ are hedges.

App-A: **Apply Alignment**

$\{x\colon (\tilde{s}_l, a_1(\tilde{s}), \tilde{s}_r) \triangleq (\tilde{q}_l, a_1(\tilde{q}), \tilde{q}_r);\ X\colon \tilde{c} \triangleq \tilde{d};$
$a_1\langle i, j\rangle a_2\langle i{\cdot}I_2, j{\cdot}J_2\rangle \dots a_m\langle i{\cdot}I_m, j{\cdot}J_m\rangle\} \cup P;\ S;\ \sigma \Longrightarrow$
$\{y\colon \tilde{s} \triangleq \tilde{q};\ Y\colon \circ \triangleq \circ;\ a_2\langle I_2, J_2\rangle \dots a_m\langle I_m, J_m\rangle\} \cup P;$
$\{x\colon \epsilon \triangleq \epsilon;\ X\colon \tilde{c}[\tilde{s}_l, \circ, \tilde{s}_r] \triangleq \tilde{d}[\tilde{q}_l, \circ, \tilde{q}_r]\} \cup S;\ \sigma\{x \mapsto a_1(Y(y))\},$

where $a_1(\tilde{s}), a_1(\tilde{q})$ are the terms at the positions i, j and $\tilde{s}_l, \tilde{s}_r, \tilde{q}_l, \tilde{q}_r$ are hedges.

Sol-H: **Solve Hedge**

$\{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \circ \triangleq \circ;\ \mathfrak{e}\} \cup P;\ S;\ \sigma \Longrightarrow P;\ \{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \circ \triangleq \circ\} \cup S;\ \sigma\{X \mapsto \circ\}.$

Res-C: **Restore Context**

$P;\ \{x\colon \epsilon \triangleq \epsilon;\ X\colon (\tilde{s}_l, \dot{c}, \tilde{s}_r) \triangleq (\tilde{q}_l, \dot{d}, \tilde{q}_r)\} \cup S;\ \sigma \Longrightarrow$
$P;\ \{x\colon \epsilon \triangleq \epsilon;\ X\colon \dot{c} \triangleq \dot{d},\ y\colon \tilde{s}_l \triangleq \tilde{q}_l;\ Y\colon \circ \triangleq \circ,\ z\colon \tilde{s}_r \triangleq \tilde{q}_r;\ Z\colon \circ \triangleq \circ\} \cup S;$
$\sigma\{X \mapsto (y, X(\circ), z)\},$

if not $\epsilon = \tilde{s}_l = \tilde{s}_r = \tilde{q}_l = \tilde{q}_r$. $\dot{c}, \dot{d}$ are singleton contexts.

Mer-S: **Merge Store**

$P;\ \{x_1\colon \tilde{s} \triangleq \tilde{q};\ X_1\colon \tilde{c} \triangleq \tilde{d},\ x_2\colon \tilde{s} \triangleq \tilde{q};\ X_2\colon \tilde{c} \triangleq \tilde{d}\} \cup S;\ \sigma \Longrightarrow$
$P;\ \{x_1\colon \tilde{s} \triangleq \tilde{q};\ X_1\colon \tilde{c} \triangleq \tilde{d}\} \cup S;\ \sigma\{x_2 \mapsto x_1, X_2 \mapsto X_1\}.$

Clr-S: **Clear Store**

$P;\ \{x\colon \epsilon \triangleq \epsilon;\ X\colon \circ \triangleq \circ\} \cup S;\ \sigma \Longrightarrow P;\ S;\ \sigma\{x \mapsto \epsilon, X \mapsto \circ\}.$

The idea of the store is to keep track of already solved AUPs in order to generalize the same AUPs in the same way, as it is illustrated in the Mer-S rule.

To compute generalizations of $\tilde{s}$ and $\tilde{q}$ with respect to an admissible alignment $\mathfrak{a}$, the procedure starts with $\{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \circ \triangleq \circ;\ \mathfrak{a}\};\ \varnothing;\ \varepsilon$, where x and X are fresh variables, and applies the rules exhaustively. We denote this procedure by $\mathfrak{G}$. The intuition is that at i's step of such a derivation, $X(x)\sigma_i$ is supposed to be a generalization of $\tilde{s}$ and $\tilde{q}$, with the idea that when the process stops with σ in the last step, then $X(x)\sigma$ is a rigid lgg of $\tilde{s}$ and $\tilde{q}$ with respect to $\mathfrak{a}$.

4.1. Explanation of the Rules

Before discussing the properties of $\mathfrak{G}$, we briefly explain informally what the rules do. At each step, each AUP $x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d};\ \mathfrak{a}$ in P represents the

hedges $\tilde{c}[\tilde{s}]$ and $\tilde{d}[\tilde{q}]$ which are to be generalized, such that the final generalization contains the function symbols from $\mathfrak{a}$. They are split according to the occurrences of alignment elements: All symbols from $\mathfrak{a}$ are in $\tilde{s}$ and $\tilde{q}$. None of them appear in $\tilde{c}$ and $\tilde{d}$.

Such an AUP can be transformed by one of the first four rules: Spl-H, Abs-L, Abs-R, or App-A. The eventual goal of these transformations is to reach the occurrences of the first alignment element in $\tilde{s}$ and $\tilde{q}$. In the course of the transformation, $\tilde{c}$ and $\tilde{d}$ are getting extended with contexts above those occurrences.

Spl-H. When the symbols in $\mathfrak{a}$ are distributed in more than one term both in $\tilde{s}$ and in $\tilde{q}$, then we use the Spl-H rule to select subhedges of $\tilde{s}$ and $\tilde{q}$ which contain all the alignment elements. (The other parts of $\tilde{s}$ and $\tilde{q}$ are moved to the store, since they will not contribute a symbol to the generalization.) Furthermore, by this rule, each of these subhedges are split into two smaller subhedges: From the $\tilde{s}$ side these are $\tilde{s}|_{i_1}^{i_k}$ and $\tilde{s}|_{i_k^{++}}^{i_m}$, and from the $\tilde{q}$ side they are $\tilde{q}|_{j_1}^{j_k}$ and $\tilde{q}|_{j_k^{++}}^{j_m}$. The split point k is decided by the following criteria:

- $\tilde{s}|_{i_1}^{i_k}$ and $\tilde{q}|_{j_1}^{j_k}$ contain the first $k > 0$ elements of $\mathfrak{a}$.
- $\tilde{s}|_{i_k^{++}}^{i_m}$ and $\tilde{q}|_{j_k^{++}}^{j_m}$ contain the elements of $\mathfrak{a}$ starting from $k+1$. There exists at least one such element.
- $\tilde{s}|_{i_1}^{i_k}$ or $\tilde{q}|_{j_1}^{j_k}$ is a term (a singleton hedge), and the $k+1$'st element of $\mathfrak{a}$ does not belong to it.

The process will continue by generalizing $\tilde{s}|_{i_1}^{i_k}$ and $\tilde{q}|_{j_1}^{j_k}$ with respect to the first k-element prefix of $\mathfrak{a}$, and generalizing $\tilde{s}|_{i_k^{++}}^{i_m}$ and $\tilde{q}|_{j_k^{++}}^{j_m}$ with respect to the elements of $\mathfrak{a}$ starting from $k+1$. Note that in the next step Spl-H is not applicable to the AUP with $\tilde{s}|_{i_1}^{i_k}$ and $\tilde{q}|_{j_1}^{j_k}$. This is because at least one of them is a single term which completely contains the alignment elements. Therefore either Abs-L, Abs-R, or App-A applies.

Example 14. Consider the hedges $(g(a), f(a, g(b)), c, g(b), e)$ and $(e, e, h(a, e), f(b), a, c, d, b)$ and the admissible alignment $a\langle 2{\cdot}1, 3{\cdot}1\rangle b\langle 2{\cdot}2{\cdot}1, 4{\cdot}1\rangle c\langle 3, 6\rangle b\langle 4{\cdot}1, 8\rangle$ of them.

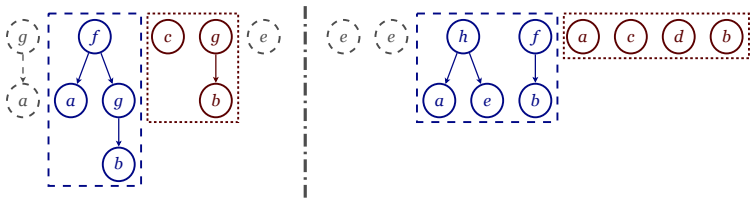

Figure 4: The hedges from Example 14.

The dashed nodes in Figure 4 denote the parts which are moved into the store. The dashed rectangle denotes $\tilde{s}|_{i_1}^{i_k}$ and $\tilde{q}|_{j_1}^{j_k}$ and the dotted one $\tilde{s}|_{i_k^{++}}^{i_m}$ and $\tilde{q}|_{j_k^{++}}^{j_m}$.

Abs-L, Abs-R. When all symbols in $\mathfrak{a}$ belong to one term in $\tilde{s}$ or in $\tilde{q}$ (or maybe both), but the root of that term is *not* the symbol a_1 from the first element of $\mathfrak{a}$, then an attempt is made to get deeper to that term, to reach the subterm whose top symbol is the a_1 from $\mathfrak{a}$. This descent is carried out by Abs-L or Abs-R, depending whether we are searching for the subterm with a_1 in the top in $\tilde{s}$ or in $\tilde{q}$.

To illustrate the rule Abs-L which serves as a representative of both, we use the above example and apply Abs-L to the AUP $x\colon f(a, g(b)) \triangleq (h(a, e), f(b))$; $X\colon \circ \triangleq \circ$; $a\langle 1{\cdot}1, 1{\cdot}1\rangle b\langle 1{\cdot}2{\cdot}1, 2{\cdot}1\rangle$ following an Spl-H application. The Abs-L transformation decomposes the left term $f(a, g(b))$ into a context $f(\circ)$ and a hedge $(a, g(b))$, resulting in the AUP $x\colon (a, g(b)) \triangleq (h(a, e), f(b))$; $X\colon f(\circ) \triangleq \circ$; $a\langle 1, 1{\cdot}1\rangle b\langle 2{\cdot}1, 2{\cdot}1\rangle$. Figure 5 demonstrates this decomposition step.

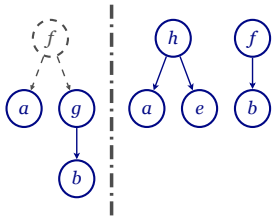

Figure 5: $f(a, g(b))$ and $(h(a, e), f(b))$.

App-A. When all symbols in $\mathfrak{a}$ belong to one term in $\tilde{s}$ and one term in $\tilde{q}$, and these terms have the same root symbol which is exactly the a_1 from the first element of $\mathfrak{a}$, then a_1 is moved to the generalization. This is what the App-A rule does. The process will continue with generalizing the hedges under the occurrences of a_1 in $\tilde{s}$ and $\tilde{q}$.

Sol-H. When the alignment is empty in $x\colon \tilde{s} \triangleq \tilde{q}$; $X\colon \tilde{c} \triangleq \tilde{d}$; $\mathfrak{e}$ in P, then the hedge there will not contribute a symbol in the generalization. Moreover, both $\tilde{c}$ and $\tilde{d}$ are holes, because only App-A can make the alignment empty, and it makes the contexts in the obtained AUP the hole. Such AUPs are considered solved, as their generalization is just x they contain. They should be put in the store, which keeps information about the differences between the hedges to be generalized. At the same time, the context variable X can be deleted, as it just stand for the hole. This is what the Sol-H rule does.

Transformation of the store. The other three rules work on the store. Clr-S removes the empty AUP from the store and eliminates the corresponding variables form the generalization. Mer-S guarantees that the same AUPs are generalized with the same variables, making sure that the same differences in the input hedges are generalized uniformly. Finally, the Res-C rule guarantees that each context variable in the generalization generalizes singleton contexts in the input hedges: A property required for rigid generalizations.

Definition 17. We define two substitutions obtained by a set S of AUPs:

$$\sigma_{\text{L}}(S) ::= \{x \mapsto \tilde{s}, X \mapsto \tilde{c} \mid x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d};\ \mathfrak{a} \in S\}$$
$$\sigma_{\text{R}}(S) ::= \{x \mapsto \tilde{q}, X \mapsto \tilde{d} \mid x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d};\ \mathfrak{a} \in S\}$$

Example 15. Let $\tilde{s} = f(a, f(b,b))$ and $\tilde{q} = (b, f(a,b), b)$ be the input hedges with the admissible alignment $\mathfrak{a} = f\langle 1,2\rangle a\langle 1{\cdot}1, 2{\cdot}1\rangle b\langle 1{\cdot}2{\cdot}1, 2{\cdot}2\rangle$. We illustrate how the algorithm $\mathfrak{G}$ exhaustively transforms the initial system to compute the rigid lgg for $\tilde{s}$, $\tilde{q}$, and $\mathfrak{a}$. In the substitution, we only keep the mappings for the two generalization variables x and X of the initial AUP.

$$\begin{aligned}
&\{x\colon f(a, f(b,b)) \triangleq (b, f(a,b), b);\ X\colon \circ \triangleq \circ;\ f\langle 1,2\rangle a\langle 1{\cdot}1, 2{\cdot}1\rangle b\langle 1{\cdot}2{\cdot}1, 2{\cdot}2\rangle\};\ \varnothing;\ \varepsilon\\
\Rightarrow_{\mathsf{App\text{-}A}}\ &\{y_1\colon (a, f(b,b)) \triangleq (a,b);\ Y_1\colon \circ \triangleq \circ;\ a\langle 1,1\rangle b\langle 2{\cdot}1, 2\rangle\};\\
&\{x\colon \epsilon \triangleq \epsilon;\ X\colon \circ \triangleq (b, \circ, b)\};\ \{x \mapsto f(Y_1(y_1))\}\\
\Rightarrow^{\mathsf{Res\text{-}C}}_{\mathsf{Clr\text{-}S}}\ &\{y_1\colon (a, f(b,b)) \triangleq (a,b);\ Y_1\colon \circ \triangleq \circ;\ a\langle 1,1\rangle b\langle 2{\cdot}1, 2\rangle\};\\
&\{z_1\colon \epsilon \triangleq b;\ Z_1\colon \circ \triangleq \circ, z_2\colon \epsilon \triangleq b;\ Z_2\colon \circ \triangleq \circ\};\ \{x \mapsto f(Y_1(y_1)), X \mapsto (z_1, \circ, z_2)\}\\
\Rightarrow_{\mathsf{Mer\text{-}S}}\ &\{y_1\colon (a, f(b,b)) \triangleq (a,b);\ Y_1\colon \circ \triangleq \circ;\ a\langle 1,1\rangle b\langle 2{\cdot}1, 2\rangle\};\\
&\{z_1\colon \epsilon \triangleq b;\ Z_1\colon \circ \triangleq \circ\};\ \{x \mapsto f(Y_1(y_1)), X \mapsto (z_1, \circ, z_1)\}\\
\Rightarrow^{\mathsf{Spl\text{-}H}}_{\mathsf{Clr\text{-}S}}\ &\{y_2\colon a \triangleq a;\ Y_2\colon \circ \triangleq \circ;\ a\langle 1,1\rangle,\ y_3\colon (f(b,b)) \triangleq b;\ Y_3\colon \circ \triangleq \circ;\ b\langle 1{\cdot}1, 1\rangle\};\\
&\{z_1\colon \epsilon \triangleq b;\ Z_1\colon \circ \triangleq \circ\};\ \{x \mapsto f(Y_2(y_2), Y_3(y_3)), X \mapsto (z_1, \circ, z_1)\}\\
\Rightarrow^{\mathsf{App\text{-}A}}_{\mathsf{Clr\text{-}S}}\ &\{y_4\colon \epsilon \triangleq \epsilon;\ Y_4\colon \circ \triangleq \circ;\ \mathfrak{e},\ y_3\colon (f(b,b)) \triangleq (b);\ Y_3\colon \circ \triangleq \circ;\ b\langle 1{\cdot}1, 1\rangle\};\\
&\{z_1\colon \epsilon \triangleq b;\ Z_1\colon \circ \triangleq \circ\};\ \{x \mapsto f(a(Y_4(y_4)), Y_3(y_3)), X \mapsto (z_1, \circ, z_1)\}\\
\Rightarrow^{\mathsf{Sol\text{-}H}}_{\mathsf{Clr\text{-}S}}\ &\{y_3\colon f(b,b) \triangleq b;\ Y_3\colon \circ \triangleq \circ;\ b\langle 1{\cdot}1, 1\rangle\};\\
&\{z_1\colon \epsilon \triangleq b;\ Z_1\colon \circ \triangleq \circ\};\ \{x \mapsto f(a, Y_3(y_3)), X \mapsto (z_1, \circ, z_1)\}\\
\Rightarrow_{\mathsf{Abs\text{-}L}}\ &\{y_3\colon (b,b) \triangleq b;\ Y_3\colon f(\circ) \triangleq \circ;\ b\langle 1,1\rangle\};\\
&\{z_1\colon \epsilon \triangleq b;\ Z_1\colon \circ \triangleq \circ\};\ \{x \mapsto f(a, Y_3(y_3)), X \mapsto (z_1, \circ, z_1)\}\\
\Rightarrow_{\mathsf{App\text{-}A}}\ &\{y_5\colon \epsilon \triangleq \epsilon;\ Y_5\colon \circ \triangleq \circ;\ \mathfrak{e}\};\ \{y_3\colon \epsilon \triangleq \epsilon;\ Y_3\colon f(\circ, b) \triangleq \circ,\ z_1\colon \epsilon \triangleq b;\ Z_1\colon \circ \triangleq \circ\};\\
&\{x \mapsto f(a, Y_3(b(Y_5(y_5)))), X \mapsto (z_1, \circ, z_1)\}\\
\Rightarrow^{\mathsf{Sol\text{-}H}}_{\mathsf{Clr\text{-}S}}\ &\varnothing;\{y_3\colon \epsilon \triangleq \epsilon;\ Y_3\colon f(\circ, b) \triangleq \circ,\ z_1\colon \epsilon \triangleq b;\ Z_1\colon \circ \triangleq \circ\};\\
&\{x \mapsto f(a, Y_3(b)), X \mapsto (z_1, \circ, z_1)\}.
\end{aligned}$$

$X(x)\sigma = (z_1, f(a, Y_3(b)), z_1)$ generalizes $\tilde{s}$ and $\tilde{q}$ with respect to $\mathfrak{a}$. From the store S we can read $\sigma_{\text{L}}(S) = \{z_1 \mapsto \epsilon, Y_3 \mapsto f(\circ, b), \ldots\}$ and $\sigma_{\text{R}}(S) = \{z_1 \mapsto b, Y_3 \mapsto \circ, \ldots\}$. Then we have $X(x)\sigma\sigma_{\text{L}}(S) = \tilde{s}$ and $X(x)\sigma\sigma_{\text{R}}(S) = \tilde{q}$.

Example 16. We illustrate the computation of a rigid lgg with respect to the given laa $a\langle 1{\cdot}1, 1\rangle f\langle 2,2\rangle g\langle 2{\cdot}1{\cdot}1, 2{\cdot}1\rangle a\langle 2{\cdot}1{\cdot}1{\cdot}1, 2{\cdot}1{\cdot}1\rangle c\langle 2{\cdot}1{\cdot}2, 2{\cdot}2\rangle$ of the two hedges $(U(a), f(U(g(a,b,b),c),b,b))$ and $(a, f(g(a,d),c,d))$. The symbol U denotes a context variable. All the other symbols are function symbols.

$$\begin{aligned}
&\{x\colon (U(a), f(U(g(a,b,b),c),b,b)) \triangleq (a, f(g(a,d),c,d));\\
&X\colon \circ \triangleq \circ;\ a\langle 1{\cdot}1, 1\rangle f\langle 2,2\rangle g\langle 2{\cdot}1{\cdot}1, 2{\cdot}1\rangle a\langle 2{\cdot}1{\cdot}1{\cdot}1, 2{\cdot}1{\cdot}1\rangle c\langle 2{\cdot}1{\cdot}2, 2{\cdot}2\rangle\};\ \varnothing;\ \varepsilon\\
\Rightarrow^{\mathsf{Spl\text{-}H}}_{\mathsf{Clr\text{-}S}}\ &\{y_1\colon U(a) \triangleq a;\ Y_1\colon \circ \triangleq \circ;\ a\langle 1{\cdot}1, 1\rangle, z_1\colon f(U(g(a,b,b),c),b,b) \triangleq f(g(a,d),c,d);
\end{aligned}$$

$$
\begin{aligned}
& Z_1{:}\circ \triangleq \circ;\ f\langle 1,1\rangle g\langle 1{\cdot}1{\cdot}1, 1{\cdot}1\rangle a\langle 1{\cdot}1{\cdot}1{\cdot}1, 1{\cdot}1{\cdot}1\rangle c\langle 1{\cdot}1{\cdot}2, 1{\cdot}2\rangle\};\\
& \varnothing;\ \{x \mapsto (Y_1(y_1), Z_1(z_1)), X \mapsto \circ\}\\
\Rightarrow_{\mathsf{Abs\text{-}L}}\ & \{y_1{:}\, a \triangleq a;\ Y_1{:}\, U(\circ) \triangleq \circ;\ a\langle 1,1\rangle;\ z_1{:}\, f(U(g(a,b,b),c),b,b) \triangleq f(g(a,d),c,d);\\
& Z_1{:}\circ \triangleq \circ;\ f\langle 1,1\rangle g\langle 1{\cdot}1{\cdot}1, 1{\cdot}1\rangle a\langle 1{\cdot}1{\cdot}1{\cdot}1, 1{\cdot}1{\cdot}1\rangle c\langle 1{\cdot}1{\cdot}2, 1{\cdot}2\rangle\};\\
& \varnothing;\ \{x \mapsto (Y_1(y_1), Z_1(z_1)), X \mapsto \circ\}\\
\Rightarrow^{\mathsf{App\text{-}A}}_{\mathsf{Sol\text{-}H}}\ & \{z_1{:}\, f(U(g(a,b,b),c),b,b) \triangleq f(g(a,d),c,d);\\
& Z_1{:}\circ \triangleq \circ;\ f\langle 1,1\rangle g\langle 1{\cdot}1{\cdot}1, 1{\cdot}1\rangle a\langle 1{\cdot}1{\cdot}1{\cdot}1, 1{\cdot}1{\cdot}1\rangle c\langle 1{\cdot}1{\cdot}2, 1{\cdot}2\rangle\};\\
& \{y_1{:}\, \epsilon \triangleq \epsilon;\ Y_1{:}\, U(\circ) \triangleq \circ\};\ \{x \mapsto (Y_1(a), Z_1(z_1)), X \mapsto \circ\}\\
\Rightarrow^{\mathsf{App\text{-}A}}_{\mathsf{Clr\text{-}S}}\ & \{z_2{:}\, (U(g(a,b,b),c),b,b) \triangleq (g(a,d),c,d);\\
& Z_2{:}\circ \triangleq \circ;\ g\langle 1{\cdot}1, 1\rangle a\langle 1{\cdot}1{\cdot}1, 1{\cdot}1\rangle c\langle 1{\cdot}2, 2\rangle\};\\
& \{y_1{:}\, \epsilon \triangleq \epsilon;\ Y_1{:}\, U(\circ) \triangleq \circ\};\ \{x \mapsto (Y_1(a), f(Z_2(z_2))), X \mapsto \circ\}\\
\Rightarrow_{\mathsf{Abs\text{-}L}}\ & \{z_2{:}\, (g(a,b,b),c) \triangleq (g(a,d),c,d);\ Z_2{:}\, (U(\circ),b,b) \triangleq \circ;\ g\langle 1,1\rangle a\langle 1{\cdot}1, 1{\cdot}1\rangle c\langle 2,2\rangle\};\\
& \{y_1{:}\, \epsilon \triangleq \epsilon;\ Y_1{:}\, U(\circ) \triangleq \circ\};\ \{x \mapsto (Y_1(a), f(Z_2(z_2))), X \mapsto \circ\}\\
\Rightarrow^{\mathsf{Spl\text{-}H}}_{\mathsf{Clr\text{-}S}}\ & \{z_3{:}\, g(a,b,b) \triangleq g(a,d); Z_3{:}\circ \triangleq \circ; g\langle 1,1\rangle a\langle 1{\cdot}1, 1{\cdot}1\rangle, z_4{:}\, c \triangleq c; Z_4{:}\circ \triangleq \circ; c\langle 1,1\rangle\};\\
& \{y_1{:}\, \epsilon \triangleq \epsilon;\ Y_1{:}\, U(\circ) \triangleq \circ,\ z_2{:}\, \epsilon \triangleq \epsilon;\ Z_2{:}\, (U(\circ),b,b) \triangleq (\circ,d)\};\\
& \{x \mapsto (Y_1(a), f(Z_2(Z_3(z_3), Z_4(z_4)))), X \mapsto \circ\}\\
\Rightarrow^{\mathsf{App\text{-}A}}_{\mathsf{Clr\text{-}S}}\ & \{z_5{:}\, (a,b,b) \triangleq (a,d);\ Z_5{:}\circ \triangleq \circ;\ a\langle 1,1\rangle, z_4{:}\, c \triangleq c;\ Z_4{:}\circ \triangleq \circ;\ c\langle 1,1\rangle\};\\
& \{y_1{:}\, \epsilon \triangleq \epsilon;\ Y_1{:}\, U(\circ) \triangleq \circ,\ z_2{:}\, \epsilon \triangleq \epsilon;\ Z_2{:}\, (U(\circ),b,b) \triangleq (\circ,d)\};\\
& \{x \mapsto (Y_1(a), f(Z_2(g(Z_5(z_5)), Z_4(z_4)))), X \mapsto \circ\}\\
\Rightarrow^{\mathsf{Res\text{-}C}}_{\mathsf{Clr\text{-}S}}\ & \{z_5{:}\, (a,b,b) \triangleq (a,d);\ Z_5{:}\circ \triangleq \circ;\ a\langle 1,1\rangle, z_4{:}\, c \triangleq c;\ Z_4{:}\circ \triangleq \circ;\ c\langle 1,1\rangle\};\\
& \{y_1{:}\, \epsilon \triangleq \epsilon;\ Y_1{:}\, U(\circ) \triangleq \circ,\ z_2{:}\, \epsilon \triangleq \epsilon;\ Z_2{:}\, U(\circ) \triangleq \circ,\ y_2{:}\, (b,b) \triangleq d;\ Y_2{:}\circ \triangleq \circ\};\\
& \{x \mapsto (Y_1(a), f(Z_2(g(Z_5(z_5)), Z_4(z_4)), y_2)), X \mapsto \circ\}\\
\Rightarrow_{\mathsf{Mer\text{-}S}}\ & \{z_5{:}\, (a,b,b) \triangleq (a,d);\ Z_5{:}\circ \triangleq \circ;\ a\langle 1,1\rangle, z_4{:}\, c \triangleq c;\ Z_4{:}\circ \triangleq \circ;\ c\langle 1,1\rangle\};\\
& \{y_1{:}\, \epsilon \triangleq \epsilon;\ Y_1{:}\, U(\circ) \triangleq \circ,\ y_2{:}\, (b,b) \triangleq d;\ Y_2{:}\circ \triangleq \circ\};\\
& \{x \mapsto (Y_1(a), f(Y_1(g(Z_5(z_5)), Z_4(z_4)), y_2)), X \mapsto \circ\}\\
\Rightarrow^{\mathsf{App\text{-}A}}_{\mathsf{Sol\text{-}H}}\ & \{z_5{:}\, (a,b,b) \triangleq (a,d);\ Z_5{:}\circ \triangleq \circ;\ a\langle 1,1\rangle\};\ \{y_1{:}\, \epsilon \triangleq \epsilon;\ Y_1{:}\, U(\circ) \triangleq \circ,\\
& y_2{:}\, (b,b) \triangleq d;\ Y_2{:}\circ \triangleq \circ\};\ \{x \mapsto (Y_1(a), f(Y_1(g(Z_5(z_5)), c), y_2)), X \mapsto \circ\}\\
\Rightarrow^{\mathsf{App\text{-}A}}_{\mathsf{Sol\text{-}H}}\ & \varnothing;\ \{y_1{:}\, \epsilon \triangleq \epsilon;\ Y_1{:}\, U(\circ) \triangleq \circ,\ y_2{:}\, (b,b) \triangleq d;\ Y_2{:}\circ \triangleq \circ,\\
& z_5{:}\, \epsilon \triangleq \epsilon;\ Z_5{:}\, (\circ,b,b) \triangleq (\circ,d)\};\ \{x \mapsto (Y_1(a), f(Y_1(g(Z_5(a)), c), y_2)), X \mapsto \circ\}\\
\Rightarrow^{\mathsf{Res\text{-}C}}_{\mathsf{Clr\text{-}S}}\ & \varnothing;\ \{y_1{:}\, \epsilon \triangleq \epsilon;\ Y_1{:}\, U(\circ) \triangleq \circ,\ y_2{:}\, (b,b) \triangleq d;\ Y_2{:}\circ \triangleq \circ,\ y_3{:}\, (b,b) \triangleq d;\ Y_3{:}\circ \triangleq \circ\};\\
& \{x \mapsto (Y_1(a), f(Y_1(g(a,y_3),c),y_2)), X \mapsto \circ\}\\
\Rightarrow_{\mathsf{Mer\text{-}S}}\ & \varnothing;\ \{y_1{:}\, \epsilon \triangleq \epsilon;\ Y_1{:}\, U(\circ) \triangleq \circ,\ y_2{:}\, (b,b) \triangleq d;\ Y_2{:}\circ \triangleq \circ\};\\
& \{x \mapsto (Y_1(a), f(Y_1(g(a,y_2),c),y_2)), X \mapsto \circ\}
\end{aligned}
$$

$X(x)\sigma = (Y_1(a), f(Y_1(g(a,y_2),c),y_2))$ generalizes the two input hedges with respect to the given alignment and the store contains all the information about the differences of the input hedges.

Example 17. As already mentioned in the introduction, the algorithm $\mathfrak{G}$ can be a useful tool for detecting software clones. For that, we translate the input

source-codes into hedges. It is straightforward to encode abstract syntax trees, like it was proposed in [6, 11, 15], with hedges. For illustration we use an example composed from the taxonomy of editing scenarios for different clone types, described in [26]. Listing 1 shows the original code and Listing 2 shows two of its clones.

```
1                 void sumProd(int n) {
2                   float sum = 0;
3                   float prod = 1;
4                   for(int i=1;i<=n;i++)
5                     {sum = sum + i;
6                     prod = prod * i;
7                     foo(sum,prod); }}
```

Listing 1: Example code for using $\mathfrak{G}$ as clone detection tool on it.

The algorithm $\mathfrak{G}$ will reveal the first clone from Listing 2 similarly to the algorithm from [18] because no higher-order variables are needed to capture the deletion of some code statements. In the second clone, the for loop has been replaced by a while loop. Since we represent loops as function applications in the syntax tree, the generalization method suggested in this work leads to better results by introducing context variables for different function applications.

```
1  void sumProd(int n) {          | void sumProd(int n) {
2    float sum = 0;               |   float sum = 0;
3    float prod = 1;              |   float prod = 1;
4    for(int i=1;i<=n;i++) {      |   int i=1; while(i<=n) {
5      sum = sum + i;             |     sum = sum + i;
6      // line deleted            |     prod = prod * i;
7      foo(sum,prod); }}          |     foo(sum,prod); i++; }}
```

Listing 2: Two clones of the code from Listing 1.

Figure 6 shows the term encoding of the abstract syntax tree from the original code and Figure 7 shows the respective term encodings of the two clones.

$$
\begin{aligned}
&sumProd(input(type(int), n),\\
&\qquad returnType(void),\\
&\quad =(type(float), sum, 0),\\
&\quad =(type(float), prod, 1),\\
&\quad for(=(type(int), i, 1), \leqslant(i, n), {+}{+}(i),\\
&\qquad =(sum, +(sum, i)),\\
&\qquad =(prod, *(prod, i)),\\
&\qquad foo(sum, prod)))
\end{aligned}
$$

Figure 6: Representation of the original code from Listing 1 as unranked term.

$$
\begin{array}{l}
sumProd(input(type(int), n), \\
\quad returnType(void), \\
\quad =(type(float), sum, 0), \\
\quad =(type(float), prod, 1), \\
\quad for(=(type(int), i, 1), \leqslant(i, n), ++(i), \\
\quad\quad =(sum, +(sum, i)), \\
\quad\quad \\
\quad\quad foo(sum, prod)))
\end{array}
\qquad
\begin{array}{l}
sumProd(input(type(int), n), \\
\quad returnType(void), \\
\quad =(type(float), sum, 0), \\
\quad =(type(float), prod, 1), \\
\quad =(type(int), i, 1), while(\leqslant(i, n), \\
\quad\quad =(sum, +(sum, i)), \\
\quad\quad =(prod, *(prod, i)), \\
\quad\quad foo(sum, prod), ++(i)))
\end{array}
$$

Figure 7: Representation of the two clones from Listing 2 as unranked terms.

For generalizing the (term representation of the) original code and one of the clones, we compute the admissible alignment of longest length (laa). We will discuss the computation of alignments in section 6. In both of our two examples, the laa is unique. (In general this is not the case.) The length of the laa for the original code and the first clone is 36. For the original code and the second clone it contains 38 symbols. Figure 8 shows the results after applying $\mathfrak{G}$ to the term representation of the original code and either of the clones with respect to the corresponding laa.

$$
\begin{array}{l}
sumProd(input(type(int), n), \\
\quad returnType(void), \\
\quad =(type(float), sum, 0), \\
\quad =(type(float), prod, 1), \\
\quad for(=(type(int), i, 1), \leqslant(i, n), ++(i), \\
\quad\quad =(sum, +(sum, i)), \\
\quad\quad \mathbf{x} \\
\quad\quad foo(sum, prod)))
\end{array}
\qquad
\begin{array}{l}
sumProd(input(type(int), n), \\
\quad returnType(void), \\
\quad =(type(float), sum, 0), \\
\quad =(type(float), prod, 1), \\
\quad \mathbf{X}(=(type(int), i, 1), \mathbf{Y}(\leqslant(i, n), \mathbf{x}, \\
\quad\quad =(sum, +(sum, i)), \\
\quad\quad =(prod, *(prod, i)), \\
\quad\quad foo(sum, prod))))
\end{array}
$$

Figure 8: Clone detection by applying $\mathfrak{G}$ to the code from Figure 6 and its clones (Figure 7).

The first result is pretty clear. The input terms can be obtained from the generalization by instantiating the variable x either with $=(prod, *(prod, i))$ or the empty hedge. From the second generalization, the term representation of the original code can be obtained by applying the substitution $\{X \mapsto for(\circ), Y \mapsto \circ, x \mapsto ++(i)\}$, and the term representation of the second clone can be obtained by applying the substitution $\{X \mapsto \circ, Y \mapsto while(\circ, ++(i)), x \mapsto \epsilon\}$. The information about the differences is available from the store, as usual.

4.2. Properties of the Algorithm

The algorithm $\mathfrak{G}$ maintains the following four invariants. We will use them to prove soundness of the algorithm.

Lemma 4 (Invariant 1). *If* $\{x_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0;\ X_0 \colon \circ \triangleq \circ;\ \mathfrak{a}_0\};\ S_0;\ \sigma_0 \Longrightarrow^* P_n;\ S_n;\ \sigma_n$ *is a derivation in* $\mathfrak{G}$*, then for all* $x_n \colon \tilde{s}_n \triangleq \tilde{q}_n;\ X_n \colon \tilde{c}_n \triangleq \tilde{d}_n;\ \mathfrak{a}_n \in P_n$ *either* $\mathfrak{a}_n \neq \mathfrak{e}$ *or* $\tilde{c}_n = \tilde{d}_n = \circ$*.*

Proof. The only rules that reduce the length of an alignment are Spl-H and App-A. The rule Spl-H splits an AUP such that the alignments of the two new AUPs are not empty. Therefore App-A is the only rule which transforms an AUP with a nonempty alignment into one with an empty alignment. Every AUP derived by App-A has the form $x_n \colon \tilde{s}_n \triangleq \tilde{q}_n;\ X_n \colon \circ \triangleq \circ;\ \mathfrak{a}_n$. □

Lemma 5 (Invariant 2). *Let* $P_0; S_0; \sigma_0 \Longrightarrow^* P_n; S_n; \sigma_n$ *be a derivation in* $\mathfrak{G}$*. If for all* $\{x_1 \colon \tilde{s}_1 \triangleq \tilde{q}_1;\ X_1 \colon \tilde{c}_1 \triangleq \tilde{d}_1;\ \mathfrak{a}_1,\ x_2 \colon \tilde{s}_2 \triangleq \tilde{q}_2;\ X_2 \colon \tilde{c}_2 \triangleq \tilde{d}_2;\ \mathfrak{a}_2\} \subseteq P_0 \cup S_0$ *holds* $x_1 \neq x_2$ *and* $X_1 \neq X_2$*, then this implies* $x_3 \neq x_4$ *and* $X_3 \neq X_4$ *for all* $\{x_3 \colon \tilde{s}_3 \triangleq \tilde{q}_3;\ X_3 \colon \tilde{c}_3 \triangleq \tilde{d}_3;\ \mathfrak{a}_3,\ x_4 \colon \tilde{s}_4 \triangleq \tilde{q}_4;\ X_4 \colon \tilde{c}_4 \triangleq \tilde{d}_4;\ \mathfrak{a}_4\} \subseteq P_n \cup S_n$*.*

Proof. Looking at the rules, it is easy to see that no rule application duplicates a variable of an AUP, and every fresh variable is only used in one AUP. □

Lemma 6 (Invariant 3). *Let* $P_0; S_0; \sigma_0 \Longrightarrow^* P_n; S_n; \sigma_n$ *be a derivation in* $\mathfrak{G}$*. If for all* $x_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0;\ X_0 \colon \tilde{c}_0 \triangleq \tilde{d}_0;\ \mathfrak{a}_0 \in P_0$ *the variables* x_0, X_0 *only appear together as term* $X_0(x_0)$ *in* σ_0 *then this implies that for all* $x_n \colon \tilde{s}_n \triangleq \tilde{q}_n;\ X_n \colon \tilde{c}_n \triangleq \tilde{d}_n;\ \mathfrak{a}_n \in P_n$ *the variables* x_n, X_n *only appear together as term* $X_n(x_n)$ *in* σ_n*. (This implies that they do not appear in* $Dom(\sigma_n)$*.)*

Proof. The rules Abs-L/Abs-R are trivial. From Lemma 5 we know that the variables of all the AUPs in $P_i \cup S_i$, $0 \leqslant i \leqslant n$ are pairwise disjoint. Furthermore, once a generalization variable appears in S_i, it will never appear in P_j again, for all $i \leqslant j \leqslant n$, because there is no rule which moves an AUP from the store S_j back to the problem set P_j. Therefore, for any generalization variable occurring in P_j, the rules Res-C, Mer-S, Clr-S (which only operate on generalization variables within S_i) have no effect on their appearance in σ_j. The rule Sol-H trivially maintains this property, as it moves an AUP to the store and therefore the condition is lifted for the corresponding variables. The rule App-A moves the selected AUP to the store, such that the condition is lifted for those generalization variables, e.g. x and X. Furthermore it introduces a new AUP with two fresh variables, say y and Y, and it composes the mapping $\sigma_i\{x \mapsto a_1(Y(y))\}$, which again maintains the property that y, Y only appear together as term $Y(y)$ in σ_{i+1}. The reasoning for the rule Spl-H is very similar. It introduces two new AUPs and also maintains this property. □

Lemma 7 (Invariant 4). *Let* $P_0;\ S_0;\ \sigma_0$ *such that for all* $x_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0;\ X_0 \colon \tilde{c}_0 \triangleq \tilde{d}_0;\ \mathfrak{a}_0 \in P_0$ *the variables* x_0, X_0 *only appear together as term* $X_0(x_0)$ *in* σ_0*. If* $P_0;\ S_0;\ \sigma_0 \Longrightarrow^* P_n;\ S_n;\ \sigma_n$ *is a derivation in* $\mathfrak{G}$ *then for all* $x_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0;\ X_0 \colon \tilde{c}_0 \triangleq \tilde{d}_0;\ \mathfrak{a}_0 \in P_0 \cup S_0$ *holds*

- $X_0(x_0)\sigma_0\sigma_{_L}(P_0 \cup S_0) = X_0(x_0)\sigma_n\sigma_{_L}(P_n \cup S_n)$,
- $X_0(x_0)\sigma_0\sigma_{_R}(P_0 \cup S_0) = X_0(x_0)\sigma_n\sigma_{_R}(P_n \cup S_n)$.

Proof. By induction on the length of derivations. The trivial base case is the derivation of length zero. Let P_0; S_0; σ_0 such that for all $x_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0$; $X_0 \colon \tilde{c}_0 \triangleq \tilde{d}_0$; $\mathfrak{a}_0 \in P_0$ the variables x_0, X_0 only appear together as term $X_0(x_0)$ in σ_0. Note that by Lemma 6 this property is an invariant of $\mathfrak{G}$. Let P_0; S_0; $\sigma_0 \Longrightarrow^* P_{n-1}$; S_{n-1}; $\sigma_{n-1} \Longrightarrow P_n$; S_n; σ_n be a derivation in $\mathfrak{G}$. As induction hypothesis (IH) we assume that for all $x_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0$; $X_0 \colon \tilde{c}_0 \triangleq \tilde{d}_0$; $\mathfrak{a}_0 \in P_0 \cup S_0$ holds

- $X_0(x_0)\sigma_0\sigma_{\mathrm{L}}(P_0 \cup S_0) = X_0(x_0)\sigma_{n-1}\sigma_{\mathrm{L}}(P_{n-1} \cup S_{n-1})$,
- $X_0(x_0)\sigma_0\sigma_{\mathrm{R}}(P_0 \cup S_0) = X_0(x_0)\sigma_{n-1}\sigma_{\mathrm{R}}(P_{n-1} \cup S_{n-1})$.

By case analysis on the applied rule, we will show that

- $X_0(x_0)\sigma_{n-1}\sigma_{\mathrm{L}}(P_{n-1} \cup S_{n-1}) = X_0(x_0)\sigma_n\sigma_{\mathrm{L}}(P_n \cup S_n)$,
- $X_0(x_0)\sigma_{n-1}\sigma_{\mathrm{R}}(P_{n-1} \cup S_{n-1}) = X_0(x_0)\sigma_n\sigma_{\mathrm{R}}(P_n \cup S_n)$.

We will only illustrate the proof for the left hand side $X_0(x_0)\sigma_{n-1}\sigma_{\mathrm{L}}(P_{n-1} \cup S_{n-1}) = X_0(x_0)\sigma_n\sigma_{\mathrm{L}}(P_n \cup S_n)$, serving as representative of both sides and skip the rule Abs-R which is a mirror image of Abs-L. For the sake of readability we will omit writing the alignments as we do not care about them in this proof.

Spl-H. $P_{n-1} = P \cup \{x \colon \tilde{s} \triangleq \tilde{q};\ X \colon \tilde{c} \triangleq \tilde{d}\}$,
$P_n = P \cup \{y \colon \tilde{s}|_{i_1}^{i_k} \triangleq \tilde{q}|_{j_1}^{j_k};\ Y \colon \circ \triangleq \circ\} \cup \{z \colon \tilde{s}|_{i_k^{++}}^{i_m} \triangleq \tilde{q}|_{j_k^{++}}^{j_m};\ Z \colon \circ \triangleq \circ\}$,
$S_n = S_{n-1} \cup \{x \colon \epsilon \triangleq \epsilon;\ X \colon \tilde{c}[\tilde{s}|_1^{i_1^{--}}, \circ, \tilde{s}|_{i_m^{++}}^{|\tilde{s}|}] \triangleq \tilde{d}[\tilde{q}|_1^{j_1^{--}}, \circ, \tilde{q}|_{j_m^{++}}^{|\tilde{q}|}]\}$,
$\sigma_n = \sigma_{n-1}\{x \mapsto (Y(y), Z(z))\}$,
$\tilde{s} = (\tilde{s}|_1^{i_1^{--}}, \tilde{s}|_{i_1}^{i_k}, \tilde{s}|_{i_k^{++}}^{i_m}, \tilde{s}|_{i_m^{++}}^{|\tilde{s}|})$, $\tilde{q} = (\tilde{q}|_1^{j_1^{--}}, \tilde{q}|_{j_1}^{j_k}, \tilde{q}|_{j_k^{++}}^{j_m}, \tilde{q}|_{j_m^{++}}^{|\tilde{q}|})$.

Using Definition 17 we have the equality

$$
\begin{aligned}
&\sigma_{n-1}\sigma_{\mathrm{L}}(P_{n-1} \cup S_{n-1}) \\
&= \sigma_{n-1}\sigma_{\mathrm{L}}(\{x \colon \tilde{s} \triangleq \tilde{q};\ X \colon \tilde{c} \triangleq \tilde{d}\} \cup P \cup S_{n-1}) \\
&= \sigma_{n-1}\{x \mapsto \tilde{s}, X \mapsto \tilde{c}\}\sigma_{\mathrm{L}}(P \cup S_{n-1}) \\
&= \sigma_{n-1}\{x \mapsto (\tilde{s}|_1^{i_1^{--}}, \tilde{s}|_{i_1}^{i_k}, \tilde{s}|_{i_k^{++}}^{i_m}, \tilde{s}|_{i_m^{++}}^{|\tilde{s}|}), X \mapsto \tilde{c}\}\sigma_{\mathrm{L}}(P \cup S_{n-1}).
\end{aligned}
$$

The variables y, Y, z, Z are fresh and therefore it holds for all $x_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0$; $X_0 \colon \tilde{c}_0 \triangleq \tilde{d}_0 \in P_0 \cup S_0$ that

$$
\begin{aligned}
&X_0(x_0)\sigma_{n-1}\{x \mapsto (\tilde{s}|_1^{i_1^{--}}, \tilde{s}|_{i_1}^{i_k}, \tilde{s}|_{i_k^{++}}^{i_m}, \tilde{s}|_{i_m^{++}}^{|\tilde{s}|}), X \mapsto \tilde{c}\}\sigma_{\mathrm{L}}(P \cup S_{n-1}) \\
&= X_0(x_0)\sigma_{n-1}\{x \mapsto (\tilde{s}|_1^{i_1^{--}}, Y(y), Z(z), \tilde{s}|_{i_m^{++}}^{|\tilde{s}|}), X \mapsto \tilde{c}\} \\
&\quad \{y \mapsto \tilde{s}|_{i_1}^{i_k}, Y \mapsto \circ\}\{z \mapsto \tilde{s}|_{i_k^{++}}^{i_m}, Z \mapsto \circ\}\sigma_{\mathrm{L}}(P \cup S_{n-1}) \\
&= X_0(x_0)\sigma_{n-1}\{x \mapsto (\tilde{s}|_1^{i_1^{--}}, Y(y), Z(z), \tilde{s}|_{i_m^{++}}^{|\tilde{s}|}), X \mapsto \tilde{c}\}
\end{aligned}
$$

$$\begin{aligned}
&\sigma_{\text{L}}(\{y\colon \tilde{s}|_{i_1}^{i_k} \triangleq \tilde{q}|_{j_1}^{j_k};\ Y\colon \circ \triangleq \circ\} \cup \{z\colon \tilde{s}|_{i_k^{++}}^{i_m} \triangleq \tilde{q}|_{j_k^{++}}^{j_m};\ Z\colon \circ \triangleq \circ\} \cup P \cup S_{n-1})\\
&= X_0(x_0)\sigma_{n-1}\{x \mapsto (\tilde{s}|_1^{i_1^{--}}, Y(y), Z(z), \tilde{s}|_{i_m^{++}}^{|\tilde{s}|}), X \mapsto \tilde{c}\}\sigma_{\text{L}}(P_n \cup S_{n-1}).
\end{aligned}$$

Further on, by Lemma 6 the variables x, X only appear together as term $X(x)$ in σ_{n-1} which leads to

$$\begin{aligned}
&X_0(x_0)\sigma_{n-1}\{x \mapsto (\tilde{s}|_1^{i_1^{--}}, Y(y), Z(z), \tilde{s}|_{i_m^{++}}^{|\tilde{s}|}), X \mapsto \tilde{c}\}\sigma_{\text{L}}(P_n \cup S_{n-1})\\
&= X_0(x_0)\sigma_{n-1}\{x \mapsto (Y(y), Z(z)), X \mapsto \tilde{c}[\tilde{s}|_1^{i_1^{--}}, \circ, \tilde{s}|_{i_m^{++}}^{|\tilde{s}|}]\}\sigma_{\text{L}}(P_n \cup S_{n-1})\\
&= X_0(x_0)\sigma_{n-1}\{x \mapsto (Y(y), Z(z))\}\{x \mapsto \epsilon, X \mapsto \tilde{c}[\tilde{s}|_1^{i_1^{--}}, \circ, \tilde{s}|_{i_m^{++}}^{|\tilde{s}|}]\}\sigma_{\text{L}}(P_n \cup S_{n-1})\\
&= X_0(x_0)\sigma_n\sigma_{\text{L}}(\{x\colon \epsilon \triangleq \epsilon;\ X\colon \tilde{c}[\tilde{s}|_1^{i_1^{--}}, \circ, \tilde{s}|_{i_m^{++}}^{|\tilde{s}|}] \triangleq \tilde{d}[\tilde{q}|_1^{j_1^{--}}, \circ, \tilde{q}|_{j_m^{++}}^{|\tilde{q}|}]\} \cup P_n \cup S_{n-1})\\
&= X_0(x_0)\sigma_n\sigma_{\text{L}}(P_n \cup S_n).
\end{aligned}$$

App-A. $P_{n-1} = \{x\colon (\tilde{s}_l, a_1(\tilde{s}), \tilde{s}_r) \triangleq (\tilde{q}_l, a_1(\tilde{q}), \tilde{q}_r);\ X\colon \tilde{c} \triangleq \tilde{d}\} \cup P$,
$P_n = \{y\colon \tilde{s} \triangleq \tilde{q};\ Y\colon \circ \triangleq \circ\} \cup P$,
$S_n = S_{n-1} \cup \{x\colon \epsilon \triangleq \epsilon;\ X\colon \tilde{c}[\tilde{s}_l, \circ, \tilde{s}_r] \triangleq \tilde{d}[\tilde{q}_l, \circ, \tilde{q}_r]\}$,
$\sigma_n = \sigma_{n-1}\{x \mapsto a_1(Y(y))\}$.

By Definition 17 we get

$$\begin{aligned}
&\sigma_{n-1}\sigma_{\text{L}}(P_{n-1} \cup S_{n-1})\\
&= \sigma_{n-1}\sigma_{\text{L}}(\{x\colon (\tilde{s}_l, a_1(\tilde{s}), \tilde{s}_r) \triangleq (\tilde{q}_l, a_1(\tilde{q}), \tilde{q}_r);\ X\colon \tilde{c} \triangleq \tilde{d}\} \cup P \cup S_{n-1})\\
&= \sigma_{n-1}\{x \mapsto (\tilde{s}_l, a_1(\tilde{s}), \tilde{s}_r), X \mapsto \tilde{c}\}\sigma_{\text{L}}(P \cup S_{n-1}).
\end{aligned}$$

The variables y, Y are fresh and therefore it holds for all $x_0\colon \tilde{s}_0 \triangleq \tilde{q}_0;\ X_0\colon \tilde{c}_0 \triangleq \tilde{d}_0 \in P_0 \cup S_0$ that

$$\begin{aligned}
&X_0(x_0)\sigma_{n-1}\{x \mapsto (\tilde{s}_l, a_1(\tilde{s}), \tilde{s}_r), X \mapsto \tilde{c}\}\sigma_{\text{L}}(P \cup S_{n-1})\\
&= X_0(x_0)\sigma_{n-1}\{x \mapsto (\tilde{s}_l, a_1(Y(y)), \tilde{s}_r), X \mapsto \tilde{c}\}\{y \mapsto \tilde{s}, Y \mapsto \circ\}\sigma_{\text{L}}(P \cup S_{n-1})\\
&= X_0(x_0)\sigma_{n-1}\{x \mapsto (\tilde{s}_l, a_1(Y(y)), \tilde{s}_r), X \mapsto \tilde{c}\}\sigma_{\text{L}}(\{y\colon \tilde{s} \triangleq \tilde{q};\ Y\colon \circ \triangleq \circ\} \cup P \cup S_{n-1})\\
&= X_0(x_0)\sigma_{n-1}\{x \mapsto (\tilde{s}_l, a_1(Y(y)), \tilde{s}_r), X \mapsto \tilde{c}\}\sigma_{\text{L}}(P_n \cup S_{n-1}).
\end{aligned}$$

Further on, by Lemma 6 the variables x, X only appear together as term $X(x)$ in σ_{n-1} which leads to

$$\begin{aligned}
&X_0(x_0)\sigma_{n-1}\{x \mapsto (\tilde{s}_l, a_1(Y(y)), \tilde{s}_r), X \mapsto \tilde{c}\}\sigma_{\text{L}}(P_n \cup S_{n-1})\\
&= X_0(x_0)\sigma_{n-1}\{x \mapsto a_1(Y(y)), X \mapsto \tilde{c}[\tilde{s}_l, \circ, \tilde{s}_r]\}\sigma_{\text{L}}(P_n \cup S_{n-1})\\
&= X_0(x_0)\sigma_{n-1}\{x \mapsto a_1(Y(y))\}\{x \mapsto \epsilon, X \mapsto \tilde{c}[\tilde{s}_l, \circ, \tilde{s}_r]\}\sigma_{\text{L}}(P_n \cup S_{n-1})\\
&= X_0(x_0)\sigma_n\sigma_{\text{L}}(\{x\colon \epsilon \triangleq \epsilon;\ X\colon \tilde{c}[\tilde{s}_l, \circ, \tilde{s}_r] \triangleq \tilde{d}[\tilde{q}_l, \circ, \tilde{q}_r]\} \cup P_n \cup S_{n-1})\\
&= X_0(x_0)\sigma_n\sigma_{\text{L}}(P_n \cup S_n).
\end{aligned}$$

Abs-L. $P_{n-1} = \{x\colon (\tilde{s}_l, \phi(\tilde{s}), \tilde{s}_r) \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d}\} \cup P$,
$P_n = \{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c}[\tilde{s}_l, \phi(\circ), \tilde{s}_r] \triangleq \tilde{d}\} \cup P$,
$S_n = S_{n-1}$, $\sigma_n = \sigma_{n-1}$.

Starting with Definition 17 we get

$$\begin{aligned}&\sigma_{n-1}\sigma_{\text{L}}(P_{n-1}\cup S_{n-1})\\ =\,&\sigma_n\sigma_{\text{L}}(\{x\colon(\tilde{s}_l,\phi(\tilde{s}),\tilde{s}_r)\triangleq\tilde{q};\ X\colon\tilde{c}\triangleq\tilde{d}\}\cup P\cup S_n)\\ =\,&\sigma_n\{x\mapsto(\tilde{s}_l,\phi(\tilde{s}),\tilde{s}_r),X\mapsto\tilde{c}\}\sigma_{\text{L}}(P\cup S_n).\end{aligned}$$

By Lemma 6 the variables x, X only appear together as term $X(x)$ in $\sigma_{n-1}=\sigma_n$. Therefore it follows that for all $x_0\colon\tilde{s}_0\triangleq\tilde{q}_0;\ X_0\colon\tilde{c}_0\triangleq\tilde{d}_0\in P_0\cup S_0$ holds

$$\begin{aligned}&X_0(x_0)\sigma_n\{x\mapsto(\tilde{s}_l,\phi(\tilde{s}),\tilde{s}_r),X\mapsto\tilde{c}\}\sigma_{\text{L}}(P\cup S_n)\\ =\,&X_0(x_0)\sigma_n\{x\mapsto\tilde{s},X\mapsto\tilde{c}[\tilde{s}_l,\phi(\circ),\tilde{s}_r]\}\sigma_{\text{L}}(P\cup S_n)\\ =\,&X_0(x_0)\sigma_n\sigma_{\text{L}}(\{x\colon\tilde{s}\triangleq\tilde{q};\ X\colon\tilde{c}[\tilde{s}_l,\phi(\circ),\tilde{s}_r]\triangleq\tilde{d}\}\cup P\cup S_n)\\ =\,&X_0(x_0)\sigma_n\sigma_{\text{L}}(P_n\cup S_n).\end{aligned}$$

Sol-H. $P_n=P_{n-1}\backslash\{x\colon\tilde{s}\triangleq\tilde{q};\ X\colon\circ\triangleq\circ\}$,
$S_n=S_{n-1}\cup\{x\colon\tilde{s}\triangleq\tilde{q};\ X\colon\circ\triangleq\circ\}$,
$\sigma_n=\sigma_{n-1}\{X\mapsto\circ\}$.

Obviously the sets $P_{n-1}\cup S_{n-1}$ and $P_n\cup S_n$ are equal such that $\sigma_{\text{L}}(P_{n-1}\cup S_{n-1})=\sigma_{\text{L}}(P_n\cup S_n)$ and furthermore $\{x\colon\tilde{s}\triangleq\tilde{q};\ X\colon\circ\triangleq\circ;\ \mathfrak{e}\}\in P_n\cup S_n$ leads to $X\sigma_{\text{L}}(P_n\cup S_n)=\circ$, by Definition 17. Finally we get

$$\begin{aligned}&\sigma_{n-1}\sigma_{\text{L}}(P_n\cup S_n)\\ =\,&\sigma_{n-1}\{X\mapsto\circ\}\sigma_{\text{L}}(P_n\cup S_n)\\ =\,&\sigma_n\sigma_{\text{L}}(P_n\cup S_n).\end{aligned}$$

Res-C. $P_n=P_{n-1}$,
$S_n\backslash\{x\colon\epsilon\triangleq\epsilon;\ X\colon\dot{c}\triangleq\dot{d},\ y\colon\tilde{s}_l\triangleq\tilde{q}_l;\ Y\colon\circ\triangleq\circ,\ z\colon\tilde{s}_r\triangleq\tilde{q}_r;\ Z\colon\circ\triangleq\circ\}$
$=S_{n-1}\backslash\{x\colon\epsilon\triangleq\epsilon;\ X\colon(\tilde{s}_l,\dot{c},\tilde{s}_r)\triangleq(\tilde{q}_l,\dot{d},\tilde{q}_r)\}$,
$\sigma_n=\sigma_{n-1}\{X\mapsto(y,X(\circ),z)\}$.

Therefore by Definition 17 holds

$$\begin{aligned}&\sigma_{n-1}\sigma_{\text{L}}(P_{n-1}\cup S_{n-1})\\ =\,&\sigma_{n-1}\{X\mapsto(\tilde{s}_l,\dot{c},\tilde{s}_r),x\mapsto\epsilon\}\sigma_{\text{L}}(P_{n-1}\cup\\ &S_{n-1}\backslash\{x\colon\epsilon\triangleq\epsilon;\ X\colon(\tilde{s}_l,\dot{c},\tilde{s}_r)\triangleq(\tilde{q}_l,\dot{d},\tilde{q}_r)\})\\ =\,&\sigma_{n-1}\{X\mapsto(\tilde{s}_l,\dot{c},\tilde{s}_r),x\mapsto\epsilon\}\sigma_{\text{L}}(P_{n-1}\cup\\ &S_n\backslash\{x\colon\epsilon\triangleq\epsilon;\ X\colon\dot{c}\triangleq\dot{d},\ y\colon\tilde{s}_l\triangleq\tilde{q}_l;\ Y\colon\circ\triangleq\circ,\ z\colon\tilde{s}_r\triangleq\tilde{q}_r;\ Z\colon\circ\triangleq\circ\})\\ =\,&\sigma_{n-1}\{X\mapsto(\tilde{s}_l,X(\circ),\tilde{s}_r)\}\{x\mapsto\epsilon,X\mapsto\dot{c}\}\sigma_{\text{L}}(P_{n-1}\cup\\ &S_n\backslash\{x\colon\epsilon\triangleq\epsilon;\ X\colon\dot{c}\triangleq\dot{d},\ y\colon\tilde{s}_l\triangleq\tilde{q}_l;\ Y\colon\circ\triangleq\circ,\ z\colon\tilde{s}_r\triangleq\tilde{q}_r;\ Z\colon\circ\triangleq\circ\})\\ =\,&\sigma_{n-1}\{X\mapsto(\tilde{s}_l,X(\circ),\tilde{s}_r)\}\sigma_{\text{L}}(P_{n-1}\cup\\ &S_n\backslash\{y\colon\tilde{s}_l\triangleq\tilde{q}_l;\ Y\colon\circ\triangleq\circ,\ z\colon\tilde{s}_r\triangleq\tilde{q}_r;\ Z\colon\circ\triangleq\circ\}).\end{aligned}$$

The variables y, z, Y, Z are fresh and therefore it holds for all $x_0\colon\tilde{s}_0\triangleq\tilde{q}_0;\ X_0\colon\tilde{c}_0\triangleq\tilde{d}_0\in P_0\cup S_0$ that

$$X_0(x_0)\sigma_{n-1}\{X\mapsto(\tilde{s}_l,X(\circ),\tilde{s}_r)\}\sigma_{\text{L}}(P_{n-1}\cup$$

$$
\begin{aligned}
&S_n\backslash\{y\colon \tilde{s}_l \triangleq \tilde{q}_l;\, Y\colon \circ \triangleq \circ,\ z\colon \tilde{s}_r \triangleq \tilde{q}_r;\, Z\colon \circ \triangleq \circ\}) \\
=\ & X_0(x_0)\sigma_{n-1}\{X \mapsto (y, X(\circ), z)\}\{y \mapsto \tilde{s}_l, z \mapsto \tilde{s}_r, Y \mapsto \circ, Z \mapsto \circ\}\sigma_{\mathrm{L}}(P_{n-1} \cup \\
&S_n\backslash\{y\colon \tilde{s}_l \triangleq \tilde{q}_l;\, Y\colon \circ \triangleq \circ,\ z\colon \tilde{s}_r \triangleq \tilde{q}_r;\, Z\colon \circ \triangleq \circ\}) \\
=\ & X_0(x_0)\sigma_{n-1}\{X \mapsto (y, X(\circ), z)\}\sigma_{\mathrm{L}}(P_{n-1} \cup S_n) \\
=\ & X_0(x_0)\sigma_n\sigma_{\mathrm{L}}(P_{n-1} \cup S_n) \\
=\ & X_0(x_0)\sigma_n\sigma_{\mathrm{L}}(P_n \cup S_n).
\end{aligned}
$$

Mer-S. $P_n = P_{n-1}$,
$S_n = S_{n-1}\backslash\{x_2\colon \tilde{s} \triangleq \tilde{q};\, X_2\colon \tilde{c} \triangleq \tilde{d}\}$,
$x_1\colon \tilde{s} \triangleq \tilde{q};\, X_1\colon \tilde{c} \triangleq \tilde{d} \in S_n$,
$\sigma_n = \sigma_{n-1}\{x_2 \mapsto x_1, X_2 \mapsto X_1\}$.
Therefore by Definition 17 holds

$$
\begin{aligned}
&\sigma_{\mathrm{L}}(P_{n-1} \cup S_{n-1}) \\
=\ & \{x_2 \mapsto \tilde{s}, X_2 \mapsto \tilde{c}\}\sigma_{\mathrm{L}}(P_{n-1} \cup S_{n-1}\backslash\{x_2\colon \tilde{s} \triangleq \tilde{q};\, X_2\colon \tilde{c} \triangleq \tilde{d}\}) \\
=\ & \{x_2 \mapsto \tilde{s}, X_2 \mapsto \tilde{c}\}\sigma_{\mathrm{L}}(P_{n-1} \cup S_n).
\end{aligned}
$$

From the fact $x_1\colon \tilde{s} \triangleq \tilde{q};\, X_1\colon \tilde{c} \triangleq \tilde{d} \in S_n$ follows that $x_1\sigma_{\mathrm{L}}(P_{n-1} \cup S_n) = \tilde{s}$ and $X_1\sigma_{\mathrm{L}}(P_{n-1} \cup S_n) = \tilde{c}$, which finally leads to

$$
\begin{aligned}
&\sigma_{n-1}\{x_2 \mapsto \tilde{s}, X_2 \mapsto \tilde{c}\}\sigma_{\mathrm{L}}(P_{n-1} \cup S_n) \\
=\ & \sigma_{n-1}\{x_2 \mapsto x_1, X_2 \mapsto X_1\}\sigma_{\mathrm{L}}(P_{n-1} \cup S_n) \\
=\ & \sigma_n\sigma_{\mathrm{L}}(P_{n-1} \cup S_n) \\
=\ & \sigma_n\sigma_{\mathrm{L}}(P_n \cup S_n).
\end{aligned}
$$

Clr-S. $P_n = P_{n-1}$,
$S_n = S_{n-1}\backslash\{x\colon \epsilon \triangleq \epsilon;\, X\colon \circ \triangleq \circ\}$,
$\sigma_n = \sigma_{n-1}\{x \mapsto \epsilon, X \mapsto \circ\}$.
By definition 17 it holds that

$$
\begin{aligned}
&\sigma_{n-1}\sigma_{\mathrm{L}}(P_{n-1} \cup S_{n-1}) \\
=\ & \sigma_{n-1}\{x \mapsto \epsilon, X \mapsto \circ\}\sigma_{\mathrm{L}}(P_{n-1} \cup S_{n-1}\backslash\{x\colon \epsilon \triangleq \epsilon;\, X\colon \circ \triangleq \circ\}) \\
=\ & \sigma_{n-1}\{x \mapsto \epsilon, X \mapsto \circ\}\sigma_{\mathrm{L}}(P_{n-1} \cup S_n) \\
=\ & \sigma_n\sigma_{\mathrm{L}}(P_{n-1} \cup S_n) \\
=\ & \sigma_n\sigma_{\mathrm{L}}(P_n \cup S_n).
\end{aligned}
$$

□

This lemma has a corollary which states that for the invariant, the initial substitution is irrelevant:

Corollary 8. *If* $P_0;\, S_0;\, \vartheta_0 \Longrightarrow^* P_n;\, S_n;\, \vartheta_0\vartheta_1 \ldots \vartheta_n$ *is a derivation in* $\mathfrak{G}$ *then for all* $x_0\colon \tilde{s}_0 \triangleq \tilde{q}_0;\, X_0\colon \tilde{c}_0 \triangleq \tilde{d}_0;\, \mathfrak{a}_0 \in P_0 \cup S_0$ *holds*

- $X_0(x_0)\sigma_{\mathrm{L}}(P_0 \cup S_0) = X_0(x_0)\vartheta_1 \ldots \vartheta_n\sigma_{\mathrm{L}}(P_n \cup S_n)$,

- $X_0(x_0)\sigma_{_R}(P_0 \cup S_0) = X_0(x_0)\vartheta_1 \dots \vartheta_n\sigma_{_R}(P_n \cup S_n)$.

Theorem 9 (Termination). *The system* $\mathfrak{G}$ *terminates on any input.*

Proof. We define the complexity measure of the triple P; S; σ as a tuple of multisets (M_P, M_S), where

$$M_P = \{\|\tilde{s}\| + \|\tilde{q}\| \mid x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d};\ \mathfrak{a} \in P\},$$
$$M_S = \{\|\tilde{c}\| + \|\tilde{d}\| \mid x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d};\ \mathfrak{a} \in S\}.$$

The measures are compared by the well-founded lexicographic ordering. Each rule strictly reduces the complexity of the triple P; S; σ. □

The soundness theorem shows that $\mathfrak{G}$ indeed computes rigid generalizations. Besides, the store keeps the information which indicates how to obtain the initial hedges from the generalization:

Theorem 10 (Soundness). *Let* P *be a set of AUPs of the form* $\{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \circ \triangleq \circ;\ \mathfrak{a}\}$. *Every exhaustive rule application in* $\mathfrak{G}$ *yields a derivation* P; $\varnothing$; $\varepsilon \Longrightarrow^{+} \varnothing$; S; σ *where* $\tilde{g} = X(x)\sigma$ *is a rigid generalization of* $\tilde{s}$ *and* $\tilde{q}$ *with respect to* $\mathfrak{a}$ *and the store* S *records all the differences such that* $\tilde{g}\sigma_{_L}(S) = \tilde{s}$ *and* $\tilde{g}\sigma_{_R}(S) = \tilde{q}$.

Proof. We will proceed in the following way:

1. For any arbitrary fixed AUP in P, there is a rule in $\mathfrak{G}$ which is applicable.
2. $\tilde{g}$ is a supporting generalization of $\tilde{s}$ and $\tilde{q}$ with respect to $\mathfrak{a}$.
   $\mathfrak{G}$ maintains the invariant that $P \cup S$ is a set of AUPs.
   S records all the differences such that $\tilde{g}\sigma_{\text{L}}(S) = \tilde{s}$ and $\tilde{g}\sigma_{\text{R}}(S) = \tilde{q}$.
3. $\tilde{g}$ is a rigid generalization of $\tilde{s}$ and $\tilde{q}$ with respect to $\mathfrak{a}$.

First we introduce some auxiliary notations used in the proof. The notation ϑ_i^k is used for a substitution composition $\vartheta_i\vartheta_{i+1}\dots\vartheta_k$. Given an AUP $x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d};\ \mathfrak{a} \in P$ we denote by $\mathring{\mathfrak{a}}$ the admissible alignment of $\tilde{c}[\tilde{s}]$ and $\tilde{d}[\tilde{q}]$. It is obtained by extending $\mathfrak{a} = a_1\langle i_1{\cdot}I_1, j_1{\cdot}J_1\rangle \dots a_m\langle i_m{\cdot}I_m, j_m{\cdot}J_m\rangle$ with the positions of the holes in $\tilde{c}$, $\tilde{d}$ donated $I_\circ{\cdot}i_\circ$, $J_\circ{\cdot}j_\circ$, respectively, in the way $\mathring{\mathfrak{a}} = a_1\langle I_\circ{\cdot}(i_\circ + i_1){\cdot}I_1, J_\circ{\cdot}(j_\circ + j_1){\cdot}J_1\rangle \dots a_m\langle I_\circ{\cdot}(i_\circ + i_m){\cdot}I_m, J_\circ{\cdot}(j_\circ + j_m){\cdot}J_m\rangle$.

Ad 1. Show that for any $x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d};\ \mathfrak{a} \in P$ with $\mathfrak{a}$ being an admissible alignment of $\tilde{s}$ and $\tilde{q}$ there is a rule which can be applied. If $\mathfrak{a} = \mathfrak{e}$ then the rule Sol-H is applicable by Lemma 4. Let $\mathfrak{a} = a_1\langle i_1{\cdot}I_1, j_1{\cdot}J_1\rangle \dots a_m\langle i_m{\cdot}I_m, j_m{\cdot}J_m\rangle \neq \mathfrak{e}$. From the condition of Spl-H it follows that the rule Spl-H is applicable iff $i_1 \neq i_m$ and $j_1 \neq j_m$. Otherwise either $i_1 = \dots = i_m$ or $j_1 = \dots = j_m$. W.l.o.g. we assume $i_1 = \dots = i_m$. If $I_1 \neq \lambda$ then Abs-L is applicable, therefore we furthermore assume $I_1 = \lambda$, which gives us an alignment of the form $a_1\langle i, j_1{\cdot}J_1\rangle \dots a_m\langle i{\cdot}I_m, j_m{\cdot}J_m\rangle$. If we also have $j_1 = \dots = j_m$, then either we can apply Abs-R or App-A. (Note that, if $a_1\langle i, j\rangle a_2\langle I_2, J_2\rangle \dots a_m\langle I_m, J_m\rangle$ is an admissible alignment of $\tilde{s}$ and $\tilde{q}$, then a_1 is the symbol at position i in $\tilde{s}$ and at position j in $\tilde{q}$.) This leaves us with the case where $j_1 \neq j_m$, leading to $j_1{\cdot}J_1 \between j_m{\cdot}J_m$ but $i \sqsubset i{\cdot}I_m$ which is a collision and cannot appear in an admissible alignment.

Ad 2. We use well-founded induction on the length of derivations. Our base case is the derivation of length zero. We know that the initial problem set P is a set of AUPs and for any AUP in P a rule is applicable. It follows that for the base case holds $P = \varnothing$. Furthermore all the AUPs in S have an empty alignment. Every generalization of two hedges is supporting with respect to an empty alignment. Therefore Corollary 8 covers the base case. Let P_0; S_0; $\vartheta_0 \Longrightarrow P_1$; S_1; $\vartheta_0\vartheta_1 \Longrightarrow^* \varnothing$; S_n; ϑ_0^n be a derivation in $\mathfrak{G}$. As induction hypothesis (IH) we assume that

- $X_1(x_1)\vartheta_2^n$ is a supporting generalization of $\tilde{c}_1[\tilde{s}_1]$ and $\tilde{d}_1[\tilde{q}_1]$ with respect to $\mathring{\mathfrak{a}}_1$ for all $x_1\colon \tilde{s}_1 \triangleq \tilde{q}_1$; $X_1\colon \tilde{c}_1 \triangleq \tilde{d}_1$; $\mathfrak{a}_1 \in P_1 \cup S_1$,

By case analysis on the applied rule, we will show that this implies

- $X_0(x_0)\vartheta_1^n$ is a supporting generalization of $\tilde{c}_0[\tilde{s}_0]$ and $\tilde{d}_0[\tilde{q}_0]$ with respect to $\mathring{\mathfrak{a}}_0$ for any arbitrary but fixed $x_0\colon \tilde{s}_0 \triangleq \tilde{q}_0$; $X_0\colon \tilde{c}_0 \triangleq \tilde{d}_0$; $\mathfrak{a}_0 \in P_0 \cup S_0$,

Spl-H. $\mathfrak{a}_0 = a_1\langle i_1{\cdot}I_1, j_1{\cdot}J_1\rangle \ldots a_k\langle i_k{\cdot}I_k, j_k{\cdot}J_k\rangle$
$a_{k+1}\langle i_{k+1}{\cdot}I_{k+1}, j_{k+1}{\cdot}J_{k+1}\rangle \ldots a_m\langle i_m{\cdot}I_m, j_m{\cdot}J_m\rangle$,
$\vartheta_1 = \{x_0 \mapsto (Y(y), Z(z))\}$.

By the IH we know that $Y(y)\vartheta_2^n$ is a supporting generalization of $\tilde{s}_0|_{i_1}^{i_k}$ and $\tilde{q}_0|_{j_1}^{j_k}$ with respect to $a_1\langle(i_1 - i_1^{--})\cdot I_1, (j_1 - j_1^{--})\cdot J_1\rangle \ldots a_k\langle(i_k - i_1^{--})\cdot I_k, (j_k - j_1^{--})\cdot J_k\rangle$, and $Z(z)\vartheta_2^n$ is a supporting generalization of $\tilde{s}_0|_{i_k^{++}}^{i_m}$ and $\tilde{q}_0|_{j_k^{++}}^{j_m}$ with respect to $a_{k+1}\langle(i_{k+1} - i_k)\cdot I_{k+1}, (j_{k+1} - j_k)\cdot J_{k+1}\rangle \ldots a_m\langle(i_m - i_k)\cdot I_m, (j_m - j_k)\cdot J_m\rangle$. It follows that $(Y(y), Z(z))\vartheta_2^n = x_0\vartheta_1^n$ is a supporting generalization of $\tilde{s}_0|_{i_1}^{i_m}$ and $\tilde{q}_0|_{j_1}^{j_m}$ with respect to $a_1\langle(i_1 - i_1^{--})\cdot I_1, (j_1 - j_1^{--})\cdot J_1\rangle \ldots a_m\langle(i_m - i_1^{--})\cdot I_m, (j_m - j_1^{--})\cdot J_m\rangle$. Notice that our contexts are hedges. Hence $X_0(x_0)\vartheta_1^n$ is a supporting generalization of $(\tilde{r}_1, \tilde{s}_0|_{i_1}^{i_m}, \tilde{r}_2)$ and $(\tilde{r}_3, \tilde{q}_0|_{j_1}^{j_m}, \tilde{r}_4)$ with respect to $a_1\langle(i_1 - i_1^{--} + |\tilde{r}_1|)\cdot I_1, (j_1 - j_1^{--} + |\tilde{r}_3|)\cdot J_1\rangle \ldots a_m\langle(i_m - i_1^{--} + |\tilde{r}_1|)\cdot I_m, (j_m - j_1^{--} + |\tilde{r}_3|)\cdot J_m\rangle$, where $\tilde{r}_1, \tilde{r}_2, \tilde{r}_3, \tilde{r}_4$ are arbitrary hedges. Finally we set $\tilde{r}_1 = \tilde{s}_0|_1^{i_1^{--}}, \tilde{r}_2 = \tilde{s}_0|_{i_m^{++}}^{|\tilde{s}_0|}, \tilde{r}_3 = \tilde{q}_0|_1^{j_1^{--}}, \tilde{r}_4 = \tilde{q}_0|_{j_m^{++}}^{|\tilde{q}_0|}$ to get $X_0(x_0)\vartheta_1^n$ is a supporting generalization of $\tilde{s}_0$ and $\tilde{q}_0$ with respect to $a_1\langle i_1{\cdot}I_1,\, j_1{\cdot}J_1\rangle \ldots a_m\langle i_m{\cdot}I_m,\, j_m{\cdot}J_m\rangle = \mathfrak{a}_0$ and it also is a supporting generalization of $\tilde{c}_0[\tilde{s}_0]$ and $\tilde{d}_0[\tilde{q}_0]$ with respect to $\mathring{\mathfrak{a}}_0$.

App-A. $\tilde{s}_0 = (\tilde{s}_l, a_1(\tilde{s}_1), \tilde{s}_r)$, $\quad \tilde{q}_0 = (\tilde{q}_l, a_1(\tilde{q}_1), \tilde{q}_r)$,
$\mathfrak{a}_0 = a_1\langle i, j\rangle\, a_2\langle i{\cdot}I_2, j{\cdot}J_2\rangle \ \ldots\ a_m\langle i{\cdot}I_m, j{\cdot}J_m\rangle$,
$\vartheta_1 = \{x_0 \mapsto a_1(Y(y))\}$.

By IH $Y(y)\vartheta_2^n$ is a supporting generalization of $\tilde{s}_1$ and $\tilde{q}_1$ with respect to $a_2\langle I_2, J_2\rangle \ldots a_m\langle I_m, J_m\rangle$. It follows that $a_1(Y(y)\vartheta_2^n) = x_0\vartheta_1^n$ is a supporting generalization of the terms $a_1(\tilde{s}_1)$ and $a_1(\tilde{q}_1)$ with respect to $a_1\langle 1, 1\rangle a_2\langle 1{\cdot}I_2, 1{\cdot}J_2\rangle \ldots a_m\langle 1{\cdot}I_m, 1{\cdot}J_m\rangle$. Similarly as for Spl-H we conclude that $X_0(x_0)\vartheta_1^n$ is a supporting generalization of $(\tilde{s}_l, a_1(\tilde{s}_1), \tilde{s}_r)$ and $(\tilde{q}_l, a_1(\tilde{q}_1), \tilde{q}_r)$ with respect to $\mathfrak{a}_0$ where $i = |\tilde{s}_l| + 1$ and $j = |\tilde{q}_l| + 1$. It also is a supporting generalization of $\tilde{c}_0[\tilde{s}_0]$ and $\tilde{d}_0[\tilde{q}_0]$ with respect to $\mathring{\mathfrak{a}}_0$.

Abs-L. $x_0 = x_1$, $X_0 = X_1$,
$\tilde{s}_0 = (\tilde{s}_l, \phi(\tilde{s}_1), \tilde{s}_r)$, $\tilde{q}_0 = \tilde{q}_1$,
$\tilde{c}_1 = \tilde{c}_0[\tilde{s}_l, \phi(\circ), \tilde{s}_r]$, $\tilde{d}_0 = \tilde{d}_1$,
$\mathfrak{a}_0 = a_1\langle i{\cdot}I_1, J_1\rangle \ldots a_m\langle i{\cdot}I_m, J_m\rangle$,
$\vartheta_1 = \varepsilon$.

It directly follows $\tilde{c}_1[\tilde{s}_1] = \tilde{c}_0[\tilde{s}_l, \phi(\circ), \tilde{s}_r][\tilde{s}_1] = \tilde{c}_0[\tilde{s}_l, \phi(\tilde{s}_1), \tilde{s}_r] = \tilde{c}_0[\tilde{s}_0]$. By IH $X_1(x_1)\vartheta_2^n = X_0(x_0)\vartheta_1^n$ is a supporting generalization of $\tilde{c}_1[\tilde{s}_1]$ and $\tilde{d}_1[\tilde{q}_0]$ with respect to $a_1\langle I_1, J_1\rangle \ldots a_m\langle I_m, J_m\rangle$. From Corollary 8 we know that $X_0(x_0)\vartheta_1^n$ is a generalization of $\tilde{c}_0[\tilde{s}_0]$ and $\tilde{d}_0[\tilde{q}_0]$. It follows that $X_0(x_0)\vartheta_1^n$ is a supporting generalization of $\tilde{c}_0[\tilde{s}_l, \phi(\tilde{s}_1), \tilde{s}_r]$ and $\tilde{d}_0[\tilde{q}_0]$ with respect to $\mathring{\mathfrak{a}}_0$, were $i = |\tilde{s}_l| + 1$.

Abs-R. The reasoning is the same as for Abs-L.

Sol-H, Res-C, Mer-S, Clr-S. Those remaining rules operate only on AUPs with empty alignments. Every generalization of two hedges is supporting with respect to an empty alignment. Therefore Corollary 8 covers those cases.

Summary. It follows that $\mathfrak{G}$ maintains the invariant that $P \cup S$ is a set of AUPs. Let P be a set of AUPs of the form $\{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \circ \triangleq \circ;\ \mathfrak{a}\}$ and $P;\ \varnothing;\ \varepsilon \Longrightarrow^+ \varnothing;\ S;\ \sigma$ be a derivation in $\mathfrak{G}$. By Definition 17 we have $X(x)\sigma_{\text{L}}(\{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \circ \triangleq \circ;\ \mathfrak{a}\}) = \tilde{s}$ and $X(x)\sigma_{\text{R}}(\{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \circ \triangleq \circ;\ \mathfrak{a}\}) = \tilde{q}$. From Lemma 7 follows that $X(x)\sigma\sigma_{\text{L}}(S) = \tilde{s}$ and $X(x)\sigma\sigma_{\text{R}}(S) = \tilde{q}$.

Ad 3. Let $P;\ \varnothing;\ \varepsilon \Longrightarrow^+ \varnothing;\ S;\ \sigma$ be a derivation in $\mathfrak{G}$ and $\{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \circ \triangleq \circ;\ \mathfrak{a}\} \in P$ arbitrary. From above we know that $\tilde{g} = X(x)\sigma$ is a supporting generalization of $\tilde{s}$ and $\tilde{q}$ with respect to $\mathfrak{a}$. First we discuss the following property:

- There are substitutions σ, ϑ with $\tilde{g}\sigma = \tilde{s}$ and $\tilde{g}\vartheta = \tilde{q}$ where all the contexts in σ, ϑ are terms (not arbitrary hedges).

The rule Res-C eliminates all those contexts which are hedges from the store S and we already showed that $\tilde{g}\sigma_{\text{L}}(S) = \tilde{s}$ and $\tilde{g}\sigma_{\text{R}}(S) = \tilde{q}$. Therefore we just set $\sigma = \sigma_{\text{L}}(S)$ and $\vartheta = \sigma_{\text{R}}(S)$.

Let $\tilde{r}$ be an arbitrary hedge. We define the predicate $\mathfrak{P}(\tilde{r})$ to be true iff the following properties hold:

- No context variable in $\tilde{r}$ applies to the empty hedge.
- $\tilde{r}$ doesn't contain horizontal consecutive hedge variables.
- $\tilde{r}$ doesn't contain vertical chains of variables.
- $\tilde{r}$ doesn't contain context variables with a hedge variable as the first or the last argument.

Let $P_0;\ S_0;\ \vartheta_0 \Longrightarrow P_1;\ S_1;\ \vartheta_0\vartheta_1 \Longrightarrow^* \varnothing;\ S_n;\ \vartheta_0^n$ be an exhaustive derivation in $\mathfrak{G}$. As IH we assume that for all $x_1\colon \tilde{s}_1 \triangleq \tilde{q}_1;\ X_1\colon \tilde{c}_1 \triangleq \tilde{d}_1;\ \mathfrak{a}_1 \in P_1$ holds $\mathfrak{P}(X_1(x_1)\vartheta_2^n)$, and furthermore for all $y_1\colon \tilde{s}_1 \triangleq \tilde{q}_1;\ Y_1\colon \tilde{c}_1 \triangleq \tilde{d}_1 \in S_1$ holds $\mathfrak{P}(y_1\vartheta_2^n)$ and $\mathfrak{P}(Y_1(\circ)\vartheta_2^n)$. (Note that from $\mathfrak{P}(X_1(x_1)\vartheta_2^n)$ follows $\mathfrak{P}(x_1\vartheta_2^n)$ and $\mathfrak{P}(X_1(\circ)\vartheta_2^n)$.) The trivial base case is the derivation of length zero where $P = \varnothing$ and $\vartheta_1 = \varepsilon$.

By case analysis on the applied rule, we will show that this implies for any arbitrary but fixed $x_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0$; $X_0 \colon \tilde{c}_0 \triangleq \tilde{d}_0$; $\mathfrak{a}_0 \in P_0$ holds $\mathfrak{P}(X_0(x_0)\vartheta_1^n)$, and for any $y_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0$; $Y_0 \colon \tilde{c}_0 \triangleq \tilde{d}_0 \in S_0$ holds $\mathfrak{P}(y_0\vartheta_1^n)$ and $\mathfrak{P}(Y_0(\circ)\vartheta_1^n)$.

Abs-L, Abs-R. $P_0 = P \cup \{x \colon \tilde{s}_0 \triangleq \tilde{q}_0;\ X \colon \tilde{c}_0 \triangleq \tilde{d}_0;\ \mathfrak{a}_0\}$,
$P_1 = P \cup \{x \colon \tilde{s}_1 \triangleq \tilde{q}_1;\ X \colon \tilde{c}_1 \triangleq \tilde{d}_1;\ \mathfrak{a}_1\}$,
$S_0 = S_1$, $\vartheta_1 = \varepsilon$.
Trivial because $X(x)\vartheta_1^n = X(x)\vartheta_2^n$ and $S_0 = S_1$.

App-A. $P_0 = P \cup \{x \colon \tilde{s}_0 \triangleq \tilde{q}_0;\ X \colon \tilde{c}_0 \triangleq \tilde{d}_0;\ \mathfrak{a}_0\}$,
$P_1 = P \cup \{y \colon \tilde{s}_1 \triangleq \tilde{q}_1;\ Y : \circ \triangleq \circ;\ \mathfrak{a}_1\}$,
$S_0 = S_1 \backslash \{x \colon \epsilon \triangleq \epsilon;\ X \colon \tilde{c}_1 \triangleq \tilde{d}_1\}$,
$\vartheta_1 = \{x \mapsto a_1(Y(y))\}$.
We have $X(x)\vartheta_1^n = X(a_1(Y(y)))\vartheta_2^n$. By IH it holds $\mathfrak{P}(Y(y)\vartheta_2^n)$ and $\mathfrak{P}(X(\circ)\vartheta_2^n)$. It follows that all the properties also hold for $X(a_1(Y(y)))\vartheta_2^n$.

Sol-H. $P_0 = P_1 \cup \{x \colon \tilde{s} \triangleq \tilde{q};\ X : \circ \triangleq \circ;\ \mathfrak{e}\}$,
$S_0 = S_1 \backslash \{x \colon \tilde{s} \triangleq \tilde{q};\ X : \circ \triangleq \circ\}$,
$\vartheta_1 = \{X \mapsto \circ\}$.
Substitution composition gives $X(x)\vartheta_1^n = x\vartheta_2^n$ and by IH we have $\mathfrak{P}(x\vartheta_2^n)$.

Mer-S. $P_0 = P_1$,
$S_0 = S \cup \{x_1 \colon \tilde{s} \triangleq \tilde{q};\ X_1 \colon \tilde{c} \triangleq \tilde{d}\} \cup \{x_2 \colon \tilde{s} \triangleq \tilde{q};\ X_2 \colon \tilde{c} \triangleq \tilde{d}\}$,
$S_1 = S \cup \{x_1 \colon \tilde{s} \triangleq \tilde{q};\ X_1 \colon \tilde{c} \triangleq \tilde{d}\}$,
$\vartheta_1 = \{x_2 \mapsto x_1, X_2 \mapsto X_1\}$.
Hence $x_2\vartheta_1^n = x_1\vartheta_2^n$ and $X_2(\circ)\vartheta_1^n = X_1(\circ)\vartheta_2^n$. By IH holds that $\mathfrak{P}(x_1\vartheta_2^n)$ and $\mathfrak{P}(X_1(\circ)\vartheta_2^n)$.

Clr-S. $P_0 = P_1$,
$S_0 = S_1 \cup \{x \colon \epsilon \triangleq \epsilon;\ X : \circ \triangleq \circ\}$,
$\vartheta_1 = \{x \mapsto \epsilon, X \mapsto \circ\}$.
We have $x\vartheta_1^n = \varepsilon$ and $X(\circ)\vartheta_1^n = \circ$, which is trivial.

Res-C. $P_0 = P_1$,
$S_0 = S \cup \{x \colon \epsilon \triangleq \epsilon;\ X \colon (\tilde{s}_l, \dot{c}, \tilde{s}_r) \triangleq (\tilde{q}_l, \dot{d}, \tilde{q}_r)\}$
$S_1 = S \cup \{x \colon \epsilon \triangleq \epsilon; X \colon \dot{c} \triangleq \dot{d},$
$\qquad y \colon \tilde{s}_l \triangleq \tilde{q}_l; Y : \circ \triangleq \circ,\ z \colon \tilde{s}_r \triangleq \tilde{q}_r; Z : \circ \triangleq \circ\}$,
$\tilde{s}_l \neq \epsilon$ or $\tilde{s}_r \neq \epsilon$ or $\tilde{q}_l \neq \epsilon$ or $\tilde{q}_r \neq \epsilon$,
$\vartheta_1 = \{X \mapsto (y, X(\circ), z)\}$.
Therefore $x\vartheta_1^n = x\vartheta_2^n$ but $X(\circ)\vartheta_1^n = (y, X(\circ), z)\vartheta_2^n$. By IH we know that $\mathfrak{P}(y\vartheta_2^n)$, $\mathfrak{P}(X(\circ)\vartheta_2^n)$ and $\mathfrak{P}(z\vartheta_2^n)$ holds. Res-C is the only rule which maps a context variable to a hedge. The rule itself produces AUPs where all the contexts are terms (not hedges). Together with the condition $\tilde{s}_l \neq \epsilon$ or $\tilde{s}_r \neq \epsilon$ or $\tilde{q}_l \neq \epsilon$ or $\tilde{q}_r \neq \epsilon$ it follows that Res-C never applies twice for the same context variable X. This considerations guarantee that $X(\circ)\vartheta_2^n$ is a term (not a hedge) which implies that $\mathfrak{P}((y, X(\circ), z)\vartheta_2^n)$ holds.

Spl-H. $P_0 = P \cup \{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \tilde{c} \triangleq \tilde{d};\ \mathfrak{a}_0\}$

$P_1 = P \cup \{y\colon \tilde{s}|_{i_1}^{i_k} \triangleq \tilde{q}|_{j_1}^{j_k};\ Y\colon \circ \triangleq \circ;\ \mathfrak{a}_1,$
$z\colon \tilde{s}|_{i_k^{++}}^{i_m} \triangleq \tilde{q}|_{j_k^{++}}^{j_m};\ Z\colon \circ \triangleq \circ;\ \mathfrak{a}_2\},$

$\mathfrak{a}_0 = a_1\langle i_1{\cdot}I_1, j_1{\cdot}J_1\rangle \ldots a_{k+1}\langle i_{k+1}{\cdot}I_{k+1}, j_{k+1}{\cdot}J_{k+1}\rangle$
$\ldots a_m\langle i_m{\cdot}I_m, j_m{\cdot}J_m\rangle,$

$\mathfrak{a}_1 = a_1\langle (i_1 - i_1^{--}){\cdot}I_1, (j_1 - j_1^{--}){\cdot}J_1\rangle$
$\ldots a_k\langle (i_k - i_1^{--}){\cdot}I_k, (j_k - j_1^{--}){\cdot}J_k\rangle,$

$\mathfrak{a}_2 = a_{k+1}\langle (i_{k+1}{-}i_k){\cdot}I_{k+1}, (j_{k+1}{-}j_k){\cdot}J_{k+1}\rangle$
$\ldots a_m\langle (i_m{-}i_k){\cdot}I_m, (j_m{-}j_k){\cdot}J_m\rangle,$

$i_1 = i_k$ or $j_1 = j_k,$

$i_1 \neq i_{k+1}$ and $j_1 \neq j_{k+1},$

$S_0 = S_1\backslash\{x\colon \epsilon \triangleq \epsilon;\ X\colon \tilde{c}[\tilde{s}|_1^{i_1^{--}}, \circ, \tilde{s}|_{i_m^{++}}^{|\tilde{s}|}] \triangleq \tilde{d}[\tilde{q}|_1^{j_1^{--}}, \circ, \tilde{q}|_{j_m^{++}}^{|\tilde{q}|}]\},$

$\vartheta_1 = \{x \mapsto (Y(y), Z(z))\}.$

We have $X(x)\vartheta_1^n = X(Y(y), Z(z))\vartheta_2^n$. By IH it holds $\mathfrak{P}(Y(y)\vartheta_2^n)$, $\mathfrak{P}(Z(z)\vartheta_2^n)$ and $\mathfrak{P}(X(\circ)\vartheta_2^n)$. The condition $i_1 \neq i_{k+1}$ implicitly demands that there are at least two elements $a_1\langle i_1{\cdot}I_1, j_1{\cdot}J_1\rangle$ and $a_{k+1}\langle i_{k+1}{\cdot}I_{k+1}, j_{k+1}{\cdot}J_{k+1}\rangle$ in $\mathfrak{a}_0$ and it follows that both alignments $\mathfrak{a}_1$ and $\mathfrak{a}_2$ are nonempty. We already know from above that $Y(y)\vartheta_2^n$ and $Z(z)\vartheta_2^n$ are supporting generalizations which contain a_1 and a_{k+1} respectively. Now we will show that

- $Y(y)\vartheta_2^n$ is a term t_1 (not an arbitrary hedge),
- $Z(z)\vartheta_2^n$ is a hedge of the form $\tilde{r}, t_2$ where t_2 is not a hedge variable.

Then it follows that $\mathfrak{P}(t_1, \tilde{r}, t_2)$ holds. Furthermore we have $X(Y(y), Z(z))\vartheta_2^n = X(\circ)\vartheta_2^n[t_1, \tilde{r}, t_2]$ such that also $\mathfrak{P}(X(\circ)\vartheta_2^n[t_1, \tilde{r}, t_2])$ holds because t_1 and t_2 are nonempty terms different from a hedge variable.

Let $\tilde{s}|_{i_1}^{i_k} = (t_{i_1}, \ldots, t_{i_k})$ and $\tilde{q}|_{j_1}^{j_k} = (t_{j_1}, \ldots, t_{j_k})$ where $t_{i_1} = \tilde{s}|_{i_1}$, $t_{i_k} = \tilde{s}|_{i_k}$, $t_{j_1} = \tilde{s}|_{j_1}$, $t_{j_k} = \tilde{s}|_{j_k}$, then all the t's are terms (allowing $t_{i_1} = t_{i_k}$, $t_{j_1} = t_{j_k}$). Furthermore t_{i_1} and t_{j_1} contain the function symbol corresponding to $a_1\langle i_1{\cdot}I_1, j_1{\cdot}J_1\rangle$, and similarly for t_{i_k}, t_{j_k} and $a_k\langle i_k{\cdot}I_k, j_k{\cdot}J_k\rangle$. We know that $Y(y)\vartheta_2^n$ is a supporting generalization of $(t_{i_1}, \ldots, t_{i_k})$ and $(t_{j_1}, \ldots, t_{j_k})$ with respect to $\mathfrak{a}_1 = a_1\langle\ldots\rangle \ldots a_k\langle\ldots\rangle$ and therefore $Y(y)\vartheta_2^n$ is of the form $(\tilde{r}_1, t_{a_1}, \ldots, t_{a_k}, \tilde{r}_2)$ where t_{a_1}, t_{a_k} denote those terms which contain a_1 and a_k respectively (allowing $t_{a_1} = t_{a_k}$) and $\tilde{r}_1, \tilde{r}_2$ are arbitrary hedges. To finish this part we have to show that $\tilde{r}_1 = \tilde{r}_2 = \epsilon$. From above we also know that $Y(y)\vartheta_2^n\sigma_{\text{L}}(S_n) = (t_{i_1}, \ldots, t_{i_k})$ and therefore $\tilde{r}_1, \tilde{r}_2$ can only contain variables which are mapped to ϵ by $\sigma_{\text{L}}(S_n)$. Because of $\mathfrak{P}(Y(y)\vartheta_2^n)$ the hedges $\tilde{r}_1, \tilde{r}_2$ do not contain vertical chains of variables and therefore no context variable can appear there also there are no horizontal consecutive hedge variables. The only remaining possibility is that $\tilde{r}_1, \tilde{r}_2$ are either empty or a hedge variable. If both are empty then we are done thus we assume w.l.o.g. that $\tilde{r}_1 = x_1$ to get a contradiction. Then there is an AUP $\{x_1\colon \epsilon \triangleq \epsilon;\ X_1\colon \tilde{c}_1 \triangleq \tilde{d}_1\} \in S_n$. As X_1 does not appear below x_1 it must have been terminated by some mapping $\vartheta_k = \{X_1 \mapsto \circ\}$, $2 \leqslant k \leqslant n$. This implies that also $\tilde{c}_1 = \tilde{d}_1 = \circ$. By assumption the derivation is exhaustive but for $\{x_1\colon \epsilon \triangleq \epsilon;\ X_1\colon \circ \triangleq \circ\}$ the rule Clr-S is applicable which is a contradiction.

Let $\tilde{s}|_{i_k^{++}}^{i_m} = (\ldots, t_{i_m})$ and $\tilde{q}|_{j_k^{++}}^{j_m} = (\ldots, t_{j_m})$ where $t_{i_m} = \tilde{s}|_{i_m}$, $t_{j_m} = \tilde{s}|_{j_m}$, then t_{i_m} and t_{j_m} are terms. Furthermore t_{i_m} and t_{j_m} contain the function symbol corresponding to $a_m\langle i_m{\cdot}I_m, j_m{\cdot}J_m\rangle$. The further reasoning is the same as above. It follows that all the properties are preserved when composing $X(Y(y), Z(z))\vartheta_2^n$. □

The next theorem is the Completeness Theorem. It, essentially, says that for a given alignment of two input hedges, a rigid generalization which is computed by $\mathfrak{G}$ is least general among all rigid generalizations of the same input.

Theorem 11 (Completeness). *Let $\tilde{g}$ be a rigid generalization of $\tilde{s}$ and $\tilde{q}$ with respect to $\mathfrak{a}$. Then there exists a derivation $\{x\colon \tilde{s} \triangleq \tilde{q};\ X\colon \circ \triangleq \circ;\ \mathfrak{a}\};\ \varnothing;\ \varepsilon \Longrightarrow^{+} \varnothing;\ S;\ \sigma$ obtained by $\mathfrak{G}$ such that $\tilde{g} \preceq X(x)\sigma$.*

Proof. Let ϑ_1, ϑ_2, σ be substitutions, $x \in \mathcal{V}_{\mathsf{H}}$, $X \in \mathcal{V}_{\mathsf{C}}$ fresh variables and S a set of AUPs such that:

- $\mathrm{Dom}(\vartheta_1) = \mathrm{Dom}(\vartheta_2)$,
- $\tilde{g}\vartheta_1 = \tilde{s}$,
- $\tilde{g}\vartheta_2 = \tilde{q}$,
- $\sigma = \{x \mapsto \tilde{g}, X \mapsto \circ\}$,
- $S = \{y\colon y\vartheta_1 \triangleq y\vartheta_2;\ Y\colon \circ \triangleq \circ \mid y \in \mathrm{Dom}(\vartheta_1), Y \text{ is fresh}\} \cup$
  $\{y\colon \epsilon \triangleq \epsilon;\ Y\colon Y\vartheta_1 \triangleq Y\vartheta_2 \mid Y \in \mathrm{Dom}(\vartheta_1), y \text{ is fresh}\}$.

We construct the final state $P;\ S;\ \sigma$ where $P = \varnothing$, $X(x)\sigma = \tilde{g}$ and furthermore we have $\tilde{s} = X(x)\sigma\vartheta_1 = X(x)\sigma\sigma_{\mathrm{L}}(S)$ and $\tilde{q} = X(x)\sigma\vartheta_2 = X(x)\sigma\sigma_{\mathrm{R}}(S)$. We start from this final state and we show that a rule R is applicable in reverse direction $P_0;\ S_0;\ \sigma_0 \ \overset{\mathsf{R}}{\Longleftarrow}\ P_1; S_1; \sigma_1$ until we reach the state $\{x_0\colon \tilde{s}_0 \triangleq \tilde{q}_0;\ X_0\colon \circ \triangleq \circ;\ \mathfrak{a}_0\} \in P_0$ and $\tilde{g}_0 = X_0(x_0)\sigma_0 = X_0(x_0)$, such that, by the invariant Lemma 7, we will get the desired result.

First, all the variables in $\tilde{g}_1$ are made distinct by reversely applying the rule Mer-S. We also keep the store sound like described in the Mer-S rule. Now we introduce some additional fresh variables, denoted by Y and y, to ensure that above every function symbol there is a context variable and that every leaf is a hedge variable, such that $\tilde{g}_0$ is still a rigid generalization of $\tilde{s}_1$ and $\tilde{q}_1$ with respect to $\mathfrak{a}_1$. This can be done by applying the rule Clr-S in reverse and we use the following transformation rules to build $\tilde{g}_0 = \phi(\tilde{g}_1)$:

$\phi(\tilde{s}) = Y(\psi(\tilde{s}))$, if $\mathrm{Head}(\tilde{s}) \in \mathcal{F}$,	$\phi(\tilde{s}) = \psi(\tilde{s})$, if $\mathrm{Head}(\tilde{s}) \notin \mathcal{F}$.
$\psi(x) = x$, $\quad \psi(a) = a(y)$,	$\psi(f(t_1, \ldots, t_n))_{n>0} = f(\phi(t_1), \ldots, \phi(t_n))$,
$\psi(X(\tilde{s})) = X(\psi(\tilde{s}))$,	$\psi(t_1, \ldots, t_n)_{n>1} = \phi(t_1), \ldots, \phi(t_n)$.

The next step is to apply the rule Res-C in reverse direction as long as possible, leaving only those hedge variables which occur as singleton terms under a function symbol, e.g., those which occur in subterms of the form $f(x)$. We still have $\tilde{g}_0$ being a rigid generalization of $\tilde{s}_1$ and $\tilde{q}_1$ with respect to $\mathfrak{a}_1$.

Now we move all the AUPs $x_1\colon \tilde{s}_1 \triangleq \tilde{q}_1;\ X_1\colon \circ \triangleq \circ$ with $x_1 \in \mathcal{V}_{\mathsf{H}}(\tilde{g})$ from S_1 to P_0 by reversely applying the rule Sol-H. After this step, all the leafs are

hedge variables and together with their two predecessors they are subterms of the form $f(X_1(x_1))$. There is one AUP in P_0 for every subterm (leaf) of this form.

We will show that one of the rules App-A, Abs-L/Abs-R, Spl-H is applicable in reverse direction until we get $P_0 = \{x_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0;\ X_0 \colon \circ \triangleq \circ;\ \mathfrak{a}_0\}$ and $X_0(x_0)\sigma_0 = X_0(x_0)$.

First, the rule App-A is applicable in reverse to every $\{x_1 \colon \tilde{s}_1 \triangleq \tilde{q}_1;\ X_1 \colon \circ \triangleq \circ;\ \mathfrak{e}\} \in P$. It adds one function symbol to the alignment $\mathfrak{a}_0$ and removes it from $\tilde{g}_1$. Furthermore, the hole $\circ$ in $\tilde{c}_0, \tilde{d}_0$ has no siblings for all the new AUPs $\{x_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0;\ X_0 \colon \tilde{c}_0 \triangleq \tilde{d}_0;\ \mathfrak{a}_0\} \in P_0$. Every reverse application of App-A strictly decreases the size of $\tilde{g}_0$. Afterwards all the leafs are of the form $X_0(x_0)$ and there is an AUP $\{x_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0;\ X_0 \colon \tilde{c}_0 \triangleq \tilde{d}_0;\ \mathfrak{a}_0\} \in P_0$, where the hole $\circ$ in $\tilde{c}_0, \tilde{d}_0$ has no siblings.

- *Case $\tilde{c}_1 \neq \circ$ or $\tilde{d}_1 \neq \circ$:* By looking at the rules App-A, Abs-L/Abs-R, Spl-H it is trivial to see that, if $\tilde{c}_1 \neq \circ$ or $\tilde{d}_1 \neq \circ$ then the only possible reverse application is Abs-L/Abs-R. Furthermore the rule is always applicable in this situation because the property that the hole $\circ$ in $\tilde{c}_1, \tilde{d}_1$ has no siblings is maintained by every reverse rule application and from that it follows that a function is applied to either $\tilde{c}_1$ or $\tilde{d}_1$. Abs-L/Abs-R maintains the property that the hole $\circ$ in $\tilde{c}_0, \tilde{d}_0$ has no siblings, such that we have the same situation with the possible rule applications App-A, Abs-L/Abs-R, Spl-H as above. Obviously Abs-L/Abs-R can only be applied finitely many times because it strictly decreases the size of $\tilde{c}_0$ or $\tilde{d}_0$.
- *Case $\tilde{c}_1 = \tilde{d}_1 = \circ$ and X_1 has no siblings:* Then there are two possibilities: Either $\tilde{g}_1 = X_1(x_1)$ and we are done or a function is applied to $X_1(x_1)$ like $f(X_1(x_1))$, such that the rule App-A is applicable again (see above).
- *Case $\tilde{c}_1 = \tilde{d}_1 = \circ$ and X_1 has siblings:* Let $X_1, \ldots, X_n, n > 1$ be the siblings. If $n = 2$ and $\tilde{g}$ is of the form $X_0(X_1(x_1), X_2(x_2))$, then Spl-H can directly be applied in reverse order. Otherwise we use Clr-S to introduce a fresh context variable Y above X_{n-1} and X_n leading to the form $X_1, \ldots, X_{n-2}, Y(X_{n-1}(x_{n-1}), X_n(x_n))$ such that Spl-H is applicable in reverse order to $Y(X_{n-1}(x_{n-1}), X_n(x_n))$. This strictly decreases the size of $\tilde{g}_0$.

Eventually we get $\tilde{g}_0$ being of the form $X_0(x_0)$ when applying this strategy exhaustively. And we know that $\{x_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0;\ X_0 \colon \circ \triangleq \circ;\ \mathfrak{a}_0\} \in P_0$. By the invariant Lemma 7 we also know that $\tilde{s} = X(x)\sigma_0\sigma_{\text{L}}(P_0 \cup S_0) = X_0(x_0)\sigma_{\text{L}}(P_0 \cup S_0)$ and $\tilde{q} = X(x)\sigma_0\sigma_{\text{R}}(P_0 \cup S_0) = X_0(x_0)\sigma_{\text{R}}(P_0 \cup S_0)$. This gives us the result $\tilde{s}_0 = \tilde{s}$ and $\tilde{q}_0 = \tilde{q}$ which proves the existence of a derivation in $\mathfrak{G}$. □

The algorithm is non-deterministic. The Uniqueness Theorem says that different transformations compute generalizations which are equivalent modulo $\simeq$, i.e., differ from each other only by variable renaming:

Theorem 12 (Uniqueness modulo $\simeq$). *Let $\mathfrak{a}$ be an admissible alignment of $\tilde{s}$ and $\tilde{q}$. If $\{x_1 \colon \tilde{s} \triangleq \tilde{q};\ X_1 \colon \circ \triangleq \circ;\ \mathfrak{a}\};\ \varnothing;\ \varepsilon \Longrightarrow^{+} \varnothing;\ S_1;\ \sigma_1$ and $\{x_2 \colon \tilde{s} \triangleq \tilde{q};$*

$X_2 \colon \circ \triangleq \circ;\ \mathfrak{a}\};\ \varnothing;\ \varepsilon \Longrightarrow^{+} \varnothing;\ S_2;\ \sigma_2$ *are two exhaustive derivations in* $\mathfrak{G}$*, then* $X_1(x_1)\sigma_1 \simeq X_2(x_2)\sigma_2$.

Proof. By Newman's lemma [21] and Theorem 9, it suffices to show local confluence. Let $P_0;\ S_0;\ \sigma_0$ be an arbitrary state in $\mathfrak{G}$. We show that for any two rule applications $P_0;\ S_0;\ \sigma_0 \Longrightarrow_{\mathsf{R}} P_1; S_1; \sigma_0\vartheta_1$ and $P_0;\ S_0;\ \sigma_0 \Longrightarrow_{\mathsf{R}'} P_1'; S_1'; \sigma_0\vartheta_1'$ there are derivations such that $P_1; S_1; \sigma_0\vartheta_1 \Longrightarrow^{*} P_i; S_i; \sigma_i \ {}^{*}\!\Longleftarrow P_1'; S_1'; \sigma_0\vartheta_1'$. It is easy to see that if the rules R and R' operate on different AUPs then it holds, for $P_1; S_1; \sigma_0\vartheta_1 \Longrightarrow_{\mathsf{R}'} P_2; S_2; \sigma_0\vartheta_1\vartheta_2$ and $P_1'; S_1'; \sigma_0\vartheta_1' \Longrightarrow_{\mathsf{R}} P_2'; S_2'; \sigma_0\vartheta_1'\vartheta_2'$, that $P_2 = P_2'$, $S_2 = S_2'$ and $\vartheta_1\vartheta_2 = \vartheta_1'\vartheta_2'$ (using the corresponding names for fresh variables), because the variables of the AUPs are disjoint (by Lemma 5). Therefore we assume that both, R and R', operate on an arbitrary but fixed AUP $\{x_0 \colon \tilde{s}_0 \triangleq \tilde{q}_0;\ X_0 \colon \tilde{c}_0 \triangleq \tilde{d}_0;\ \mathfrak{a}_0\} \in P_0 \cup S_0$.

If R is one of Spl-H, App-A, Sol-H, then no other rule is applicable to the selected AUP and it follows that $\mathsf{R}' = \mathsf{R}$. If $\mathsf{R} =$ Abs-L then either $\mathsf{R}' = \mathsf{R}$ or $\mathsf{R}' =$ Abs-R. The first case is the trivial one. In the latter case, the rules Abs-L and Abs-R operate on different sides of the equations such that it does not matter which one is applied first, if both are applicable at the same time. Therefore we obtain for $P_1; S_1; \sigma_0 \Longrightarrow_{\mathsf{R}'} P_2; S_2; \sigma_0$ and $P_1'; S_1'; \sigma_0 \Longrightarrow_{\mathsf{R}} P_2'; S_2'; \sigma_0$, that $P_2 = P_2'$ and $S_2 = S_2'$. Obviously the same reasoning holds for $\mathsf{R} =$ Abs-R. It remains to show local confluence for the cases $\mathsf{R} =$ Res-C, $\mathsf{R} =$ Clr-S and $\mathsf{R} =$ Mer-S.

R = Res-C. $S_0 = S \cup \{x_1 \colon \epsilon \triangleq \epsilon;\ X_1 \colon (\tilde{s}_l, \dot{c}, \tilde{s}_r) \triangleq (\tilde{q}_l, \dot{d}, \tilde{q}_r)\}$,
$S_1 = S \cup \{x_1 \colon \epsilon \triangleq \epsilon; X_1 \colon \dot{c} \triangleq \dot{d}\} \cup$
$\{y_1 \colon \tilde{s}_l \triangleq \tilde{q}_l; Y_1 \colon \circ \triangleq \circ, z_1 \colon \tilde{s}_r \triangleq \tilde{q}_r; Z_1 \colon \circ \triangleq \circ\}$,
$\tilde{s}_l \neq \epsilon$ or $\tilde{s}_r \neq \epsilon$ or $\tilde{q}_l \neq \epsilon$ or $\tilde{q}_r \neq \epsilon$,
$\vartheta_1 = \{X_1 \mapsto (y_1, X_1(\circ), z_1)\}$.

The rule Clr-S is not applicable for the selected AUP because of the condition $\tilde{s}_l \neq \epsilon$ or $\tilde{s}_r \neq \epsilon$ or $\tilde{q}_l \neq \epsilon$ or $\tilde{q}_r \neq \epsilon$. Therefore the only nontrivial case is $\mathsf{R}' =$ Mer-S, and from the condition of Mer-S we know that there is an AUP $\{x_2 \colon \epsilon \triangleq \epsilon;\ X_2 \colon (\tilde{s}_l, \dot{c}, \tilde{s}_r) \triangleq (\tilde{q}_l, \dot{d}, \tilde{q}_r)\} \in S$ which is the second one selected by Mer-S. To show local confluence we select this AUP and apply Res-C again $P_1; S_1; \sigma_0\vartheta_1 \Longrightarrow_{\text{Res-C}} P_2; S_2; \sigma_0\vartheta_1\vartheta_2$, with $\vartheta_2 = \{X_2 \mapsto (y_2, X_2(\circ), z_2)\}$ and $P_2 = P_1 = P_0$. Furthermore we get $\{x_2 \colon \epsilon \triangleq \epsilon; X_2 \colon \dot{c} \triangleq \dot{d}, y_2 \colon \tilde{s}_l \triangleq \tilde{q}_l; Y_2 \colon \circ \triangleq \circ, z_2 \colon \tilde{s}_r \triangleq \tilde{q}_r; Z_2 \colon \circ \triangleq \circ\} \subset S_2$. W.l.o.g. let $P_0;\ S_0;\ \sigma_0 \Longrightarrow_{\mathsf{R}'} P_1'; S_1'; \sigma_0\vartheta_1'$ such that $P_1' = P_0$, $S_1' = S$ and $\vartheta_1' = \{x_1 \mapsto x_2, X_1 \mapsto X_2\}$. Now we continue with $P_2; S_2; \sigma_0\vartheta_1\vartheta_2$ and apply the rule Mer-S three times to obtain $P_2; S_2; \sigma_0\vartheta_1\vartheta_2 \Longrightarrow^{3}_{\text{Mer-S}} P_i; S_i; \sigma_0\vartheta_1\vartheta_2\{x_1 \mapsto x_2, X_1 \mapsto X_2\}\{y_1 \mapsto y_2, Y_1 \mapsto Y_2\}\{z_1 \mapsto z_2, Z_1 \mapsto Z_2\}$. Finally we apply to $P_1'; S_1'; \sigma_0\vartheta_1'$ the rule Res-C such that we get $P_1'; S_1'; \sigma_0\vartheta_1' \Longrightarrow_{\text{Res-C}} P_i; S_i; \sigma_0\vartheta_1'\{X_2 \mapsto (y_2, X_2(\circ), z_2)\}$. It remains to compare the two obtained substitutions $\sigma_i = \{X_1 \mapsto (y_1, X_1(\circ), z_1)\}\{X_2 \mapsto (y_2, X_2(\circ), z_2)\}\{x_1 \mapsto x_2, X_1 \mapsto X_2\}\{y_1 \mapsto y_2, Y_1 \mapsto Y_2\}\{z_1 \mapsto z_2, Z_1 \mapsto Z_2\}$ and $\sigma_i' = \{x_1 \mapsto x_2, X_1 \mapsto X_2\}\{X_2 \mapsto (y_2, X_2(\circ), z_2)\}$. Therefore we compose both substitutions exhaustively: $\sigma_i = \{x_1 \mapsto x_2, X_1 \mapsto (y_2, X_2(\circ), z_2), X_2 \mapsto (y_2, X_2(\circ), z_2), y_1 \mapsto y_2, Y_1 \mapsto Y_2, z_1 \mapsto z_2, Z_1 \mapsto Z_2\}$ and

$\sigma_i' = \{x_1 \mapsto x_2, X_1 \mapsto (y_2, X_2(\circ), z_2), X_2 \mapsto (y_2, X_2(\circ), z_2)\}$. As the fresh variables y_1, Y_1, z_1, Z_1 have been introduced during this reasoning process and all the AUPs containing one of those variables have been eliminated by applications of the rule Mer-S, the extra mappings in σ_i do not have any effect and may also be omitted.

R = Clr-S. $S_1 = S_0 \backslash \{x_1 \colon \epsilon \triangleq \epsilon;\ X_1 \colon \circ \triangleq \circ\}$,
$\vartheta_1 = \{x_1 \mapsto \epsilon, X_1 \mapsto \circ\}$.

For the same reason as above, the rule Res-C is not applicable for the selected AUP, such that the only nontrivial case is R′ = Mer-S. The reasoning can be done similarly to the case R = Res-C. We leave this easy exercise to the reader.

R = Mer-S. $S_0 = S \cup \{x_1 \colon \tilde{s} \triangleq \tilde{q};\ X_1 \colon \tilde{c} \triangleq \tilde{d}\} \cup \{x_2 \colon \tilde{s} \triangleq \tilde{q};\ X_2 \colon \tilde{c} \triangleq \tilde{d}\}$,
$S_1 = S \cup \{x_2 \colon \tilde{s} \triangleq \tilde{q};\ X_2 \colon \tilde{c} \triangleq \tilde{d}\}$,
$\vartheta_1 = \{x_1 \mapsto x_2, X_1 \mapsto X_2\}$.

If R = Mer-S then there are the three cases R′ = R, R′ = Res-C and R′ = Clr-S. The first case is the trivial one and we already showed local confluence for the other two cases. □

Theorem 13 (Complexity). *The anti-unification algorithm* $\mathfrak{G}$ *has* $O(n^2)$ *time complexity and* $O(n)$ *space complexity, where* n *is the number of symbols in the input.*

Proof. Let $P_0; S_0; \sigma_0 = \{x \colon \tilde{s} \triangleq \tilde{q};\ X \colon \circ \triangleq \circ;\ \mathfrak{a}\}; \varnothing; \varepsilon$ be the initial state of $\mathfrak{G}$ and $P_{i-1}; S_{i-1}; \sigma_{i-1} \Longrightarrow P_i; S_i; \sigma_i$ an arbitrary rule application. By Theorem 12 we can arrange the rule applications as we like to obtain a maximal derivation. First the rules Spl-H, Abs-L/Abs-R, App-A and Sol-H are applied exhaustively. This are the only rules that operate on P_{i-1} and furthermore they do not have conditions on S_{i-1} or σ_{i-1} such that $P_0; S_0; \sigma_0 \Longrightarrow^+ \varnothing; S_j; \sigma_j$, for some j. Afterwards they are not applicable again and Res-C is applied exhaustively $\varnothing; S_j; \sigma_j \Longrightarrow^*_{\text{Res-C}} \varnothing; S_k; \sigma_k$. It transforms all the contexts in the store to terms. The rules Clr-S and Mer-S operate on S_k but they only remove AUPs from there, such that Res-C will not be applicable again. Finally we postpone the application of Mer-S to the very end, leading to a partial derivation $\varnothing; S_k; \sigma_k \Longrightarrow^*_{\text{Clr-S}} \varnothing; S_l; \sigma_l \Longrightarrow^*_{\text{Mer-S}} \varnothing; S_n; \sigma_n$ where no more rule is applicable because Mer-S does not introduce any AUPs, to which another rule could apply.

Now we analyze the first phase $P_0; S_0; \sigma_0 \Longrightarrow^+ \varnothing; S_j; \sigma_j$. The rule Spl-H splits an AUP into two AUPs and moves some parts into the store. The space overhead for one application is constant because the two new AUPs in P_i and the one in S_i together exactly cover the original one from P_{i-1}, and four new variables are introduced. It can be applied $O(n)$ many times because both of the new AUPs are nonempty. It needs linear time (by the length of the alignment) to check for applicability and find the position for splitting the AUP. Also the context application needs linear time. The rules Abs-L/Abs-R are also applicable $O(n)$ many times. They strictly reduce the size of a hedge in P_i. The space overhead is zero. The test for applicability, the context application as well as

the operations on the alignment need linear time. App-A is applicable $O(n)$ many times as well and one application needs linear time and constant space. It strictly reduces the size of a hedge in P_i and one application needs linear time, for the same reasons as the above rules. As Spl-H is applicable at most $O(n)$ many times and doubles the elements of P_i at each application and all the other rules do not increase the length of P_i, Sol-H is applicable $O(n)$ many times too. It follows that the number of introduced variables is $O(n)$ and the size of S_j is also bound by $O(n)$.

We compose the substitution σ_i immediately, but we only keep the mappings for x and X in σ_i, such that $\sigma_i = \{x \mapsto \tilde{r}_i, X \mapsto \tilde{c}_i\}$, for some $\tilde{r}_i, \tilde{c}_i$. As all the introduced variables in Spl-H and App-A are fresh, they only appear once in $\tilde{r}_i$ or $\tilde{c}_i$. This invariant of the first phase leads to $O(n)$ size of σ_i as well as $O(n)$ time for the substitution composition in Spl-H, App-A and Sol-H. All together we get $O(n^2)$ time complexity and $O(n)$ space complexity for the first phase.

The second phase is $\varnothing;\ S_j;\ \sigma_j \Longrightarrow^*_{\mathsf{Res\text{-}C}} \varnothing;\ S_k;\ \sigma_k$. The rule Res-C is applicable only once per AUP leading to $O(n)$ many applications. The space overhead is constant at each application, introducing four fresh variables. It needs linear time at each application. We again compose σ_i immediately and for similar reasons as above, the substitution composition in Res-C only needs $O(n)$ time, leading to an overall time complexity of $O(n^2)$ and space complexity $O(n)$.

From the $O(n)$ size of the store, it follows that also the store cleaning rule is applicable $O(n)$ many times and the overall time complexity of this phase is $O(n^2)$, as we compose substitutions immediately like before. The space overhead for Clr-S is zero.

It remains to show that $\varnothing;\ S_l;\ \sigma_l \Longrightarrow^*_{\mathsf{Mer\text{-}S}} \varnothing;\ S_n;\ \sigma_n$ only needs $O(n^2)$ time. Therefore we postpone substitution composition. Comparing $O(n) * O(n)$ AUPs in the store needs $O(n^2)$ time and removing an AUP from the store needs constant time using a linked list. As the size of the store is bound by $O(n)$ and Mer-S removes one AUP at each application, there are $O(n)$ postponed substitution compositions. Each of them of constant size as they all are just variable renamings. This leads to linear space overhead and we have to compose $O(n)$ substitutions where each composition needs $O(n)$ time.

This concludes our complexity analysis where we showed that the algorithm runs in $O(n^2)$ time using $O(n)$ space. □

5. Implementation

The rule based system $\mathfrak{G}$ has been implemented in Java. It is freely available for download and there also exists a convenient web interface to try out the algorithm online. The web address is

`http://www.risc.jku.at/projects/stout/software/urauc.php`.

From this website, the user may also download a shell version, the source code, or a Java library to embed the algorithm in her/his own project. A sample code of the latter option is also available from the Web.

The library implementation uses existing Java classes for common data structures, like `ArrayList`, `Deque`, or `HashMap`, but the creation of new structures is done by a factory class named `DataStructureFactory`. Therefore, the user is free to choose an arbitrary implementation of all these common data structures, which might have some advantages on the performance of the provided algorithm.

Implemented Strategy. Before we demonstrate the usage of the library on some Java code fragments, we give an overview of the strategy which is used in the implementation.

```
1  INPUT:
2    E ← Given set of hedge equations {s̃_1 ≙ q̃_1, ..., s̃_n ≙ q̃_n}
3    𝔉_𝔞 ← Given alignment computation function
4    𝔉_c ← Given callback function
5  COMPUTE:
6    For each equation s̃ ≙ q̃ in E
7      For each admissible alignment 𝔞 in 𝔉_𝔞(s̃, q̃)
8        System = (P, S, σ) ← ({x: s̃ ≙ q̃; X: ∘ ≙ ∘; 𝔞}, ∅, {x ↦ X(x)})
9        While System.P ≠ ∅
10         Fix AUP p_i ← x_i: s̃_i ≙ q̃_i; X_i: c̃_i ≙ d̃_i; 𝔞_i from System.P
11         If 𝔞_i = 𝔢
12           Transform System ⟹^{p_i}_{Sol-H} System
13         Else If 𝔞_i = a_1⟨j_1, j_2⟩ ... a_m⟨j_1·I_m, j_2·J_m⟩ with j_1, j_2 ∈ ℕ
14           Transform System ⟹^{p_i}_{App-A} System
15         Else If 𝔞_i = a_1⟨j·I_1, J_1⟩ ... a_m⟨j·I_m, J_m⟩ with I_1 ≠ λ
16           Transform System ⟹^{p_i}_{Abs-L} System
17         Else If 𝔞_i = a_1⟨I_1, j·J_1⟩ ... a_m⟨I_m, j·J_m⟩ with J_1 ≠ λ
18           Transform System ⟹^{p_i}_{Abs-R} System
19         Else
20           Transform System ⟹^{p_i}_{Spl-H} System
21       For each s_i = {x_i: s̃_i ≙ q̃_i; X_i: c̃_i ≙ d̃_i} in System.S
22         If | c̃_i | > 1 or | d̃_i | > 1
23           Transform System ⟹^{s_i}_{Res-C} System
24       For each s_i = {x_i: s̃_i ≙ q̃_i; X: c̃_i ≙ d̃_i} in System.S
25         If s̃_i = q̃_i = ϵ_i & c̃_i = d̃_i = ∘
26           Transform System ⟹^{s_i}_{Clr-S} System
27       For each s_i = {x_i: s̃_i ≙ q̃_i; X_i: c̃_i ≙ d̃_i} in System.S
28         For each s_j = {x_j: s̃_j ≙ q̃_j; X_j: c̃_j ≙ d̃_j} in System.S
29           If s_i ≠ s_j & s̃_i = s̃_j & q̃_i = q̃_j & c̃_i = c̃_j & d̃_i = d̃_j
30             Transform System ⟹^{s_i, s_j}_{Mer-S} System
31       𝔉_c(System, x) invoke call back function
```

Listing 3: The implemented strategy for $\mathfrak{G}$.

The lines 1-4 in Listing 3 define the input assumptions. The alignment computation is a parameter of the algorithm and has to be provided by the user. Two prototype implementations are available from the library. For each AUP in the input set E, the alignment computation function will be invoked (line 6). Its return type is an `Iterator`, from which admissible alignments are pulled one by one (line 7). Line 8 shows how the initial system is constructed from one hedge equation and one alignment. The library uses a customizable factory pattern, the so called class `NodeFactory`, to generate fresh variables, e.g. x, X.

Afterwards the rules Sol-H, App-A, Abs-L, Abs-R, and Spl-H are applied exhaustively, as shown at the lines 9-20. For each iteration, an arbitrary AUP from P is fixed and a rule which is applicable for this AUP is determined.

For the following transformation of the store (lines 21-30), the rules Res-C, Clr-S, and Mer-S, are applied exhaustively, one after another. (Note that Res-C might introduce empty AUPs. Therefore Clr-S applications have to follow.)

Line 31 shows how the callback function, which is provided by the user, is invoked, following an exhaustive system transformation. The user is free to process this result further as he/she needs. Obviously, `System.`σ applied to x gives the computed generalization, but also the store is available via `System.`S.

Embedding the Library. To embed $\mathfrak{G}$ into another Java program, the first thing to do is to create an `EquationSystem` of type `AntiUnifyProblem`. Listing 4 shows how to create a new equation system.

```
1  eqSys = new EquationSystem<AntiUnifyProblem>() {
2    public AntiUnifyProblem newEquation() {
3      return new AntiUnifyProblem();
4    }
5  };
```

Listing 4: Creation of an equation system of type AntiUnifyProblem.

After instantiating the equation system, some equations should be added to the empty system. A convenient way to do this is by using the class `InputParser`, as shown in Listing 5. It takes two arbitrary `Reader` instances, e.g. `in1` and `in2`, each representing one input hedge, creates a new `Equation`, and adds the equation to a given set of equations, e.g. `eqSys`. Depending on the context of the library integration, those hedges may also be composed manually.

```
1  new InputParser<>(eqSys).parseHedgeEquation(in1,in2);
```

Listing 5: Parsing two given hedges in1 and in2, and putting them into eqSys.

For invoking the main algorithm $\mathfrak{G}$, we assume that the variable `aFnc` is an alignment computation function. All alignment computation functions are of the abstract type `AlignFnc`. The library offers two such functions: The first one, called `AlignFncLAA`, computes longest admissible alignments (see section 6). The other one is `AlignFncInput` and can be used to specify a certain admissible alignment for one given anti-unification equation. `AlignFncInput` only

works for a singleton equation system. Listing 6 shows how the anti-unification algorithm can be invoked using the built equation system.

```
1 new AntiUnify(aFnc, eqSys, DebugLevel.SILENT) {
2   public void callback(AntiUnifySystem sy,Variable x){
3     System.out.println(sy.getSigma().get(x));
4   };
5 }.antiUnify(true, false, null);
```

Listing 6: Sample code for calling the algorithm $\mathfrak{G}$.

The third argument of `AntiUnify` (line 1) defines the verbosity of the computation. For production use, normally one wants to silently compute the generalization. Whoever, if the debug level differs from `SILENT`, then also a `PrintStream` has to be provided in line 5 as third argument instead of `null`. By the first argument of the method `antiUnify`, the alignment iteration can be turned off, such that only one generalization is computed for each equation. The second argument in line 5 specifies, whether or not to justify the computed generalization. The justification works in the following way:

Let $\tilde{g}$ be a generalization of an input equation $\tilde{s} \triangleq \tilde{q}$ obtained by our anti-unification algorithm, and let S be the corresponding store. (In the demonstration code the store can be obtained by `sy.getStore()`.) The recorded differences in S are used to obtain two substitutions $\sigma_{\text{L}}(S)$ and $\sigma_{\text{R}}(S)$. The justification fails if either $\tilde{g}\sigma_{\text{L}}(S) \neq \tilde{s}$ or $\tilde{g}\sigma_{\text{R}}(S) \neq \tilde{q}$.

Last but not least, the lines 2-4 show a very simple implementation of a callback function, which prints the generalization $x\sigma$ to the standard output stream, for each generalization variable x which corresponds to one of the input equations.

6. Computing Admissible Alignments

The algorithm $\mathfrak{G}$ is independent from the alignment computation function. However, from the application point of view it is interesting to experiment with various such functions and identify practically well-behaving ones. It is not our goal to give here a survey of possible alignment computation functions. We just mention that to have $O(n^2)$ runtime complexity for computing higher-order generalizations (the best we can hope for due to Theorem 13), we could use some known techniques to compute an admissible alignment in quadratic time, e.g., as a constrained longest common subforest [2, 29] or an agreement subhedge/subtree [14, 22].

In this section we just want to give an idea of alignment computation by discussing a certain example, which is not very efficient, but it is simple and is implemented in our Java library. The function computes admissible alignments of *longest length* for two given input hedges.

Since an alignment can be seen as a common subsequence of the word representation of two input hedges (see Definition 9), computing the complete set of all admissible alignments can be done by a generate and test method: generate

all the common subsequences of the word representation of the input hedges and test them for admissibility (see Definition 11). However, this approach is not tractable because the number of subsequences of one sequence is already exponential in its length. Therefore it makes sense to reduce the number of generated common subsequences. One approach might be to compute some admissible alignments of longest length for two input hedges. (Notice, that this approach will not lead to a complete set of rigid lggs because there might be incomparable rigid lggs which contain less common function symbols.)

Definition 18 (Longest admissible alignment). A *longest admissible alignment* (laa) of two hedges is their admissible alignment with a longest length. I.e., an admissible alignment $\mathfrak{a}$ of two hedges $\tilde{s}$ and $\tilde{q}$ is an laa iff $|\mathfrak{a}| \geqslant |\mathfrak{a}'|$ for all admissible alignments $\mathfrak{a}'$ of the hedges $\tilde{s}$ and $\tilde{q}$.

A complete set of longest admissible alignments can be computed with the aid of a function that for two hedges, $\tilde{s}$ and $\tilde{q}$, returns a set of alignments of length $k \in \mathbb{N}$. We denote this function by $\mathfrak{Cs}(\tilde{s}, \tilde{q}, k)$ because it computes common subsequences of length k for the word representations of the input hedges $\tilde{s}$ and $\tilde{q}$. Given a set of alignments A, the subset of all admissible alignments is denoted by admissible(A). We formulate a simple generate and test algorithm by setting k initially to the length of the longest alignment (see, e.g., [17]) of two input hedges $\tilde{s}$ and $\tilde{q}$, and successively reducing k until we get a *nonempty* subset of *admissible* alignments from $\mathfrak{Cs}(\tilde{s}, \tilde{q}, k-i)$, for some $0 \leqslant i \leqslant k$. The algorithm can be described by four simple steps:

1. $k :=$ Length of longest alignment of $\tilde{s}$ and $\tilde{q}$.
2. $A := \mathfrak{Cs}(\tilde{s}, \tilde{q}, k)$.
3. If admissible(A) $= \varnothing$ then
   $k := k - 1$ and goto step 2.
4. return admissible(A).

This approach can still lead to iterating exponentially many alignments for two given input hedges. Consider a longest alignment of length k which is not admissible. By successively reducing the length to $k - i$, there are $\binom{k}{k-i}$ possible alignments which have to be tested for admissibility. However, our implementation (see section 5) offers this approach as an example of computing a set of admissible alignments for two input hedges.

7. Final Remarks

The algorithm $\mathfrak{G}$ presented in this paper computes a rigid lgg, which corresponds to a given admissible alignment for two given hedges. The vertical differences are abstracted by context variables, and the horizontal ones by hedge variables. The algorithm runs in quadratic time and requires linear space with respect to the size of the input.

The algorithm works for one admissible alignment, but an alignment computation function may return a finite set A of admissible alignments for the given

input hedges. It may even happen that one alignment in A is a subsequence of another one. To compute a minimal set of lggs for two input hedges and a given set of alignments, we need to solve a higher order matching problem for hedges, instantiating context and hedge variables. Design of such an algorithm is left for a future work.

Acknowledgments

This research has been partially supported by the Austrian Science Fund (FWF) with the project SToUT (P 24087-N18), and by the strategic program "Innovatives OÖ 2010plus" by the Upper Austrian Government.

References

[1] M. Alpuente, S. Escobar, J. Espert, and J. Meseguer. A modular order-sorted equational generalization algorithm. *Information and Computation*, 235(0):98 – 136, 2014. ISSN 0890-5401. doi: http://dx.doi.org/10.1016/j.ic.2014.01.006. Special issue on Functional and (Constraint) Logic Programming.

[2] A. Amir, T. Hartman, O. Kapah, B. R. Shalom, and D. Tsur. Generalized LCS. *Theor. Comput. Sci.*, 409(3):438–449, 2008.

[3] E. Armengol and E. Plaza. Bottom-up induction of feature terms. *Machine Learning*, 41(3):259–294, 2000.

[4] F. Baader. Unification, weak unification, upper bound, lower bound, and generalization problems. In R. V. Book, editor, *RTA*, volume 488 of *Lecture Notes in Computer Science*, pages 86–97. Springer, 1991. ISBN 3-540-53904-2.

[5] A. Baumgartner, T. Kutsia, J. Levy, and M. Villaret. A variant of higher-order anti-unification. In F. van Raamsdonk, editor, *RTA*, volume 21 of *LIPIcs*, pages 113–127. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013. ISBN 978-3-939897-53-8.

[6] I. D. Baxter, A. Yahin, L. M. de Moura, M. Sant'Anna, and L. Bier. Clone detection using abstract syntax trees. In *ICSM*, pages 368–377, 1998.

[7] H. Boley. Finite domains and exclusions as first-class citizens. In R. Dyckhoff, editor, *ELP*, volume 798 of *Lecture Notes in Computer Science*, pages 37–61. Springer, 1993. ISBN 3-540-58025-5.

[8] P. E. Bulychev, E. V. Kostylev, and V. A. Zakharov. Anti-unification algorithms and their applications in program analysis. In A. Pnueli, I. Virbitskaite, and A. Voronkov, editors, *Ershov Memorial Conference*, volume 5947 of *Lecture Notes in Computer Science*, pages 413–423. Springer, 2009. ISBN 978-3-642-11485-4.

[9] J. Burghardt. E-generalization using grammars. *Artif. Intell.*, 165(1):1–35, 2005.

[10] A. L. Delcher and S. Kasif. Efficient parallel term matching and anti-unification. *J. Autom. Reasoning*, 9(3):391–406, 1992.

[11] W. S. Evans, C. W. Fraser, and F. Ma. Clone detection via structural abstraction. *Software Quality Journal*, 17(4):309–330, 2009.

[12] R. W. Hasker. *The Replay of Program Derivations*. PhD thesis, University of Illionois at Urbana-Champaign, 1995.

[13] G. Huet. *Résolution d'équations dans des langages d'ordre* $1, 2, \ldots, \omega$. PhD thesis, Université Paris VII, September 1976.

[14] M.-Y. Kao, T. W. Lam, W.-K. Sung, and H.-F. Ting. An even faster and more unifying algorithm for comparing trees via unbalanced bipartite matchings. *J. Algorithms*, 40(2):212–233, 2001.

[15] R. Koschke, R. Falke, and P. Frenzel. Clone detection using abstract syntax suffix trees. In *WCRE*, pages 253–262. IEEE Computer Society, 2006. ISBN 0-7695-2719-1.

[16] U. Krumnack, A. Schwering, H. Gust, and K.-U. Kühnberger. Restricted higher-order anti-unification for analogy making. In M. A. Orgun and J. Thornton, editors, *Australian Conference on Artificial Intelligence*, volume 4830 of *Lecture Notes in Computer Science*, pages 273–282. Springer, 2007. ISBN 978-3-540-76926-2.

[17] S. Kuo and G. R. Cross. An improved algorithm to find the length of the longest common subsequence of two strings. *SIGIR Forum*, 23(3-4):89–99, Apr. 1989. ISSN 0163-5840. doi: 10.1145/74697.74702.

[18] T. Kutsia, J. Levy, and M. Villaret. Anti-unification for unranked terms and hedges. *J. Autom. Reasoning*, 52(2):155–190, 2014.

[19] H. Li and S. J. Thompson. Similar code detection and elimination for Erlang programs. In M. Carro and R. Peña, editors, *PADL*, volume 5937 of *Lecture Notes in Computer Science*, pages 104–118. Springer, 2010. ISBN 978-3-642-11502-8.

[20] J. Lu, J. Mylopoulos, M. Harao, and M. Hagiya. Higher order generalization and its application in program verification. *Ann. Math. Artif. Intell.*, 28 (1-4):107–126, 2000.

[21] M. H. A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of mathematics*, pages 223–243, 1942.

[22] Z. Peng and H. Ting. An $O(n \log n)$-time algorithm for the maximum constrained agreement subtree problem for binary trees. In R. Fleischer and G. Trippen, editors, *Algorithms and Computation*, volume 3341 of *Lecture Notes in Computer Science*, pages 754–765. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-24131-7. doi: 10.1007/978-3-540-30551-4_65.

[23] F. Pfenning. Unification and anti-unification in the calculus of constructions. In *LICS*, pages 74–85. IEEE Computer Society, 1991.

[24] G. D. Plotkin. A note on inductive generalization. *Machine Intel.*, 5(1): 153–163, 1970.

[25] J. C. Reynolds. Transformational systems and the algebraic structure of atomic formulas. *Machine Intel.*, 5(1):135–151, 1970.

[26] C. K. Roy, J. R. Cordy, and R. Koschke. Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of Computer Programming*, 74(7):470 – 495, 2009. ISSN 0167-6423. doi: 10.1016/j.scico.2009.02.007. Special Issue on Program Comprehension (ICPC 2008).

[27] U. Schmid. *Inductive Synthesis of Functional Programs, Universal Planning, Folding of Finite Programs, and Schema Abstraction by Analogical Reasoning*, volume 2654 of *Lecture Notes in Computer Science*. Springer, 2003. ISBN 3-540-40174-1.

[28] A. Yamamoto, K. Ito, A. Ishino, and H. Arimura. Modelling semi-structured documents with hedges for deduction and induction. In C. Rouveirol and M. Sebag, editors, *ILP*, volume 2157 of *Lecture Notes in Computer Science*, pages 240–247. Springer, 2001. ISBN 3-540-42538-1.

[29] K. Zhang. Algorithms for the constrained editing problem between ordered labeled trees and related problems. *Pattern Recognition*, 28:463–474, 1995.