Unranked Anti-Unification with Hedge and Context Variables

Alexander Baumgartner and Temur Kutsia

RISC, 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 instances of $\tilde{g}$ under some substitutions. Context variables are used to abstract vertical differences in the input hedges, and hedge variables are used to abstract horizontal differences. A rule based system in Huet's style will be presented, which computes a set of generalizations of input hedges and records all the differences. The computed generalizations are least general among a certain class of generalizations.

1 Introduction

The anti-unification problem for two terms requires finding their generalization: A term whose substitution instances are the original terms. The interesting generalizations are least general ones (lggs). Anti-unification algorithms are supposed to compute such generalizations.

In 1970, Plotkin [9] and Reynolds [10] 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, and was formulated in the imperative style. Later, Huet [4] proposed another algorithm for the same problem, expressing it as a pair of recursive equations. Since then, a number of algorithms and their modifications have been developed, addressing the problem in various theories and from different application points of view.

In this paper, we consider anti-unification for unranked terms and hedges. The terms are constructed from function symbols that do not have a fixed arity. Hedges are finite sequences of unranked terms. They may contain two kinds of variables: one for hedges, and the other one for contexts. Contexts are hedges with a single occurrence of the distinguished symbol "hole". They are functions which can apply to a context or to a hedge, which are then "plugged" in the place of the hole.

Algorithms for computing lggs for unranked terms and hedges have been described earlier, see, e.g., [7, 11, 2]. The languages there 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.

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 [8]. A restricted use of higher-order variables in generalizations turned out helpful for analogy making with Heuristic-Driven Theory Projection [6]. Anti-unification with combinator terms plays a role in replaying program derivations [3].

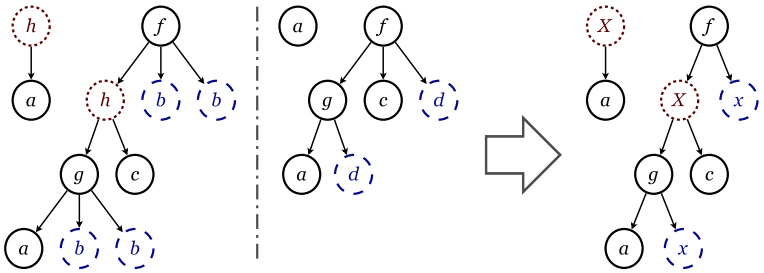
Unranked anti-unification can be used to detect similarities, for instance, between pieces of software code, or between XML documents. These pieces and documents can be abstracted by unranked trees. It can often be the case that, say, the sequence of arguments of a subtree at depth d_1 in one tree is similar to the sequence of arguments of a subtree at depth d_2 in the other tree. It is desirable to detect these similarities. However, The current algorithms for unranked generalization are not designed for that. This is the problem what 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 [1], or an agreement subhedge/subtree [5] 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. We need to restrict variable occurrences in the generalization to guarantee that for each admissible alignment a single lgg is computed. The restriction forbids consecutive hedge variables, chains of context variables, and puts a couple of other constraints (Definition 3.1). It may happen that the skeleton computation function returns a set of admissible alignments, not necessarily a single one. In this case we compute lggs with respect to each admissible alignment, and then minimize the obtained set, to get lggs (not necessarily a single one) with respect to the entire set of admissible alignments.

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

Example 1.1. *The hedge* $(X(a), f(X(g(a,x),c),x))$ *is a generalization of the two hedges* $(h(a), f(h(g(a,b,b),c),b,b))$ *and* $(a, f(g(a,d),c,d))$.



2 Preliminaries

Let F be a countable set of unranked function symbols, V_H be a countable set of hedge variables (with arity zero) and V_C be a countable set of unranked context variables. Let $\circ$ be a special symbol called the hole. F, V_H and V_C are pairwise distinct. *Terms* t, *hedges* $\tilde{s}$ and *contexts* $\tilde{c}$ are defined by the following grammar:

$$t := \phi(\tilde{s}) \mid x \qquad \tilde{s} := (t_1, \ldots, t_n) \qquad \tilde{c} := (\tilde{s}_1, \circ, \tilde{s}_2) \mid (\tilde{s}_1, \phi(\tilde{c}), \tilde{s}_2)$$

where $\phi \in F \cup V_C$, $x \in V_H$, and $n \geqslant 0$. The length of a hedge $\tilde{s}$, denoted $|\tilde{s}|$, is the number of elements in it. 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 $\tilde{s}|_i^j = \varepsilon$.

A *substitution* σ 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 and the hole in it will be replaced by that argument hedge. σ 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 $\tilde{s}$.

A hedge $\tilde{s}$ is the *instance* of another 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 fact by $\tilde{q} \preceq \tilde{s}$ ($\simeq$ and $\prec$ are defined as usual). 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 $\omega(\tilde{s})$ of a hedge $\tilde{s}$ is defined by the concatenation of the depth-first pre-order traversals of the constituent terms. For instance, the word representation of $(a, f(g(a, g(b, b)), c))$ would be $afgagbbc$. Note that generalizations contain certain common subsequences of the word representation of the input hedges. Observe, e.g., the hedges from example 1.1:

$$\begin{array}{rl} \tilde{s} = & (\ h(\mathbf{a}), \mathbf{f}(\ h(\mathbf{g}(\mathbf{a}, b, b), \mathbf{c}), b, b)) \\ \tilde{q} = & (\quad \mathbf{a}\ , \mathbf{f}(\quad \mathbf{g}(\mathbf{a}, d\ \ ), \mathbf{c}\ , d\ \ )) \\ \hline \tilde{g} = & (X(\mathbf{a}), \mathbf{f}(X(\mathbf{g}(\mathbf{a}, x\ \ ), \mathbf{c}), x\ \ )) \end{array}$$

We will use this property in the formulation of our algorithm. 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.

A set of *position indexes* (or just positions) of a term is a prefix-closed set of integer sequences. We use $\cdot$ for sequence concatenation. For a hedge $\tilde{s}$, the position index $i \cdot I$ denotes the position I in $\tilde{s}|_i$. With $<$ we denote the *lexicographic ordering* and with $\sqsubset$ the *strict prefix relation* on positions. Given three position indexes I_1, I_2 and I_3, the ternary relation $\bowtie$ is defined as

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

Definition 2.1 (Alignment)**.** *Given two hedges $\tilde{s}$ and $\tilde{q}$, an alignment is a sequence of the form $a_1\langle I_1, J_1\rangle \ldots a_m\langle I_m, J_m\rangle$ such that:*

- *$I_1 < \cdots < I_m$ and $J_1 < \cdots < J_m$, and*
- *a_k is the symbol at position I_k in $\tilde{s}$ and at position J_k in $\tilde{q}$ for all $1 \leqslant k \leqslant m$.*

We write $|\mathfrak{a}|$ for the length of an alignment $\mathfrak{a}$.

Definition 2.2 (Admissible alignment)**.** *An alignment $\mathfrak{a}$ of two hedges $\tilde{s}$ and $\tilde{q}$ is admissible if there are no collisions of the form defined below:*

- *A collision appears at two elements $a_e\langle I_e, J_e\rangle$, $a_f\langle I_f, J_f\rangle$ of $\mathfrak{a}$ if either ($I_e \sqsubset I_f$ and $J_e \not\sqsubset J_f$) or ($I_e \not\sqsubset I_f$ and $J_e \sqsubset J_f$).*
- *A collision appears at three elements $a_e\langle I_e, J_e\rangle$, $a_f\langle I_f, J_f\rangle$, $a_g\langle I_g, J_g\rangle$ of $\mathfrak{a}$ if $I_e \bowtie_{I_g} I_f$ and $J_f \bowtie_{J_e} J_g$.*

Note that for any two elements $a_e\langle I_e, J_e\rangle$, $a_f\langle I_f, J_f\rangle$ of an admissible alignment $\mathfrak{a}$, $I_e < I_f$ iff $J_e < J_f$ and $I_e \sqsubset I_f$ iff $J_e \sqsubset J_f$. A *longest admissible alignment* (laa) of two hedges is their admissible alignment with a longest length.

Theorem 2.1. *Let $\mathfrak{a}$ be an alignment of the hedges $\tilde{s}$ and $\tilde{q}$. A generalization $\tilde{g}$ of $\tilde{s}$ and $\tilde{q}$ with a bijective name-preserving mapping from all the symbols in $\mathfrak{a}$ to the set of all function symbols of $\tilde{g}$, respecting $<$ and $\sqsubset$ between the positions in $\mathfrak{a}$, exists iff $\mathfrak{a}$ is admissible. (We call $\tilde{g}$ a supporting generalization of $\mathfrak{a}$.)*

Example 2.1. *Let $\tilde{s} = (a, a(b, b))$ and $\tilde{q} = (a(a(b(b))), b, b)$.*

- *$b\langle 2\cdot 2, 1\cdot 1\cdot 1\rangle\, a\langle 1, 1\rangle$ is not an alignment of $\tilde{s}, \tilde{q}$.*
- *$a\langle 1, 1\rangle\, a\langle 2, 1\cdot 1\rangle\, b\langle 2\cdot 1, 1\cdot 1\cdot 1\rangle\, b\langle 2\cdot 2, 3\rangle$ is a non-admissible alignment of $\tilde{s}, \tilde{q}$.*
- *$a\langle 1, 1\rangle\, b\langle 2\cdot 2, 3\rangle$ is an admissible alignment of $\tilde{s}, \tilde{q}$ and $(a(x), y, X(b))$ is a supporting generalization for this alignment.*
- *$a\langle 2, 1\rangle\, b\langle 2\cdot 2, 1\cdot 1\cdot 1\cdot 1\rangle$ would be another admissible alignment of $\tilde{s}, \tilde{q}$, with $(x, a(y, Y(b)), z)$ being a supporting generalization of it.*
- *$a\langle 1, 1\cdot 1\rangle\, b\langle 2\cdot 1, 2\rangle\, b\langle 2\cdot 2, 3\rangle$ is an laa of $\tilde{s}, \tilde{q}$ and $(X(a(x)), Y(b, b))$ is a supporting generalization for this alignment.*

In our prototype implementation we compute only one kind of generalizations: Those which support *longest* admissible alignments. The laa's are computed with the aid of a function that for two hedges returns a set of common alignments with maximum length, not exceeding a given upper bound $k \in \mathbb{N}$. Setting k initially large enough (e.g. k is the number of all symbols in one of the given hedges) this function returns all longest common alignments. If this set contains no admissible alignments, then we reduce k and repeat the computation. Otherwise we return the subset of admissible alignments. The algorithm can be described in four steps:

1. $k := |\omega(\tilde{s})|$.
2. $A := \mathfrak{R}(k, \tilde{s}, \tilde{q})$.
3. If $A \neq \varnothing$ and $\text{collfree}(A) = \varnothing$ and $k > 0$ then $k := \max(\{|\mathfrak{a}| : \mathfrak{a} \in A\}) - 1$ and go to 2.
4. return $\text{collfree}(A)$.

To test whether an alignment is admissible or not, we use definition 2.2 and its two cases of collisions. With $\text{collfree}(A)$ we denote the subset of admissible alignments (those without collisions). Other classes of admissible alignments may be obtained in a similar way, if a function can be formulated which computes some alignments and takes an upper bound k, that can be reduced if none of the computed alignments is admissible.

Example 2.2 (Computing laa of two hedges)**.** *We illustrate the computation of laas with the two hedges $\tilde{s} = (f(a, f(b, b)), c)$ and $\tilde{q} = (f(a, b), b, g(c))$.*

1. $k := |fafbbc| = 6$.
2. $A := \mathfrak{R}(6, (f(a, f(b, b)), c), (f(a, b), b, g(c))) =$
   $\{f\langle 1, 1\rangle a\langle 1\cdot 1, 1\cdot 1\rangle b\langle 1\cdot 2\cdot 1, 1\cdot 2\rangle b\langle 1\cdot 2\cdot 2, 2\rangle c\langle 2, 3\cdot 1\rangle\}$.

3. collfree$(A) = \varnothing \Longrightarrow k := 5 - 1$ *and go to* 2.

2. $A := \mathfrak{R}(4, (f(a, f(b, b)), c), (f(a, b), b, g(c))) =$
$\{f\langle 1,1\rangle a\langle 1\cdot 1, 1\cdot 1\rangle b\langle 1\cdot 2\cdot 1, 1\cdot 2\rangle b\langle 1\cdot 2\cdot 2, 2\rangle, f\langle 1,1\rangle a\langle 1\cdot 1, 1\cdot 1\rangle b\langle 1\cdot 2\cdot 1, 1\cdot 2\rangle c\langle 2, 3\cdot 1\rangle, \dots\}$.

3. collfree$(A) \neq \varnothing$.

4. *return* collfree$(A) =$
$\{f\langle 1,1\rangle a\langle 1\cdot 1, 1\cdot 1\rangle b\langle 1\cdot 2\cdot 1, 1\cdot 2\rangle c\langle 2, 3\cdot 1\rangle, f\langle 1,1\rangle a\langle 1\cdot 1, 1\cdot 1\rangle b\langle 1\cdot 2\cdot 2, 1\cdot 2\rangle c\langle 2, 3\cdot 1\rangle\}$.

The result of this computation is a set which consists of the two longest admissible alignments of $\tilde{s}$ and $\tilde{q}$. In the next section an algorithm will be presented to compute supporting generalizations for given admissible alignments.

3 Computing Least General Rigid Generalizations

Two symbols μ, ν of a hedge are *horizontal consecutive* if their position indexes $I_\mu \cdot i_\mu$ and $I_\nu \cdot i_\nu$ are in the relation $I_\mu = I_\nu$ and $i_\mu + 1 = i_\nu$. They are in a *vertical chain* if their position indexes I_μ and I_ν are in the relation $I_\nu = I_\mu \cdot 1$ and there is no term at position $I_\mu \cdot 2$.

Definition 3.1 (Rigid generalization)**.** *Given two hedges $\tilde{s}, \tilde{q}$ and an admissible alignment $\mathfrak{a}$, we say that a hedge $\tilde{g}$ is a rigid generalization of $\tilde{s}$ and $\tilde{q}$ with respect to $\mathfrak{a}$, if either $\mathfrak{a}$ is an empty sequence and $\tilde{g}$ is a hedge variable, or $\tilde{g}$ is a supporting generalization of $\mathfrak{a}$ such that:*

- *No context variable in $\tilde{g}$ applies to the empty hedge.*
- *There are substitutions σ_1, σ_2 with $\tilde{g}\sigma_1 = \tilde{s}, \tilde{g}\sigma_2 = \tilde{q}$ where all the contexts in σ_1, σ_2 are terms (not arbitrary hedges).*
- *$\tilde{g}$ 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.*

This definition puts some restrictions on the usage of the variables. For instance $X(a, b)$ is a rigid generalization of $(f(g(a, b, c)))$ and (a, b) with respect to $a\langle 1\cdot 1\cdot 1, 1\rangle b\langle 1\cdot 1\cdot 2, 2\rangle$. $X(a, b, x)$ and $X(Y(a, b))$ are not rigid generalizations.

Definition 3.2 (Rigid lgg with respect to A)**.** *A generalization $\tilde{g}$ of two hedges $\tilde{s}, \tilde{q}$ is called a rigid lgg with respect to a set A of alignments, if $\tilde{g}$ is a rigid generalization of $\tilde{s}, \tilde{q}$ with respect to some $\mathfrak{a} \in A$ and there is no rigid generalization $\tilde{h}$ of $\tilde{s}, \tilde{q}$ with respect to some $\mathfrak{b} \in A$ with $\tilde{g} \prec \tilde{h}$.*

The rule based system $\mathfrak{G}$ works on a quadruple P; Q; S; σ, where P is a set of anti-unification problems (AUPs), Q is a set called the vertical store, S is a set called the horizontal store and σ is a substitution. An AUP has the form $x : \tilde{s} \triangleq \tilde{q}$; $X : \tilde{c} \triangleq \tilde{d}$; $\mathfrak{a}$, where x is a hedge variable, X a context variable, $\tilde{c}, \tilde{d}$ are contexts and $\mathfrak{a}$ is an admissible alignment of the hedges $\tilde{s}, \tilde{q}$.

In the following rules, we use the symbols Y, Z for fresh context variables and y, z for fresh hedge variables. We use the brackets [] for context application and the symbol $\cup$ stands for disjoint union.

Dec-H: **Decompose Hedge**
$\{x : \tilde{s} \triangleq \tilde{q};\ X : \tilde{c} \triangleq \tilde{d};\ a_1\langle i_1 \cdot I_1, j_1 \cdot J_1\rangle \dots a_k\langle i_k \cdot I_k, j_k \cdot J_k\rangle$
$a_{k+1}\langle i_{k+1} \cdot I_{k+1}, j_{k+1} \cdot J_{k+1}\rangle \dots a_m\langle i_m \cdot I_m, j_m \cdot J_m\rangle\} \cup P;\ Q;\ S;\ \sigma \implies$
$\{y : \tilde{s}|_{i_1}^{i_k} \triangleq \tilde{q}|_{j_1}^{j_k};\ Y : \circ \triangleq \circ;\ a_1\langle (i_1 - i_1^{--}) \cdot I_1, (j_1 - j_1^{--}) \cdot J_1\rangle \dots$
$a_k\langle (i_k - i_1^{--}) \cdot I_k, (j_k - j_1^{--}) \cdot J_k\rangle\} \cup$
$\{z : \tilde{s}|_{i_k^{++}}^{i_m} \triangleq \tilde{q}|_{j_k^{++}}^{j_m};\ Z : \circ \triangleq \circ;\ a_{k+1}\langle (i_{k+1} - i_k) \cdot I_{k+1}, (j_{k+1} - j_k) \cdot J_{k+1}\rangle \dots$
$a_m\langle (i_m - i_k) \cdot I_m, (j_m - j_k) \cdot J_m\rangle\} \cup P;$
$\{X : \tilde{c}[\tilde{s}|_1^{i_1^{--}}, \circ, \tilde{s}|_{i_m^{++}}^{|\tilde{q}|}] \triangleq \tilde{d}[\tilde{q}|_1^{j_1^{--}}, \circ, \tilde{q}|_{j_m^{++}}^{|\tilde{q}|}]\} \cup Q;\ S;\ \sigma\{x \mapsto (Y(y), Z(z))\},$
if $m > 1$ and not$(i_1 = \dots = i_m)$ and not$(j_1 = \dots = j_m)$. Where $i_1 \neq i_{k+1}$ and $j_1 \neq j_{k+1}$ and $(i_1 = i_k$ or $j_1 = j_k)$. We write i^{--} for $i - 1$ and i^{++} for $i + 1$.

Abs-C: **Abstract left Context**
$\{x : (\tilde{s}_l, \phi(\tilde{s}), \tilde{s}_r) \triangleq \tilde{q};\ X : \tilde{c} \triangleq \tilde{d};\ a_1\langle i \cdot I_1, J_1\rangle \dots a_m\langle i \cdot I_m, J_m\rangle\} \cup P;\ Q;\ S;\ \sigma \Longrightarrow$
$\{x : \tilde{s} \triangleq \tilde{q};\ X : \tilde{c}[\tilde{s}_l,\ \phi(\circ),\ \tilde{s}_r] \triangleq \tilde{d};\ a_1\langle I_1, J_1\rangle \dots a_m\langle I_m, J_m\rangle\} \cup P;\ Q;\ S;\ \sigma,$
if $I_1 \neq \epsilon$. Where $\phi(\tilde{s})$ is the term at the positions i and $\tilde{s}_l, \tilde{s}_r$ are hedges.

Abs-C: **Abstract right Context**
$\{x : \tilde{s} \triangleq (\tilde{q}_l, \phi(\tilde{q}), \tilde{q}_r);\ X : \tilde{c} \triangleq \tilde{d};\ a_1\langle I_1, j \cdot J_1\rangle \dots a_m\langle I_m, j \cdot J_m\rangle\} \cup P;\ Q;\ S;\ \sigma \Longrightarrow$
$\{x : \tilde{s} \triangleq \tilde{q};\ X : \tilde{c} \triangleq \tilde{d}[\tilde{q}_l,\ \phi(\circ),\ \tilde{q}_r];\ a_1\langle I_1, J_1\rangle \dots a_m\langle I_m, J_m\rangle\} \cup P;\ Q;\ S;\ \sigma,$
if $J_1 \neq \epsilon$. Where $\phi(\tilde{q})$ is the term at the positions j and $\tilde{q}_l, \tilde{q}_r$ are hedges.

App-A: **Apply Alignment**
$\{x : (\tilde{s}_l, \phi(\tilde{s}), \tilde{s}_r) \triangleq (\tilde{q}_l, \psi(\tilde{q}), \tilde{q}_r);\ X : \tilde{c} \triangleq \tilde{d};$
$a_1\langle i, j\rangle a_2\langle i \cdot I_2, j \cdot J_2\rangle \dots a_m\langle i \cdot I_m, j \cdot J_m\rangle\} \cup P;\ Q;\ S;\ \sigma \implies$
$\{y : \tilde{s} \triangleq \tilde{q};\ Y : \circ \triangleq \circ;\ a_2\langle I_2, J_2\rangle \dots a_m\langle I_m, J_m\rangle\} \cup P;$
$\{X : \tilde{c}[\tilde{s}_l, \circ, \tilde{s}_r] \triangleq \tilde{d}[\tilde{q}_l, \circ, \tilde{q}_r]\} \cup Q;\ S;\ \sigma\{x \mapsto a_1(Y(y))\},$
where $\phi(\tilde{s}), \psi(\tilde{q})$ 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**
$\{x : \tilde{s} \triangleq \tilde{q};\ X : \tilde{c} \triangleq \tilde{d};\ \epsilon\} \cup P;\ Q;\ S;\ \sigma \implies P;\ Q;\ \{x : \tilde{s} \triangleq \tilde{q}\} \cup S;\ \sigma\{X \mapsto \circ\}.$

Res-C: **Restore Context Variable**
$P;\ \{X : (\tilde{c}_l, c, \tilde{c}_r) \triangleq (\tilde{d}_l, d, \tilde{d}_r)\} \cup Q;\ S;\ \sigma \implies$
$P;\ \{X : c \triangleq d\} \cup Q;\ \{y : \tilde{c}_l \triangleq \tilde{d}_l\} \cup \{z : \tilde{c}_r \triangleq \tilde{d}_r\} \cup S;\ \sigma\{X \mapsto (y, X(\circ), z)\},$
if not $\varepsilon = \tilde{c}_l = \tilde{c}_r = \tilde{d}_l = \tilde{d}_r$. Where c, d are those terms where the hole appears.

Mer-H: **Merge Hedge Variable**
$P;\ Q;\ \{x_1 : \tilde{s} \triangleq \tilde{q}, x_2 : \tilde{s} \triangleq \tilde{q}\} \cup S;\ \sigma \Longrightarrow P;\ Q;\ \{x_1 : \tilde{s} \triangleq \tilde{q}\} \cup S;\ \sigma\{x_2 \mapsto x_1\}.$

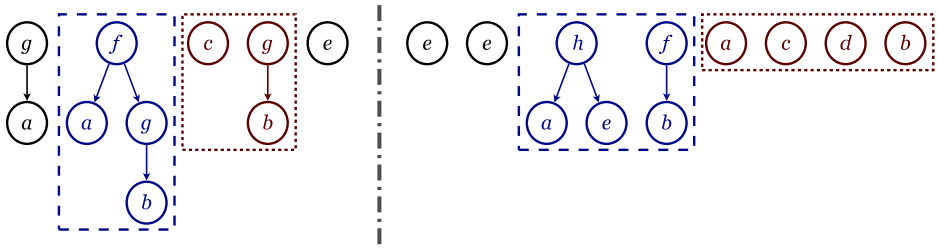
Mer-C: **Merge Context Variable**
$P;\ \{X_1 : \tilde{c} \triangleq \tilde{d}, X_2 : \tilde{c} \triangleq \tilde{d}\} \cup Q;\ S;\ \sigma \Longrightarrow P;\ \{X_1 : \tilde{c} \triangleq \tilde{d}\} \cup Q;\ S;\ \sigma\{X_2 \mapsto X_1(\circ)\}.$

Clr-H: **Clear Hedge Variable**
$P;\ Q;\ \{x : \varepsilon \triangleq \varepsilon\} \cup S;\ \sigma \implies P;\ Q;\ S;\ \sigma\{x \mapsto \varepsilon\}.$

Clr-C: **Clear Context Variable**
$P;\ \{X : \circ \triangleq \circ\} \cup Q;\ S;\ \sigma \implies P;\ Q;\ S;\ \sigma\{X \mapsto \circ\}.$

The system is initialized with $\{x : \tilde{s} \triangleq \tilde{q};\ X : \circ \triangleq \circ;\ \mathfrak{a}\};\ \varnothing;\ \varnothing;\ \{x \mapsto X(x)\}$, where $\mathfrak{a}$ is an admissible alignment of the hedges $\tilde{s}, \tilde{q}$ for which we want to compute the generalization. X is a fresh context variable and x is a fresh hedge variable. The rules are applied exhaustively. When no more rule is applicable, then the final state has been reached and $x\sigma$ is the computed generalization. The stores Q and S will contain the vertical and horizontal differences respectively.

The decomposition rule splits an AUP into two parts. We will explain it on an example and therefore 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 these hedges.



The very left and right subhedges, which do not have any corresponding alignment elements will be moved into the store. In the example these are the subhedges $g(a)$ and e in the first hedge, and (e, e) in the second hedge. The splitting will be performed in a way such that one of the new AUPs (the left part of the split hedges) is minimal, in the sense that one of the two sides will be a term. (Note that there are no collisions at three elements of an alignment.) In our example, the clips which are marked by the blue dashed rectangles will become this new minimal AUP. The remaining rest will form another AUP (the red dotted rectangles in the example).

The abstract context rules record and reduce vertical differences. We consider the blue dashed clippings $f(a, g(b)) \triangleq (h(a, e), f(b))$ of the figure above. The alignment is $a\langle 1\cdot 1, 1\cdot 1\rangle$ $b\langle 1\cdot 2\cdot 1, 2\cdot 1\rangle$. The abstract left context rule will detect and record the vertical displacement f at the left hedge and reduce the problem to $(a, g(b)) \triangleq (h(a, e), f(b))$, with $a\langle 1, 1\cdot 1\rangle b\langle 2\cdot 1, 2\cdot 1\rangle$ as new alignment.

The rule apply alignment finally uses the head element of an alignment and adds this element to the substitution σ. It is only applicable if all the vertical differences are resolved and no decomposition is possible, which means that all the other elements of the alignment are "descendants of the head element". The AUP will be reduced such that the head element of the alignment and the corresponding elements in the hedges will be removed. The subhedges at the left and right side of the corresponding elements (Note that they cannot have any corresponding alignment elements) will be moved into the store, like in the decomposition rule.

Theorem 3.1 (Correctness of $\mathfrak{G}$)**.** *Given two hedges $\tilde{s}$ and $\tilde{q}$ and their admissible alignment $\mathfrak{a}$, the rule based system $\mathfrak{G}$ terminates and computes a rigid lgg $\tilde{g}$ of $\tilde{s}$ and $\tilde{q}$ with respect to $\{\mathfrak{a}\}$. Moreover, $\tilde{g}$ is unique modulo $\simeq$ w.r.t. $\{\mathfrak{a}\}$.*

Example 3.1. *We illustrate how to compute the rigid lgg which corresponds to the laa $f\langle 1, 1\rangle$ $a\langle 1\cdot 1, 1\cdot 1\rangle b\langle 1\cdot 2\cdot 1, 1\cdot 2\rangle$ of the hedges $(f(a, f(b, b)))$ and $(f(a, b), b)$.*

$$\{x\colon (f(a, f(b, b))) \triangleq (f(a, b), b);\ X\colon \circ \triangleq \circ;\ f\langle 1, 1\rangle a\langle 1\cdot 1, 1\cdot 1\rangle b\langle 1\cdot 2\cdot 1, 1\cdot 2\rangle\};\ \varnothing;\ \varnothing;\ \{x \mapsto X(x)\}$$
$$\Rightarrow_{\mathsf{App\text{-}A}} \{y_1\colon (a, f(b, b)) \triangleq (a, b);\ Y_1\colon \circ \triangleq \circ;\ a\langle 1, 1\rangle b\langle 2\cdot 1, 2\rangle\};\ \{X\colon \circ \triangleq \circ, b\};\ \varnothing;\ \{x \mapsto X(f(Y_1(y_1)))\}$$
$$\Rightarrow^{\mathsf{Res\text{-}C}}_{\mathsf{Clear}} \{y_1\colon (a, f(b, b)) \triangleq (a, b);\ Y_1\colon \circ \triangleq \circ;\ a\langle 1, 1\rangle b\langle 2\cdot 1, 2\rangle\};\ \varnothing;\ \{z\colon \varepsilon \triangleq b\};\ \{x \mapsto f(Y_1(y_1)), z\}$$
$$\Rightarrow^{\mathsf{Dec\text{-}H}}_{\mathsf{Clear}} \{y_2\colon (a) \triangleq (a);\ Y_2\colon \circ \triangleq \circ;\ a\langle 1, 1\rangle,\ y_3\colon (f(b, b)) \triangleq (b);\ Y_3\colon \circ \triangleq \circ;\ b\langle 1\cdot 1, 1\rangle\};$$
$$\varnothing;\ \{z\colon \varepsilon \triangleq b\};\ \{x \mapsto f(Y_2(y_2), Y_3(y_3)), z\}$$
$$\Rightarrow^{\mathsf{App\text{-}A}}_{\mathsf{Clear}} \{y_4\colon \varepsilon \triangleq \varepsilon;\ Y_4\colon \circ \triangleq \circ;\ \epsilon,\ y_3\colon (f(b, b)) \triangleq (b);\ Y_3\colon \circ \triangleq \circ;\ b\langle 1\cdot 1, 1\rangle\};$$
$$\varnothing;\ \{z\colon \varepsilon \triangleq b\};\ \{x \mapsto f(a(Y_4(y_4)), Y_3(y_3)), z\}$$

$\Rightarrow^{\text{Sol-H}}_{\text{Clear}}$ $\{y_3\colon (f(b,b)) \triangleq (b);\ Y_3\colon \circ \triangleq \circ;\ b\langle 1\cdot 1, 1\rangle\};\ \varnothing;\ \{z\colon \varepsilon \triangleq b\};\ \{x \mapsto f(a, Y_3(y_3)), z\}$
$\Rightarrow_{\text{Abs-C}}$ $\{y_3\colon (b,b) \triangleq (b);\ Y_3\colon f(\circ) \triangleq \circ;\ b\langle 1, 1\rangle\};\ \varnothing;\ \{z\colon \varepsilon \triangleq b\};\ \{x \mapsto f(a, Y_3(y_3)), z\}$
$\Rightarrow_{\text{App-A}}$ $\{y_5\colon \varepsilon \triangleq \varepsilon;\ Y_5\colon \circ \triangleq \circ;\ \epsilon\};\ \{Y_3\colon f(\circ, b) \triangleq \circ\};\ \{z\colon \varepsilon \triangleq b\};\ \{x \mapsto f(a, Y_3(b(Y_5(y_5)))), z\}$
$\Rightarrow^{\text{Sol-H}}_{\text{Clear}}$ $\varnothing;\ \{Y_3\colon f(\circ, b) \triangleq \circ\};\ \{z\colon \varepsilon \triangleq b\};\ \{x \mapsto f(a, Y_3(b)), z\}$.

Example 3.2. *We illustrate the computation of a rigid lgg which corresponds to the 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. (Note that the hedges to be generalized are assumed to be variable disjoint.)*

$\{x\colon (U(a), f(U(g(a,b,b),c),b,b)) \triangleq (a, f(g(a,d),c,d));\ X\colon \circ \triangleq \circ;$
$a\langle 1\cdot 1, 1\rangle f\langle 2, 2\rangle g\langle 2\cdot 1\cdot 1, 2\cdot 1\rangle a\langle 2\cdot 1\cdot 1\cdot 1, 2\cdot 1\cdot 1\rangle c\langle 2\cdot 1\cdot 2, 2\cdot 2\rangle\};\ \varnothing;\ \varnothing;\ \{x \mapsto X(x)\}$
$\Rightarrow^{\text{Dec-H}}_{\text{Clear}}$ $\{y_1\colon U(a) \triangleq a;\ Y_1\colon \circ \triangleq \circ;\ a\langle 1\cdot 1, 1\rangle;\ z_1\colon f(U(g(a,b,b),c),b,b) \triangleq f(g(a,d),c,d);\ Z_1\colon \circ \triangleq \circ;$
$f\langle 1, 1\rangle g\langle 1\cdot 1\cdot 1, 1\cdot 1\rangle a\langle 1\cdot 1\cdot 1\cdot 1, 1\cdot 1\cdot 1\rangle c\langle 1\cdot 1\cdot 2, 1\cdot 2\rangle\};\ \varnothing;\ \varnothing;\ \{x \mapsto (Y_1(y_1), Z_1(z_1))\}$
$\Rightarrow_{\text{Abs-C}}$ $\{y_1\colon a \triangleq a;\ Y_1\colon U(\circ) \triangleq \circ;\ a\langle 1, 1\rangle;\ z_1\colon f(U(g(a,b,b),c),b,b) \triangleq f(g(a,d),c,d);\ Z_1\colon \circ \triangleq \circ;$
$f\langle 1, 1\rangle g\langle 1\cdot 1\cdot 1, 1\cdot 1\rangle a\langle 1\cdot 1\cdot 1\cdot 1, 1\cdot 1\cdot 1\rangle c\langle 1\cdot 1\cdot 2, 1\cdot 2\rangle\};\ \varnothing;\ \varnothing;\ \{x \mapsto (Y_1(y_1), Z_1(z_1))\}$
$\Rightarrow^{\text{App-A}}_{\text{Sol-H}}$ $\{z_1\colon f(U(g(a,b,b),c),b,b) \triangleq f(g(a,d),c,d);\ Z_1\colon \circ \triangleq \circ;$
$f\langle 1, 1\rangle g\langle 1\cdot 1\cdot 1, 1\cdot 1\rangle a\langle 1\cdot 1\cdot 1\cdot 1, 1\cdot 1\cdot 1\rangle c\langle 1\cdot 1\cdot 2, 1\cdot 2\rangle\};\ \{Y_1\colon U(\circ) \triangleq \circ\};\ \varnothing;\ \{x \mapsto (Y_1(a), Z_1(z_1))\}$
$\Rightarrow^{\text{App-A}}_{\text{Clear}}$ $\{z_2\colon (U(g(a,b,b),c),b,b) \triangleq (g(a,d),c,d);\ Z_2\colon \circ \triangleq \circ;\ g\langle 1\cdot 1, 1\rangle a\langle 1\cdot 1\cdot 1, 1\cdot 1\rangle c\langle 1\cdot 2, 2\rangle\};$
$\{Y_1\colon U(\circ) \triangleq \circ\};\ \varnothing;\ \{x \mapsto (Y_1(a), f(Z_2(z_2)))\}$
$\Rightarrow_{\text{Abs-C}}$ $\{z_2\colon (g(a,b,b),c) \triangleq (g(a,d),c,d);\ Z_2\colon (U(\circ),b,b) \triangleq \circ;\ g\langle 1, 1\rangle a\langle 1\cdot 1, 1\cdot 1\rangle c\langle 2, 2\rangle\};$
$\{Y_1\colon U(\circ) \triangleq \circ\};\ \varnothing;\ \{x \mapsto (Y_1(a), f(Z_2(z_2)))\}$
$\Rightarrow^{\text{Dec-H}}_{\text{Clear}}$ $\{z_3\colon g(a,b,b) \triangleq g(a,d);\ Z_3\colon \circ \triangleq \circ;\ g\langle 1, 1\rangle a\langle 1\cdot 1, 1\cdot 1\rangle;\ z_4\colon c \triangleq c;\ Z_4\colon \circ \triangleq \circ;\ c\langle 1, 1\rangle\};$
$\{Y_1\colon U(\circ) \triangleq \circ;\ Z_2\colon (U(\circ),b,b) \triangleq (\circ,d)\};\ \varnothing;\ \{x \mapsto (Y_1(a), f(Z_2(Z_3(z_3), Z_4(z_4))))\}$
$\Rightarrow^{\text{App-A}}_{\text{Clear}}$ $\{z_5\colon (a,b,b) \triangleq (a,d);\ Z_5\colon \circ \triangleq \circ;\ a\langle 1, 1\rangle;\ z_4\colon c \triangleq c;\ Z_4\colon \circ \triangleq \circ;\ c\langle 1, 1\rangle\};$
$\{Y_1\colon U(\circ) \triangleq \circ;\ Z_2\colon (U(\circ),b,b) \triangleq (\circ,d)\};\ \varnothing;\ \{x \mapsto (Y_1(a), f(Z_2(g(Z_5(z_5)), Z_4(z_4))))\}$
$\Rightarrow^{\text{Res-C}}_{\text{Clear}}$ $\{z_5\colon (a,b,b) \triangleq (a,d);\ Z_5\colon \circ \triangleq \circ;\ a\langle 1, 1\rangle;\ z_4\colon c \triangleq c;\ Z_4\colon \circ \triangleq \circ;\ c\langle 1, 1\rangle\};$
$\{Y_1\colon U(\circ) \triangleq \circ;\ Z_2\colon U(\circ) \triangleq \circ\};\ \{y_2\colon (b,b) \triangleq d\};\ \{x \mapsto (Y_1(a), f(Z_2(g(Z_5(z_5)), Z_4(z_4)), y_2))\}$
$\Rightarrow_{\text{Mer-C}}$ $\{z_5\colon (a,b,b) \triangleq (a,d);\ Z_5\colon \circ \triangleq \circ;\ a\langle 1, 1\rangle;\ z_4\colon c \triangleq c;\ Z_4\colon \circ \triangleq \circ;\ c\langle 1, 1\rangle\};$
$\{Y_1\colon U(\circ) \triangleq \circ\};\ \{y_2\colon (b,b) \triangleq d\};\ \{x \mapsto (Y_1(a), f(Y_1(g(Z_5(z_5)), Z_4(z_4)), y_2))\}$
$\Rightarrow^{\text{App-A}}_{\text{Sol-H}}$ $\{z_5\colon (a,b,b) \triangleq (a,d);\ Z_5\colon \circ \triangleq \circ;\ a\langle 1, 1\rangle\};$
$\{Y_1\colon U(\circ) \triangleq \circ\};\ \{y_2\colon (b,b) \triangleq d\};\ \{x \mapsto (Y_1(a), f(Y_1(g(Z_5(z_5)), c), y_2))\}$
$\Rightarrow^{\text{App-A}}_{\text{Sol-H}}$ $\varnothing;\ \{Y_1\colon U(\circ) \triangleq \circ;\ Z_5\colon (\circ,b,b) \triangleq (\circ,d)\};\ \{y_2\colon (b,b) \triangleq d\};\ \{x \mapsto (Y_1(a), f(Y_1(g(Z_5(a)), c), y_2))\}$
$\Rightarrow^{\text{Res-C}}_{\text{Clear}}$ $\varnothing;\ \{Y_1\colon U(\circ) \triangleq \circ\};\ \{y_2\colon (b,b) \triangleq d;\ y_3\colon (b,b) \triangleq d\};\ \{x \mapsto (Y_1(a), f(Y_1(g(a, y_3), c), y_2))\}$
$\Rightarrow_{\text{Mer-H}}$ $\varnothing;\ \{Y_1\colon U(\circ) \triangleq \circ\};\ \{y_2\colon (b,b) \triangleq d\};\ \{x \mapsto (Y_1(a), f(Y_1(g(a, y_2), c), y_2))\}$

4 Final Remarks

The rule based system $\mathfrak{G}$ computes a rigid lgg which corresponds to one given admissible alignment, but the alignment computation may return a finite set of admissible alignments A. It may be that one admissible alignment in A is a subsequence of another one. Even if this

is not the case, it may happen that one of the computed rigid lggs (with respect to a certain alignment) is more general than another one (with respect to another alignment).

Therefore we need a minimization step which filters out the rigid lggs with respect to A and removes equivalent results. To perform this step we need to solve a hedge-matching problem, which allows context and hedge variables in both sides of matching equations. To the best of our knowledge, there is no algorithm to solve this kind of matching problems yet.

Acknowledgments

This research has been supported by the Austrian Science Fund (FWF) under the project SToUT (P 24087-N18).

References

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

[2] 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.

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

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

[5] 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.

[6] 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.

[7] T. Kutsia, J. Levy, and M. Villaret. Anti-unification for unranked terms and hedges. *J. Autom. Reasoning*, 2013. To appear. A preliminary version in Proc. RTA 2011.

[8] 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.

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

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

[11] 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.