

Quantitative generalization of variadic structures with binders

Alexander Baumgartner¹ and Temur Kutsia²

¹ Instituto de Ciencias de la Ingeniería, Universidad de O'Higgins, Rancagua, Chile
alexander.baumgartner.x@gmail.com

² Research Institute for Symbolic Computation, Johannes Kepler University, Linz, Austria
kutsia@risc.jku.at

Abstract

This work addresses the quantitative generalization problem in the variadic nominal language. The language combines variadic (i.e., flexible arity) functions with binding structures and freshness constraints, making it more universal than the standard (i.e., fixed arity) nominal language. A possible usage example would be to represent abstract syntax trees (ASTs) that serve as a structural representation of some program code, where the key advantage of the nominal language is that variable names can be bounded (and therefore renamed) within a certain scope. The quantitative generalization problem is concerned with discovering structural similarities of two given input expressions (e.g., two ASTs), where similarity of function symbols is specified by a given fuzzy relation. To name a real-life example: One can define that when comparing two pieces of Python code, the functions `str()` and `repr()` should be treated with a certain degree of similarity.

We introduce a novel algorithm to compute quantitative generalizations of two variadic nominal input expressions. It combines techniques for handling variable binding with support for variadic expressions and fuzzy proximity or similarity. To compute generalizations that maximally preserve structural similarities, the algorithm distinguishes between atoms that can be bounded, function symbols where proximity/similarity is defined by a fuzzy relation, and two kinds of variables (term and hedge variables). It also provides detailed information about the differences and quantifies the structural similarities of the input expressions. The algorithm is formulated by a set of transformation rules so that it is universally applicable wherever fuzzy or precise comparison of variadic and binder-rich representations is required.

keywords: Generalization, anti-unification, variadic structures, nominal expressions, quantitative theories.

Contents

1	Introduction	2
2	Notions, notation, terminology	3
3	Quantitative rigid nominal generalization	7
4	The algorithm $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ for linear generalizations	12
5	The general case: the algorithm $\text{VNAU}(\mathcal{R}_{\mathcal{F}})$	16
6	Examples	19
7	Final remarks	23

1 Introduction

The generalization problem, also known as anti-unification problem, is concerned with discovering similarities in structured or semi-structured data, which arises in a wide range of circumstances and in different disguise, for instance as HTML DOM, XML, JSON, propositional formula, proof tree, abstract syntax tree (AST), etc. Given two input expressions t_1 and t_2 , their generalization is an expression r that preserves some similarities that appear in t_1 and in t_2 , and marks the differences by incorporating variables in r . Interesting generalizations are the least general ones (lgg), that is, those that preserve as many similarities of the input expressions as possible. The generalization problem has been introduced in [21, 22] and has already been studied in various different languages (see [10] for a survey). When dealing with semi-structured data, variadic languages (in which function symbols do not have a fixed arity) are required, as suggested in [13]. On the other hand, binder-rich languages, like higher-order patterns [18] and nominal terms [12, 20] allow for alpha-equivalent renaming of bound variables. Such alpha-equivalence comes in handy when comparing, for instance, two formulas with quantified variables, or two pieces of program code (represented as ASTs) where variable names are bounded within a certain scope. The generalization problem has been studied for (fixed arity) higher-order patterns and nominal terms [5–7, 24]. However, it has not been investigated in any setting that combines variadic functions and binding structures, apart from a preliminary report by the same authors [4].

We study the generalization problem in the variadic nominal language, which distinguishes between atoms that can be bound, term variables that can be instantiated by a single nominal term, hedge variables that can be instantiated by a (possibly empty) sequence of terms (called hedge), and variadic functions that can be applied to any hedge of arguments. The language is further enriched with a fuzzy relation that allows for flexible specification of proximal function symbols, i.e., the degree or probability by which two different symbols represent the same concept. Such relations come in handy when the underlying data is imprecise due to possible data loss, or when different names represent the same concept by a certain degree or probability (e.g., in Python the functions `str()` and `repr()` often return the same value, or in plagiarism detection when small modifications of names should be ignored). Generalizations are quantified by generalization degrees which are combinations of values given by the fuzzy relation, expressing proximity or similarity utilizing some T-norm. Quantitative generalizations have been studied in [1, 15, 19] for fixed arity languages without binding structures.

We formulate a rule-based algorithm that computes generalizations of two variadic nominal input expressions modulo proximity of function symbols and alpha-equivalence of the nominal language. This setting comes with some key challenges. First, freshness constraints of the nominal language can lead to nonexistence of an lgg for two given input expressions, as shown in [5]. Second, hedge variables in the variadic language can lead to exponentially many lgs with respect to the input size (see [13]). Finally, since fuzzy relations are not guaranteed to be transitive, one can not rely on consecutive comparison of terms, which is frequently used in rule formulations. To overcome those issues we formulate rules that do not rely on transitivity and combine techniques from [5] and [13]. In particular, we use the idea of a rigidity function to reduce the computational complexity [13] and consider a finite set of atoms [5]. Putting all this together leads to an algorithm that is parametric by a fuzzy relation, a T-norm, a rigidity function, and a set of atoms. We prove that it computes lgs of the given input expressions, considering all those parameters.

The paper is organized in the following way. Section 2 introduces the necessary notions. In Section 3, we define the problem and illustrate important properties. The base algorithm

(for linear generalizations) is introduced in Section 4 and is extended for the general case in Section 5. Section 6 contains a couple of illustrating examples. A summarizing discussion and final remarks can be found in Section 7.

2 Notions, notation, terminology

This section defines important notions that are used throughout the work. For more details we refer the reader to related work, e.g., [5, 11, 13–16].

2.1 Expressions, substitutions, freshness contexts

The alphabet of variadic nominal expressions considers four pairwise disjoint sets: a countable infinite set of *atoms* $\mathbf{A} = \{a, b, c, \dots\}$, a countable set of *variadic function symbols* $\Sigma = \{f, g, h, \dots\}$, a countable infinite set of *term variables* $\mathbf{x} = \{x, y, z, \dots\}$, and a countable infinite set of *hedge variables* $\mathbf{X} = \{X, Y, Z, \dots\}$. Like in the standard nominal setting, we consider *permutations* of atoms, say π , with finite support $\text{supp}(\pi) := \{a \in \mathbf{A} \mid \pi(a) \neq a\}$. The inverse of π is denoted by π^{-1} , and Id is the identity permutation (i.e., $\text{supp}(Id) = \emptyset$). Permutations are represented by a finite sequence of swappings, e.g., $(ab)(ac)$. By doing so, permutation composition simply reduces to sequence concatenation, and the inverse corresponds to reversing the sequence.

Variadic nominal expressions (expressions, for short) are either terms t or hedges (finite sequences) \tilde{s} , given by the grammar:

$$t ::= a \mid \pi \cdot x \mid a.t \mid f(\tilde{s}), \quad r ::= \pi \cdot X \mid t, \quad \tilde{s} ::= r_1, \dots, r_n, \quad n \geq 0,$$

where a is an *atom*, $\pi \cdot x$ is a *term suspension*, $a.t$ denotes the *abstraction* of atom a in the term t , $f(\tilde{s})$ is a *function application* of the variadic function f to a (possibly empty) hedge of arguments \tilde{s} , and $\pi \cdot X$ is a *hedge suspension*. To improve readability, hedges are sometimes written within parentheses, like (r_1, \dots, r_n) . A singleton hedge that consists of a term t is considered the same as t itself. Hedges are assumed to be flat, i.e., there is no difference between $\tilde{s}_1, (\tilde{s}_2), \tilde{s}_3$ and $\tilde{s}_1, \tilde{s}_2, \tilde{s}_3$. The empty hedge is denoted by ε . We use ℓ, τ, u to emphasize that we are talking about either type of expression, i.e., a hedge or a term. The notion *variable* refers to either a hedge or a term variable, and χ denotes some variable from $\mathbf{x} \cup \mathbf{X}$. *Suspension* simply refers to either an term or a hedge suspension.

The *effect of a swapping* over an atom is defined by $(a b) \cdot a = b$, $(a b) \cdot b = a$ and $(a b) \cdot c = c$, when $c \notin \{a, b\}$. It is extended to the rest of expressions: $(a b) \cdot (\pi \cdot \chi) = (a b) \pi \cdot \chi$ (where $(a b) \pi$ is the composition of π and $(a b)$), $(a b) \cdot (c.t) = ((a b) \cdot c) \cdot ((a b) \cdot t)$, $(a b) \cdot f(\tilde{s}) = f((a b) \cdot \tilde{s})$, and $(a b) \cdot (r_1, \dots, r_n) = ((a b) \cdot r_1, \dots, (a b) \cdot r_n)$. The *effect of a permutation* is defined by $(a_1 b_1) \cdots (a_n b_n) \cdot t = (a_1 b_1) \cdot ((a_2 b_2) \cdots (a_n b_n) \cdot t)$. The effect of the empty permutation is $Id \cdot \ell = \ell$. We write χ as a shortcut of $Id \cdot \chi$ if it is clear from the context.

The i th element of a hedge \tilde{s} is denoted by $\tilde{s}|_i$, and $\tilde{s}|_i^j$ denotes the subhedge from the i th element to the j th element, both included. The case $j < i$ corresponds to the empty hedge ε . The *length* of a hedge \tilde{s} is the number of elements in it, written as $|\tilde{s}|$. Terms and hedge suspensions are treated as singleton hedges, i.e., the above notions naturally extend to them. The *head* of a term or hedge suspension r is defined as $\text{head}(a) = a$, $\text{head}(\pi \cdot \chi) = \chi$, $\text{head}(f(\tilde{s})) = f$, $\text{head}(a.t) = .$, and the head of a hedge $\text{head}(r_1, \dots, r_n)$ is the word $\text{head}(r_1) \dots \text{head}(r_n)$.¹

¹It is assumed that some special characters, e.g., " $\cup, ()$ ", do not occur in the alphabet.

Substitutions, denoted by σ, γ, \dots , are finite mappings from variables to expressions, under the condition that term variables are mapped to terms. They are defined by a finite set of *assignments* of the form $\{\chi_1 \mapsto u_1, \dots, \chi_n \mapsto u_n\}$. Given a substitution $\sigma = \{\chi_1 \mapsto u_1, \dots, \chi_n \mapsto u_n\}$, the *domain* of σ is the finite set $\text{dom}(\sigma) := \{\chi_1, \dots, \chi_n\}$, and the *range* is $\text{ran}(\sigma) := \{u_1, \dots, u_n\}$. For all variables $\chi \in (\mathbf{x} \cup \mathbf{X}) \setminus \text{dom}(\sigma)$, the mapping is extended as being the identity suspension $\text{Id} \cdot \chi$. The variables in $\text{dom}(\sigma)$ are also said to be *instantiated* by σ . The identity substitution with the empty domain is denoted by id . The *effect of a substitution* σ on an expression u , denoted as $u\sigma$, is inductively defined by

$$a\sigma = a, \quad (\pi \cdot \chi)\sigma = \pi \cdot \chi\sigma, \quad (a.t)\sigma = a.(t\sigma), \quad f(\tilde{s})\sigma = \tilde{s}\sigma, \quad (r_1, \dots, r_n)\sigma = r_1\sigma, \dots, r_n\sigma.$$

Freshness constraints are pairs of the form $a\#\chi$ (a is fresh for χ), forbidding the free occurrence of some atom a in instantiations of some variable χ . A *freshness context* (denoted by ∇, Γ, \dots) is a finite set of freshness constraints. Given a substitution σ and a freshness context $\nabla = \{a_1\#\chi_1, \dots, a_n\#\chi_n\}$, we say that σ *respects* ∇ iff $\nabla \vdash a_i\#\chi_i\sigma$, for all $1 \leq i \leq n$, where $\nabla \vdash a\#\ell$ is defined as follows:

$$\begin{array}{c} \frac{a \neq b}{\nabla \vdash a\#b} \text{ (#atom)} \quad \frac{\nabla \vdash a\#\tilde{s}}{\nabla \vdash a\#f(\tilde{s})} \text{ (#appl)} \quad \frac{\nabla \vdash a\#r_1 \quad \dots \quad \nabla \vdash a\#r_n}{\nabla \vdash a\#(r_1, \dots, r_n)} \text{ (#hedge)} \\ \\ \frac{}{\nabla \vdash a\#a.t} \text{ (#abs1)} \quad \frac{a \neq b \quad \nabla \vdash a\#t}{\nabla \vdash a\#b.t} \text{ (#abs2)} \quad \frac{(\pi^{-1} \cdot a\#\chi) \in \nabla}{\nabla \vdash a\#\pi \cdot \chi} \text{ (#susp)} \end{array}$$

Given a freshness context ∇ and a substitution σ that respects ∇ , the *instance of ∇ under σ* , written $\nabla\sigma$, is the smallest freshness context Γ such that $\Gamma \vdash a\#\chi\sigma$ for all $a\#\chi \in \nabla$. Note that Γ can easily be obtained by the above rules.

An *expression-in-context* or, shortly, *eic* is a pair of a freshness context ∇ and an expression u , written $\langle \nabla, u \rangle$. We use E, G to denote expressions-in-context.

The *difference set* of two permutations π and π' is defined as $\text{ds}(\pi, \pi') = \{a \in \mathbf{A} \mid \pi \cdot a \neq \pi' \cdot a\}$. The set of atoms that occur in a suspension $\pi \cdot \chi$ is defined as $\text{supp}(\pi)$. We use $\text{atoms}(u)$, $\text{atoms}(\nabla)$, $\text{atoms}(\langle \nabla, u \rangle)$ and $\text{atoms}(\sigma)$ to denote the set of all atoms that occur in an expression u , a freshness context ∇ , an eic $\langle \nabla, u \rangle$ and in $\text{ran}(\sigma)$, respectively. An expression u , a freshness context ∇ , an eic $\langle \nabla, u \rangle$, or a substitution σ is *based on a set of atoms A* (A -based), iff, respectively, $\text{atoms}(u) \subseteq A$, $\text{atoms}(\nabla) \subseteq A$, $\text{atoms}(\langle \nabla, u \rangle) \subseteq A$, or $\text{atoms}(\sigma) \subseteq A$. A permutation π is A -based iff $\text{supp}(\pi) \subseteq A$.

2.2 Fuzzy relations and quantitative α -equivalence

A (binary) *fuzzy relation* on a set S is a mapping from $S \times S$ to the real interval $[0, 1]$. A fuzzy relation is *crisp* if its range is $\{0, 1\}$, i.e., it represents an ordinary relation.

A *T-norm* \otimes is an associative, commutative, non-decreasing binary operation on $[0, 1]$ with 1 as the unit element. Some prominent T-norms are Gödel or minimum T-norm ($v_1 \otimes v_2 = \min(v_1, v_2)$), product T-norm ($v_1 \otimes v_2 = v_1 * v_2$) and Łukasiewicz T-norm ($v_1 \otimes v_2 = \max(0, v_1 + v_2 - 1)$), where $(v_1, v_2) \in [0, 1] \times [0, 1]$. The minimum and product T-norms do not have zero divisors (i.e., for them, $v_1 \otimes v_2 = 0$ implies $v_1 = 0$ or $v_2 = 0$), while the Łukasiewicz T-norm permits them.

Definition 1 (Proximity and similarity relations). *A fuzzy relation \mathcal{F} on a set S is called a proximity relation, if it is reflexive ($\mathcal{F}(s, s) = 1$ for all $s \in S$) and symmetric ($\mathcal{F}(s_1, s_2) = \mathcal{F}(s_2, s_1)$ for all $s_1, s_2 \in S$). It is called a similarity relation with respect to the T-norm \otimes if it, in addition, is \otimes -transitive ($\mathcal{F}(s_1, s_2) \geq \mathcal{F}(s_1, s) \otimes \mathcal{F}(s, s_2)$ for any $s_1, s_2, s \in S$).*

We sometimes write ‘ \otimes -similarity relation’ to make the T-norm explicit. In the role of S , we would like to take the set of expressions of our language. We assume a fuzzy relation \mathcal{F} is defined on the alphabet $\mathbf{A} \cup \Sigma \cup \mathbf{x} \cup \mathbf{X}$ in such a way that

- $\mathcal{F}(\chi, \chi') = 0$ for all $\chi, \chi' \in \mathbf{x} \cup \mathbf{X}$ with $\chi \neq \chi'$,
- $\mathcal{F}(a, b) = 0$ for all $a, b \in \mathbf{A}$ with $a \neq b$,
- $\mathcal{F}(\chi, f) = \mathcal{F}(f, \chi) = 0$ for all $\chi \in \mathbf{x} \cup \mathbf{X}$ and $f \in \Sigma$,
- $\mathcal{F}(\chi, a) = \mathcal{F}(a, \chi) = 0$ for all $\chi \in \mathbf{x} \cup \mathbf{X}$ and $a \in \mathbf{A}$,
- $\mathcal{F}(a, f) = \mathcal{F}(f, a) = 0$ for all $a \in \mathbf{A}$ and $f \in \Sigma$.

In the following, any fuzzy relation on the alphabet fulfills those properties.

Definition 2. *Given a fuzzy relation \mathcal{F} on the alphabet and a T-norm \otimes , we define a fuzzy relation $\approx_{\mathcal{F}}$ on expressions under a given context below. The notation $\nabla \vdash u_1 \approx_{\mathcal{F}} u_2 = \mathfrak{d}$ means that $\approx_{\mathcal{F}}(u_1, u_2)$ equals to $\mathfrak{d} \in [0, 1]$ under context ∇ .*

$$\frac{}{\nabla \vdash \varepsilon \approx_{\mathcal{F}} \varepsilon = 1} (\approx_{\mathcal{F}} \varepsilon) \qquad \frac{\nabla \vdash r_1 \approx_{\mathcal{F}} r'_1 = \mathfrak{d}_1 \ \cdots \ \nabla \vdash r_n \approx_{\mathcal{F}} r'_n = \mathfrak{d}_n \quad n > 1}{\nabla \vdash (r_1, \dots, r_n) \approx_{\mathcal{F}} (r'_1, \dots, r'_n) = \mathfrak{d}_1 \otimes \cdots \otimes \mathfrak{d}_n} (\approx_{\mathcal{F}} \text{hedge})$$

$$\frac{}{\nabla \vdash a \approx_{\mathcal{F}} a = 1} (\approx_{\mathcal{F}} \text{atom}) \qquad \frac{\nabla \vdash \tilde{s} \approx_{\mathcal{F}} \tilde{q} = \mathfrak{d}}{\nabla \vdash f(\tilde{s}) \approx_{\mathcal{F}} f'(\tilde{q}) = \mathcal{F}(f, f') \otimes \mathfrak{d}} (\approx_{\mathcal{F}} \text{appl})$$

$$\frac{\nabla \vdash t \approx_{\mathcal{F}} t' = \mathfrak{d}}{\nabla \vdash a.t \approx_{\mathcal{F}} a.t' = \mathfrak{d}} (\approx_{\mathcal{F}} \text{abs1}) \qquad \frac{\nabla \vdash t \approx_{\mathcal{F}} (a b).t' = \mathfrak{d} \quad \nabla \vdash a \# b.t' = \mathfrak{d}}{\nabla \vdash a.t \approx_{\mathcal{F}} b.t' = \mathfrak{d} \quad a \neq b} (\approx_{\mathcal{F}} \text{abs2})$$

$$\frac{a \# \chi \in \nabla \text{ for all } a \in \text{ds}(\pi, \pi')}{\nabla \vdash \pi \cdot \chi \approx_{\mathcal{F}} \pi' \cdot \chi = 1} (\approx_{\mathcal{F}} \text{susp})$$

Besides, we also have the following default rule, which applies if no other rule is applicable:

$$\frac{\text{For any other choice of } u_1 \text{ and } u_2}{\nabla \vdash u_1 \approx_{\mathcal{F}} u_2 = 0} (\approx_{\mathcal{F}} \text{else})$$

The following lemma characterizes basic properties of the $\approx_{\mathcal{F}}$ relation:

Lemma 3. *Let u_1 and u_2 be two expressions and \mathcal{F} be a fuzzy relation on the alphabet together with some T-norm.*

1. *If u_1 and u_2 have different structures (different sets of positions), then $\nabla \vdash u_1 \approx_{\mathcal{F}} u_2 = 0$ for any ∇ .*
2. *$\nabla \vdash a \# u_1$ and $\nabla \vdash u_1 \approx_{\mathcal{F}} u_2 > 0$ imply $\nabla \vdash a \# u_2$.*
3. *$\nabla \vdash \pi \cdot u_1 \approx_{\mathcal{F}} u_2 = \mathfrak{d}$ implies $\nabla \vdash u_1 \approx_{\mathcal{F}} \pi^{-1} \cdot u_2 = \mathfrak{d}$.*
4. *$\nabla \vdash u_1 \approx_{\mathcal{F}} u_2 = \mathfrak{d}$ iff $\nabla \vdash \pi \cdot u_1 \approx_{\mathcal{F}} \pi \cdot u_2 = \mathfrak{d}$ for any permutation π .*
5. *$\nabla \vdash a \# u_1$ for all $a \in \text{ds}(\pi, \pi')$, iff $\nabla \vdash \pi \cdot u_1 \approx_{\mathcal{F}} \pi' \cdot u_1 = 1$.*

Proof. The items 1 and 2 are proved by induction of the derivation cases of $\approx_{\mathcal{F}}$. The items 3–5 are proved by induction on the structure of r_1 . In the proof of item 4, we will also use item 3. The statements 2–5 are $\approx_{\mathcal{F}}$ -counterparts of Lemmas 2–5 formulated for \approx in [2]. \square

Lemma 4. *Let \mathcal{F} be a fuzzy relation on $\mathbf{A} \cup \Sigma \cup \mathbf{x} \cup \mathbf{X}$, \otimes be a T -norm, and $\approx_{\mathcal{F}}$ be the corresponding fuzzy relation on expressions over $\mathbf{A} \cup \Sigma \cup \mathbf{x} \cup \mathbf{X}$ for an arbitrary context ∇ . If \mathcal{F} is a \otimes -similarity relation, then $\approx_{\mathcal{F}}$ is also a \otimes -similarity relation.*

Proof. In the proof we will use the equivariance property of freshness from [2]: $\nabla \vdash a \# r$ iff $\nabla \vdash \pi \cdot a \# \pi \cdot r$ for any π .

Reflexivity follows directly from the definition of $\approx_{\mathcal{F}}$ and reflexivity of \mathcal{F} . Symmetry of $\approx_{\mathcal{F}}$ can be proved analogously to symmetry of \approx , taking into account the symmetry of \mathcal{F} and commutativity of \otimes . For similarity, we need to show that \otimes -transitivity holds:

$$\text{If } \nabla \vdash u_1 \approx_{\mathcal{F}} u = \mathfrak{d}_1 \text{ and } \nabla \vdash u \approx_{\mathcal{F}} u_2 = \mathfrak{d}_2, \text{ then } \nabla \vdash u_1 \approx_{\mathcal{F}} u_2 = \mathfrak{d} \geq \mathfrak{d}_1 \otimes \mathfrak{d}_2.$$

Assume without loss of generality that $\mathfrak{d}_1 \otimes \mathfrak{d}_2 > 0$. Then we get $\mathfrak{d}_1 > 0$ and $\mathfrak{d}_2 > 0$, which by Lemma 3 means that u_1, u_2 , and u have the same structure. We proceed by induction on this structure.

- (a) When u_1, u_2 , and u are atoms, the property holds as they all should be the same.
- (b) When $u_1 = f_1(\tilde{r}_1)$, $u_2 = f_2(\tilde{r}_2)$, and $u = f(\tilde{r})$, then we have to show

$$\mathcal{F}(f_1, f_2) \otimes \mathfrak{d} \geq \mathcal{F}(f_1, f) \otimes \mathfrak{d}_1 \otimes \mathcal{F}(f, f_2) \otimes \mathfrak{d}_2, \quad (1)$$

where $\nabla \vdash \tilde{r}_1 \approx_{\mathcal{F}} \tilde{r}_2 = \mathfrak{d}$, $\nabla \vdash \tilde{r}_1 \approx_{\mathcal{F}} \tilde{r} = \mathfrak{d}_1$, and $\nabla \vdash \tilde{r} \approx_{\mathcal{F}} \tilde{r}_2 = \mathfrak{d}_2$. By the fact that \mathcal{F} is \otimes -similarity, we have $\mathcal{F}(f_1, f_2) \geq \mathcal{F}(f_1, f) \otimes \mathcal{F}(f, f_2)$. By the induction hypothesis, $\mathfrak{d} \geq \mathfrak{d}_1 \otimes \mathfrak{d}_2$. Therefore, by monotonicity of \otimes , we get (1).

- (c) The case $u_1 = u_2 = u = \varepsilon$ is trivial.
- (d) When $u_1 = (r_1, \dots, r_n)$, $u_2 = (q_1, \dots, q_n)$, and $u = (s_1, \dots, s_n)$, $n > 1$, then the property follows from the definition of $\approx_{\mathcal{F}}$ for such hedges, the induction hypothesis, and monotonicity of \otimes .
- (e) When $u_1 = \pi_1 \cdot x_1$, $u_2 = \pi_2 \cdot x_2$, and $u = \pi \cdot x$, the proof is similar to the proof of transitivity of \approx , where no fuzzy relations are involved.
- (f) When $u_1 = a_1 \cdot t_1$, $u_2 = a_2 \cdot t_2$, and $u = a \cdot t$, then we have to show that
 - if $\nabla \vdash t_1 \approx_{\mathcal{F}} (a_1 a) \cdot t = \mathfrak{d}_1$, $\nabla \vdash a_1 \# a \cdot t$, $\nabla \vdash t \approx_{\mathcal{F}} (a a_2) \cdot t_2 = \mathfrak{d}_2$, $\nabla \vdash a \# a_2 \cdot t_2$,
 - then $\nabla \vdash t_1 \approx_{\mathcal{F}} (a_1 a_2) \cdot t_2 = \mathfrak{d}$, $\nabla \vdash a_1 \# a_2 \cdot t_2$ and $\mathfrak{d} \geq \mathfrak{d}_1 \otimes \mathfrak{d}_2$.

The proof follows the structure of the analogous case for \approx from [2]:

- $a_1 = a = a_2$: The result follows by induction hypothesis.
- $a_1 = a \neq a_2$: By definition of $\approx_{\mathcal{F}}$, we have $\nabla \vdash t_1 \approx_{\mathcal{F}} t = \mathfrak{d}_1$ and $\nabla \vdash t \approx_{\mathcal{F}} (a a_2) \cdot t_2 = \mathfrak{d}_2$ with $\nabla \vdash a \# a_2 \cdot t_2$. By definition of $\approx_{\mathcal{F}}$ and IH, we get $\nabla \vdash t_1 \approx_{\mathcal{F}} (a a_2) \cdot t_2 = \mathfrak{d}_1 \geq \mathfrak{d}_1 \otimes \mathfrak{d}_2$. But $a_1 = a_2$. Hence, we get $\nabla \vdash t_1 \approx_{\mathcal{F}} (a_1 a_2) \cdot t_2 = \mathfrak{d} \geq \mathfrak{d}_1 \otimes \mathfrak{d}_2$ and $\nabla \vdash a_1 \# a_2 \cdot t_2$.

- $a_1 \neq a = a_2$: We have $\nabla \vdash t_1 \approx_{\mathcal{F}} (a_1 a_2) \cdot t = \mathfrak{d}_1$, $\nabla \vdash a_1 \# a_2 \cdot t$, and $\nabla \vdash t \approx_{\mathcal{F}} t_2 = \mathfrak{d}_2$. From the latter, by item 4 of Lemma 3, we get $\nabla \vdash (a_1 a_2) \cdot t \approx_{\mathcal{F}} (a_1 a_2) \cdot t_2 = \mathfrak{d}_2$. By definition of $\approx_{\mathcal{F}}$ and IH, we get $\nabla \vdash t_1 \approx_{\mathcal{F}} (a_1 a_2) \cdot t_2 = \mathfrak{d} \geq \mathfrak{d}_2 \otimes \mathfrak{d}_2$.
From $\nabla \vdash (a_1 a_2) \cdot t \approx_{\mathcal{F}} (a_1 a_2) \cdot t_2 = \mathfrak{d}_2$, by definition of $\approx_{\mathcal{F}}$, we get $\nabla \vdash a_1 \cdot (a_1 a_2) \cdot t \approx_{\mathcal{F}} a_1 \cdot (a_1 a_2) \cdot t_2 = \mathfrak{d}_2$. From $\nabla \vdash a_1 \# a_2 \cdot t$, by equivariance of freshness (see [2]), we get $\nabla \vdash a_2 \# a_1 \cdot (a_1 a_2) \cdot t$. From this, since by assumption $\mathfrak{d}_2 > 0$, item 2 of Lemma 3 gives $\nabla \vdash a_2 \# a_1 \cdot (a_1 a_2) \cdot t_2$. The latter, by equivariance of freshness, gives $\nabla \vdash a_1 \# a_2 \cdot t_2$.
- $a_1 \neq a, a_1 = a_2$: We have $\nabla \vdash t_1 \approx_{\mathcal{F}} (a_2 a) \cdot t = \mathfrak{d}_1$ and $\nabla \vdash t \approx_{\mathcal{F}} (a a_2) \cdot t_2 = \mathfrak{d}_2$. Then $\nabla \vdash (a a_2) \cdot t \approx_{\mathcal{F}} t_2 = \mathfrak{d}_2$ by item 3 of Lemma 3. From the definition of $\approx_{\mathcal{F}}$ and IH, we get $\nabla \vdash t_1 \approx_{\mathcal{F}} t_2 = \mathfrak{d} \geq \mathfrak{d}_1 \otimes \mathfrak{d}_2$. (We do not have to consider freshness constraints, since the applied rule is $(\approx_{\mathcal{F}} \text{abs1})$.)
- $a_1 \neq a, a \neq a_2, a_2 \neq a_1$: We should prove $\nabla \vdash t_1 \approx_{\mathcal{F}} (a_1 a_2) \cdot t_2 = \mathfrak{d} \geq \mathfrak{d}_1 \otimes \mathfrak{d}_2$ and $\nabla \vdash a_1 \# a_2 \cdot t_2$. We start with the freshness condition. By definition of $\approx_{\mathcal{F}}$, we have $\nabla \vdash a_1 \# a \cdot t$ and $\nabla \vdash t \approx_{\mathcal{F}} (a a_2) \cdot t_2 = \mathfrak{d}_2$. The latter implies $\nabla \vdash a \cdot t \approx_{\mathcal{F}} a \cdot (a a_2) \cdot t_2 = \mathfrak{d}_2$. By assumption, $\mathfrak{d}_2 > 0$, therefore, by item 2 of Lemma 3, we get $\nabla \vdash a_1 \# a \cdot (a a_2) \cdot t_2$ and, by equivariance of freshness, $\nabla \vdash a_1 \# a_2 \cdot t_2$.
Now we prove $\nabla \vdash t_1 \approx_{\mathcal{F}} (a_1 a_2) \cdot t_2 = \mathfrak{d} \geq \mathfrak{d}_1 \otimes \mathfrak{d}_2$. By item 4 of Lemma 3, from $\nabla \vdash t \approx_{\mathcal{F}} (a a_2) \cdot t_2 = \mathfrak{d}_2$ we have $\nabla \vdash (a_1 a) \cdot t \approx_{\mathcal{F}} (a_1 a) \cdot (a a_2) \cdot t_2 = \mathfrak{d}_2$. Since $\text{ds}((a a_2), (a_1 a) \cdot (a a_2)) = \{a_1, a\}$ and both atoms are fresh in t_2 , we get $\nabla \vdash (a_1 a) \cdot (a a_2) \cdot t_2 \approx_{\mathcal{F}} (a a_2) \cdot t_2 = 1$ by item 5 of Lemma 3. Applying IH gives $\nabla \vdash (a_1 a) \cdot t \approx_{\mathcal{F}} (a a_2) \cdot t_2 = \mathfrak{d}'_2 \geq (\mathfrak{d}_2 \otimes 1) = \mathfrak{d}_2$. On the other hand, we have $\nabla \vdash t_1 \approx_{\mathcal{F}} (a_1 a) \cdot t = \mathfrak{d}_1$. Applying IH again gives $\nabla \vdash t_1 \approx_{\mathcal{F}} (a_1 a_2) \cdot t_2 = \mathfrak{d} \geq \mathfrak{d}_1 \otimes \mathfrak{d}'_2 \geq \mathfrak{d}_1 \otimes \mathfrak{d}_2$.

□

3 Quantitative rigid nominal generalization

Throughout the rest of the paper, we work under the following two assumptions:

1. T-norms do not permit zero divisors.
2. Every proximity relation \mathcal{F} on the alphabet has the set $\{g \mid \mathcal{F}(f, g) > 0\}$ (the *proximity class* of f) finite for each $f \in \Sigma$. Since such a class is singleton for any element from $\mathbf{A} \cup \mathbf{x} \cup \mathbf{X}$, we get that the proximity class for each expression u under ∇ , defined as $\{u' \mid \nabla \vdash u \approx_{\mathcal{F}} u' > 0\}$, is finite.

Definition 5. Let $\langle \nabla_1, u_1 \rangle$ and $\langle \nabla_2, u_2 \rangle$ be two eics, and $\approx_{\mathcal{F}}$ be a proximity relation on expressions. We say that $\langle \nabla_1, u_1 \rangle$ is more general than $\langle \nabla_2, u_2 \rangle$ with respect to $\approx_{\mathcal{F}}$, written $\langle \nabla_1, u_1 \rangle \preceq_{\mathcal{F}} \langle \nabla_2, u_2 \rangle$, if there exists a substitution σ such that

- σ respects ∇_1 ,
- $\nabla_1 \sigma \subseteq \nabla_2$, and
- $\nabla_2 \vdash u_1 \sigma \approx_{\mathcal{F}} u_2 = \mathfrak{d} > 0$.

In such a case, we say that $\langle \nabla_1, u_1 \rangle$ matches $\langle \nabla_2, u_2 \rangle$ with substitution σ and degree \mathfrak{d} .

The number $\mathfrak{d} \in [0, 1]$ is called the generalization degree of $\langle \nabla_1, u_1 \rangle$ over $\langle \nabla_2, u_2 \rangle$ if \mathfrak{d} is the maximum of all degrees by which $\langle \nabla_1, u_1 \rangle$ matches $\langle \nabla_2, u_2 \rangle$ (with some substitution). We use $\langle \nabla_1, u_1 \rangle \preceq_{\mathcal{F}} \langle \nabla_2, u_2 \rangle = \mathfrak{d}$ to denote $\langle \nabla_1, u_1 \rangle \preceq_{\mathcal{F}} \langle \nabla_2, u_2 \rangle$ with generalization degree \mathfrak{d} .²

Note that there might be two eics $\langle \nabla_1, u_1 \rangle$ and $\langle \nabla_1, u'_1 \rangle$ with different generalization degrees over $\langle \nabla_2, u_2 \rangle$, even though $\nabla_1 \vdash u_1 \approx_{\mathcal{F}} u'_1 > 0$.

Lemma 6. Let $\langle \nabla_1, u_1 \rangle$ and $\langle \nabla_2, u_2 \rangle$ be eics, and \mathcal{F} be a proximity relation such that $\langle \nabla_1, u_1 \rangle \preceq_{\mathcal{F}} \langle \nabla_2, u_2 \rangle = \mathfrak{d}_1$ and $\langle \nabla_2, u_2 \rangle \preceq_{\mathcal{F}} \langle \nabla_1, u_1 \rangle = \mathfrak{d}_2$. Then $\mathfrak{d}_1 = \mathfrak{d}_2$.

Proof. There are substitutions σ_1, σ_2 such that $\nabla_2 \vdash u_1 \sigma_1 \approx_{\mathcal{F}} u_2 = \mathfrak{d}_1 > 0$ and $\nabla_1 \vdash u_2 \sigma_2 \approx_{\mathcal{F}} u_1 = \mathfrak{d}_2 > 0$. By inspecting the rules of Definition 2, it is easy to see that extending ∇_2 doesn't affect \mathfrak{d}_1 , since $\mathfrak{d}_1 > 0$. The same holds for ∇_1 and \mathfrak{d}_2 . Therefore, $\nabla_1 \cup \nabla_2 \vdash u_1 \sigma_1 \approx_{\mathcal{F}} u_2 = \mathfrak{d}_1$ and $\nabla_1 \cup \nabla_2 \vdash u_2 \sigma_2 \approx_{\mathcal{F}} u_1 = \mathfrak{d}_2$. Moreover, u_1 and u_2 have similar structure, i.e., the instantiations $u_1 \sigma_1$ and $u_2 \sigma_2$ do not introduce any function symbols (or atoms). By symmetry of $\approx_{\mathcal{F}}$ (Lemma 4) follows that $\mathfrak{d}_1 = \mathfrak{d}_2$. \square

We say that $\langle \nabla_1, u_1 \rangle$ and $\langle \nabla_2, u_2 \rangle$ are \mathcal{F} -*equi-general* and write $\langle \nabla_1, u_1 \rangle \simeq_{\mathcal{F}} \langle \nabla_2, u_2 \rangle$ if $\langle \nabla_1, u_1 \rangle \preceq_{\mathcal{F}} \langle \nabla_2, u_2 \rangle$ and $\langle \nabla_2, u_2 \rangle \preceq_{\mathcal{F}} \langle \nabla_1, u_1 \rangle$. It is reflexive: $\langle \nabla, u \rangle \simeq_{\mathcal{F}} \langle \nabla, u \rangle = 1$. Since $\langle \nabla_1, u_1 \rangle \preceq_{\mathcal{F}} \langle \nabla_2, u_2 \rangle = \mathfrak{d} = \langle \nabla_2, u_2 \rangle \preceq_{\mathcal{F}} \langle \nabla_1, u_1 \rangle$, the induced relation $\simeq_{\mathcal{F}}$ is also symmetric, and, hence, is a proximity relation: $\langle \nabla_1, u_1 \rangle \simeq_{\mathcal{F}} \langle \nabla_2, u_2 \rangle = \mathfrak{d} = \langle \nabla_2, u_2 \rangle \simeq_{\mathcal{F}} \langle \nabla_1, u_1 \rangle$. The notation $\langle \nabla_1, u_1 \rangle \prec_{\mathcal{F}} \langle \nabla_2, u_2 \rangle$ means $\langle \nabla_1, u_1 \rangle \preceq_{\mathcal{F}} \langle \nabla_2, u_2 \rangle$ and $\langle \nabla_1, u_1 \rangle \not\preceq_{\mathcal{F}} \langle \nabla_2, u_2 \rangle$. The generalization degree for $\prec_{\mathcal{F}}$ comes directly from $\preceq_{\mathcal{F}}$.

Example 7. Consider the proximity relation \mathcal{F} defined as $\mathcal{F}(f_1, f_2) = 0.7$ and $\mathcal{F}(g_1, g_2) = 0.5$, the minimum T-norm \otimes , and the eics $E_1 = \langle \nabla, u_1 \rangle$, $E_2 = \langle \nabla, u_2 \rangle$ and $E_3 = \langle \emptyset, u_3 \rangle$ where $u_1 = a.f_1(X, a, g_1, (a\ c)\cdot X)$, $u_2 = b.f_2((b\ d)\cdot Y, b, g_2, (b\ c)(b\ d)\cdot Y)$, $u_3 = c.f_1(c, g_2)$, and $\nabla = \{b\#X, a\#Y\}$. Those eics are related as follows:

- $E_1 \preceq_{\mathcal{F}} E_2 = 0.5$ because the witness substitution $\sigma = \{X \mapsto (a\ b)(b\ d)\cdot Y\}$ respects ∇ , $\nabla\sigma \subseteq \nabla$, and applying $u_1\sigma$ gives $u'_1 = a.f_1((a\ b)(b\ d)\cdot Y, a, g_1, (a\ c)(a\ b)(b\ d)\cdot Y)$. Then, we get $\nabla \vdash u'_1 \approx_{\mathcal{F}} u_2 = 0.5$, because
 - $\nabla \vdash u'_1 \approx_{\mathcal{F}} u_2 = \nabla \vdash u'_1 \approx_{\mathcal{F}} (a\ b)\cdot u_2$, since $\nabla \vdash a\#u_2$,
 - $(a\ b)\cdot u_2 = a.f_2((a\ b)(b\ d)\cdot Y, a, g_2, (a\ b)(b\ c)(b\ d)\cdot Y)$,
 - $\nabla \vdash u'_1 \approx_{\mathcal{F}} (a\ b)\cdot u_2 = \mathcal{F}(f_1, f_2) \otimes \mathcal{F}(g_1, g_2) = 0.5$. Notice that in this calculation the suspensions in both expressions are proximal with degree 1. While for $(a\ b)(b\ d)\cdot Y$ in both expressions this is obvious, $\nabla \vdash (a\ c)(a\ b)(b\ d)\cdot Y \approx_{\mathcal{F}} (a\ b)(b\ c)(b\ d)\cdot Y = 1$ since $(a\ c)(a\ b)(b\ d)$ and $(a\ b)(b\ c)(b\ d)$ are equal permutations.
 - Finally, there is no other substitution σ' such that $\nabla \vdash u_1\sigma' \approx_{\mathcal{F}} u_2 > 0.5$.
- $E_2 \preceq_{\mathcal{F}} E_1 = 0.5$, by using the witness substitution $\{Y \mapsto (b\ d)(a\ b)\cdot X\}$ and the same reasoning steps as above.
- $E_1 \prec_{\mathcal{F}} E_3 = 0.5$, by using the witness substitution $\{X \mapsto \varepsilon\}$.
- $E_2 \prec_{\mathcal{F}} E_3 = 0.7$, by using the witness substitution $\{Y \mapsto \varepsilon\}$.

Note that replacing the minimum T-norm by the product T-norm affects only the generalization degrees and leads to $E_1 \simeq_{\mathcal{F}} E_2 = 0.35$, $E_1 \prec_{\mathcal{F}} E_3 = 0.5$, and $E_2 \prec_{\mathcal{F}} E_3 = 0.7$. \blacktriangleright

²This induces a fuzzy relation again, which is reflexive: $\langle \nabla, u \rangle \preceq_{\mathcal{F}} \langle \nabla, u \rangle = 1$.

The $\preceq_{\mathcal{F}}$ relation, in general, is not transitive, which follows directly from $\approx_{\mathcal{F}}$ not being transitive. For instance, take a proximity relation $\approx_{\mathcal{F}}$ on expressions induced by the proximity relation \mathcal{F} defined as $\mathcal{F}(f_1, f_2) = 0.7$ and $\mathcal{F}(f_2, f_3) = 0.8$, and the eics $\langle \nabla, f_1 \rangle$, $\langle \nabla, f_2 \rangle$ and $\langle \nabla, f_3 \rangle$ where ∇ and the T-norm are arbitrary. Then $\langle \nabla, f_1 \rangle \preceq_{\mathcal{F}} \langle \nabla, f_2 \rangle = 0.7$ and $\langle \nabla, f_2 \rangle \preceq_{\mathcal{F}} \langle \nabla, f_3 \rangle = 0.8$, but $\langle \nabla, f_1 \rangle \not\preceq_{\mathcal{F}} \langle \nabla, f_3 \rangle$. However, the following relations can be easily shown for any proximity relation $\approx_{\mathcal{F}}$ and eics E_1, E_2 , and E_3 :

- If $E_1 \preceq_{\mathcal{F}} E_2$ with degree \mathfrak{d} and $E_2 \preceq E_3$ (syntactically more general), then $E_1 \preceq_{\mathcal{F}} E_3$ with degree \mathfrak{d} . Note that $\approx_{\mathcal{F}}$ doesn't need to be transitive in that case, since the function symbols in E_2 have an exact correspondence in E_3 .
- If $E_1 \preceq E_2$ (syntactically) and $E_2 \preceq_{\mathcal{F}} E_3$ with degree \mathfrak{d} , then $E_1 \preceq_{\mathcal{F}} E_3$ with degree \mathfrak{d} . In this case, the function symbols in E_1 have an exact correspondence in E_2 .
- If $\approx_{\mathcal{F}}$ is an \otimes -similarity relation, $E_1 \preceq_{\mathcal{F}} E_2$ with degree \mathfrak{d}_1 , and $E_2 \preceq_{\mathcal{F}} E_3$ with degree \mathfrak{d}_2 , then $E_1 \preceq_{\mathcal{F}} E_3$ with degree $\mathfrak{d} \geq \mathfrak{d}_1 \otimes \mathfrak{d}_2$.

This means that $\preceq_{\mathcal{F}}$ is transitive if and only if the proximity relation \mathcal{F} is transitive (i.e., it is a similarity), and the corresponding equi-generality relation $\simeq_{\mathcal{F}}$ induces a similarity relation if and only if \mathcal{F} is a similarity relation.

Definition 8 (\mathcal{F} -Generalization). *An expression-in-context G is a generalization of expressions-in-context E_1 and E_2 with respect to a proximity relation $\approx_{\mathcal{F}}$, or, briefly, \mathcal{F} -generalization of E_1 and E_2 , if $G \preceq_{\mathcal{F}} E_1$ and $G \preceq_{\mathcal{F}} E_2$. Further, G is a least general \mathcal{F} -generalization (\mathcal{F} -lgg) of E_1 and E_2 if there exists no other \mathcal{F} -generalization G' of E_1 and E_2 such that $G \prec_{\mathcal{F}} G'$.*

A drawback of the nominal setting is that the relation $\prec_{\mathcal{F}}$ is not well-founded. That is, an infinite chain of strictly less general eics can be constructed by simply adding freshness constraints. This is a crucial observation that has been discussed in [5] for standard nominal terms. Therefore, it has been suggested to consider eics based on a *finite set* of atoms $A \subset \mathbf{A}$, meaning that A -based eics may contain atoms only from A . This is the assumption we make throughout the paper. In particular, for a finite set of atoms A and a set τ of all A -based eics, the relation $\preceq_{\mathcal{F}}$ is redefined to be $\preceq_{\mathcal{F}} \cap \tau \times \tau$. Subsequent definitions, like the generalization degree, are based on that redefinition.

Variadic nominal \mathcal{F} -generalization problems might have too many lgg, even if we take \mathcal{F} crisp (i.e., each symbol is proximal only to itself with degree 1):

Example 9. Let $E_1 = \langle \emptyset, f(a, b, b, a) \rangle$ and $E_2 = \langle \emptyset, f(c, c) \rangle$ with $A = \{a, b, c\}$. Assume \mathcal{F} is crisp. The following eics are pairwise incomparable \mathcal{F} -lgs of E_1 and E_2 :

$$\begin{aligned}
G_1 &= \langle \{b\#y, a\#Z, c\#Z\}, f(y, (a\ b)\cdot y, Z, (a\ b)\cdot Z) \rangle \\
G_2 &= \langle \{b\#y, a\#Z, c\#Z\}, f(y, Z, (a\ b)\cdot y, (a\ b)\cdot Z) \rangle \\
G_3 &= \langle \{b\#y, a\#Z, c\#Z\}, f(y, Z, Z, y) \rangle \\
G_4 &= \langle \{a\#y, b\#Z, c\#Z\}, f(Z, y, y, Z) \rangle \\
G_5 &= \langle \{a\#y, b\#Z, c\#Z\}, f(Z, y, (a\ b)\cdot Z, (a\ b)\cdot y) \rangle \\
G_6 &= \langle \{a\#y, b\#Z, c\#Z\}, f(Z, (a\ b)\cdot Z, y, (a\ b)\cdot y) \rangle \\
G_7 &= \langle \{b\#y, a\#Y, b\#Y, a\#Z, c\#Z\}, f(y, Y, Z, Z, (a\ b)\cdot Z) \rangle \\
&\dots \text{ all positional permutations of } y, Y, Z, Z, Z \dots \\
G_{27} &= \langle \{a\#Y, b\#Y, a\#Z, c\#Z\}, f(Y, Y, (a\ b)\cdot Z, Z, Z, (a\ b)\cdot Z) \rangle
\end{aligned}$$

... all positional permutations of Y, Y, Z, Z, Z, Z ...

There are $\binom{4}{2} + \binom{5}{1}\binom{4}{1} + \binom{6}{2} = 41$ lggs. Note that G_7 is an lgg because of the freshness constraints. Without them, one could take $\sigma = \{Z \mapsto \varepsilon, Y \mapsto (Z, Z, y)\}$ to obtain $G_7\sigma = G_3$. However, the assignment $Y \mapsto (Z, Z, y)$ in σ does not respect $\{a\#Y, b\#Y\}$. \blacktriangleright

This example suggests considering more efficient variants of variadic nominal generalization. One of the reasons of inefficiency is consecutive suspensions. In [13], a rigid variant was proposed to deal with similar problems. Below we adapt that idea to our setting.

Definition 10 (\mathcal{F} -alignment). *Let w_1 and w_2 be two words (consisting of symbols of our alphabet, plus the dot ‘.’ that is used in abstraction) and \mathcal{F} be a fuzzy proximity relation on the alphabet, which is further extended to ‘.’ so that $\mathcal{F}(., h) = \mathcal{F}(h, .) = 0$ holds for any symbol h different from the dot ‘.’. Let \otimes be a T-norm.*

The triple $(h_1[i_1, j_1] \cdots h_n[i_n, j_n], \mathfrak{d}_1, \mathfrak{d}_2)$, consisting of the sequence of symbols and position pairs $h_1[i_1, j_1] \cdots h_n[i_n, j_n]$, $n \geq 0$, and two numbers $\mathfrak{d}_1, \mathfrak{d}_2 \in (0, 1]$ is called an alignment of w_1 and w_2 with respect to \mathcal{F} (or, briefly, \mathcal{F} -alignment), if

- $0 < i_1 < \cdots < i_n \leq |w_1|$, $0 < j_1 < \cdots < j_n \leq |w_2|$,
- $\mathcal{F}(h_k, w_1|_{i_k}) > 0$, $\mathcal{F}(h_k, w_2|_{j_k}) > 0$, $h_k \notin \mathbf{x} \cup \mathbf{X}$ for all $1 \leq k \leq n$, and
- $\mathfrak{d}_1 = \mathcal{F}(h_1, w_1|_{i_1}) \otimes \cdots \otimes \mathcal{F}(h_n, w_1|_{i_n})$, and $\mathfrak{d}_2 = \mathcal{F}(h_1, w_2|_{j_1}) \otimes \cdots \otimes \mathcal{F}(h_n, w_2|_{j_n})$.

Definition 11 (\mathcal{F} -rigidity function). *Given some fixed T-norm, a rigidity function with respect to \mathcal{F} (or just \mathcal{F} -rigidity function), denoted by $\mathcal{R}_{\mathcal{F}}$, is a function that computes a set of non-empty \mathcal{F} -alignments of any pair of words.*

Note that alignments computed by rigidity functions contain only non-empty sequences.

Example 12. Let \mathcal{F} be a proximity relation defined as $\mathcal{F}(f_1, f_2) = 0.6$, $\mathcal{F}(f_2, f_3) = 0.7$, $\mathcal{F}(f_1, f_3) = 0.8$, and $\mathcal{F}(g_1, g_2) = 0.7$. (It is even a similarity relation for product and Łukasiewicz T-norms, but not for the minimum T-norm.) Let $w_1 = g_2 f_1 . a g_1 x$ and $w_2 = f_2 . g_2 a x$. Below we give examples of various \mathcal{F} -rigidity functions for the product T-norm.

- $\mathcal{R}_{\mathcal{F}}(w_1, w_2)$ computes all longest \mathcal{F} -approximate common subsequences (short, \mathcal{F} -lcs):

$$\begin{aligned} & \{(f_1[2, 1].[3, 2]g_1[5, 3], 1, 0.42), (f_1[2, 1].[3, 2]g_2[5, 3], 0.7, 0.6), \\ & (f_2[2, 1].[3, 2]g_1[5, 3], 0.6, 0.7), (f_2[2, 1].[3, 2]g_2[5, 3], 0.42, 1), \\ & (f_3[2, 1].[3, 2]g_1[5, 3], 0.8, 0.49), (f_3[2, 1].[3, 2]g_2[5, 3], 0.56, 0.7), \\ & (f_1[2, 1].[3, 2] a[4, 4], 1, 0.6), (f_2[2, 1].[3, 2] a[4, 4], 0.6, 1), \\ & (f_3[2, 1].[3, 2] a[4, 4], 0.8, 0.7)\}. \end{aligned}$$

- $\mathcal{R}_{\mathcal{F}}(w_1, w_2)$ computes all \mathcal{F} -lcs that precisely (i.e., with degree 1) match w_1 :

$$\{(f_1[2, 1].[3, 2]g_1[5, 3], 1, 0.42), (f_1[2, 1].[3, 2]a[4, 4], 1, 0.6)\}.$$

- $\mathcal{R}_{\mathcal{F}}(w_1, w_2)$ computes all \mathcal{F} -lcs with the cumulative degree ≥ 0.5 , i.e., $\mathfrak{d}_1 * \mathfrak{d}_2 \geq 0.5$:

$$\{(f_1[2, 1].[3, 2]a[4, 4], 1, 0.6), (f_2[2, 1].[3, 2]a[4, 4], 0.6, 1), (f_3[2, 1].[3, 2]a[4, 4], 0.8, 0.7)\}.$$

- $\mathcal{R}_{\mathcal{F}}(w_1, w_2)$ computes all \mathcal{F} -lcs whose length is at least 4: In this case the answer is \emptyset .
- $\mathcal{R}_{\mathcal{F}}(w_1, w_2)$ computes all longest \mathcal{F} -approximate common substrings (consecutive subsequences): $\{(f_1[2, 1].[3, 2], 1, 0.6), (f_2[2, 1].[3, 2], 0.6, 1), (f_3[2, 1].[3, 2], 0.8, 0.7)\}$.

►

Definition 13 ($\mathcal{R}_{\mathcal{F}}$ -generalization). Let \mathcal{F} be a proximity relation on the alphabet, $\mathcal{R}_{\mathcal{F}}$ be an \mathcal{F} -rigidity function, and $\langle \nabla, \ell \rangle$ and $\langle \nabla, \tau \rangle$ be two A -based eics. An A -based \mathcal{F} -generalization $\langle \Gamma, u \rangle$ of $\langle \nabla, \ell \rangle$ and $\langle \nabla, \tau \rangle$ is their A -based $\mathcal{R}_{\mathcal{F}}$ -generalization if either

- $\mathcal{R}_{\mathcal{F}}(\text{head}(\ell), \text{head}(\tau)) = \emptyset$ and u is a hedge variable, or
- there exists an alignment $h_1[i_1, j_1] \cdots h_n[i_n, j_n] \in \mathcal{R}_{\mathcal{F}}(\text{head}(\ell), \text{head}(\tau))$ such that the following conditions are fulfilled:
 1. The expression u does not contain pairs of consecutive suspensions where a hedge variable is involved. That means, neighboring elements like $\pi \cdot \chi, \pi' \cdot \chi'$ where either $\chi \in \mathbf{X}$ or $\chi' \in \mathbf{X}$ are not allowed to appear in u . On the other hand, consecutive suspensions of the form $\pi_1 \cdot x_1, \dots, \pi_n \cdot x_n$, where $x_1, \dots, x_n \in \mathbf{x}$ are permitted.
 2. If all the suspensions from u are removed, then a hedge of n terms t_1, \dots, t_n remains, such that for each $1 \leq k \leq n$, the following hold:
 - $\text{head}(t_k) = h_k, \mathcal{F}(\text{head}(t_k), \text{head}(\ell|_{i_k})) > 0$ and $\mathcal{F}(\text{head}(t_k), \text{head}(\tau|_{j_k})) > 0$,
 - if t_k is an application $t_k = h_k(u_k)$, then there exists a pair of expressions ℓ_k and τ_k such that $\ell|_{i_k} = h'_k(\ell_k), \tau|_{j_k} = h''_k(\tau_k)$ and $\langle \Gamma, u_k \rangle$ is an A -based $\mathcal{R}_{\mathcal{F}}$ -generalization of $\langle \nabla, \ell_k \rangle$ and $\langle \nabla, \tau_k \rangle$,
 - if t_k is an atom $t_k = a$, then $\ell|_{i_k} = a$ and $\tau|_{j_k} = a$,
 - if t_k is an abstraction $t_k = a.u_k$, then there exists a pair of terms ℓ_k and τ_k such that $\ell|_{i_k} = b.\ell_k, \tau|_{j_k} = b'.\tau_k$ and $\langle \Gamma, u_k \rangle$ is an A -based $\mathcal{R}_{\mathcal{F}}$ -generalization of $\langle \nabla, (a b).\ell_k \rangle$ and $\langle \nabla, (a b').\tau_k \rangle$.

Example 14. Let $\mathcal{F}(f_1, f) = 0.8, \mathcal{F}(f, f_2) = 0.7$, and $\mathcal{R}_{\mathcal{F}}$ be a rigidity function that computes a set of \mathcal{F} -lcs. Assume $A = \{a, b, c\}$. The only A -based $\mathcal{R}_{\mathcal{F}}$ -lgg (modulo variable renaming) of $E_1 = \langle \emptyset, c.f_1(a, c) \rangle$ and $E_2 = \langle \emptyset, b.f_2(b, c) \rangle$ (for any T-norm) is $G = \langle \{b\#V, c\#V, a\#W, b\#W\}, b.f(V, b, W) \rangle$ with degrees 0.8 and 0.7, because

- $\mathcal{R}_{\mathcal{F}}(f_1, f_2) = \{(f[1, 1], 0.8, 0.7)\}$ and $\mathcal{R}_{\mathcal{F}}(ab, bc) = \{(b[2, 1], 1, 1)\}$ after renaming the bound atom c of E_1 to b .
- $G \preceq_{\mathcal{F}} E_1$, since $G\{V \mapsto a, W \mapsto \varepsilon\} = \langle \emptyset, b.f(a, b) \rangle$, and we have $\emptyset \subseteq \emptyset$ and $\emptyset \vdash b.f(a, b) \approx_{\mathcal{F}} c.f_1(a, c) = 0.8$,
- $G \preceq_{\mathcal{F}} E_2$, since $G\{V \mapsto \varepsilon, W \mapsto c\} = \langle \emptyset, b.f(b, c) \rangle$, and we have $\emptyset \subseteq \emptyset$ and $\emptyset \vdash b.f(b, c) \approx_{\mathcal{F}} b.f_2(b, c) = 0.7$,
- no other $\mathcal{R}_{\mathcal{F}}$ -generalization of E_1 and E_2 is strictly less general than G .

However, there exists another A -based \mathcal{F} -lgg of E_1 and E_2 which is not an $\mathcal{R}_{\mathcal{F}}$ -generalization: $G' = \langle \{c\#z\}, b.f(z, (b a)(c b) \cdot z) \rangle$ with the generalization degrees 0.8 and 0.7. Substitutions $\{z \mapsto a\}$ and $\{z \mapsto b\}$ would lead us from G' to E_1 and E_2 , respectively.

If we modified the $\mathcal{R}_{\mathcal{F}}$ function to only select those longest common subsequences where the \mathcal{F} -proximal symbols appear exactly in the same positions (i.e., each position pair has the form $[i, i]$), then $\mathcal{R}_{\mathcal{F}}(ab, bc)$ would be \emptyset and G' would be an A -based $\mathcal{R}_{\mathcal{F}}$ -lgg of E_1 and E_2 . ►

Note that the problem in Example 9 (with 41 lggs) has only one rigid lgg: $\langle \emptyset, f(X) \rangle$, if the rigidity function returns alignments of length 1, otherwise $\langle \emptyset, X \rangle$.

Given a proximity relation $\approx_{\mathcal{F}}$ and a rigidity function $\mathcal{R}_{\mathcal{F}}$, the variadic nominal generalization problem with respect to $\mathcal{R}_{\mathcal{F}}$ and $\approx_{\mathcal{F}}$ (a nominal $\mathcal{R}_{\mathcal{F}}$ -generalization problem) is formulated as follows:

Given: Two eics $E_1 = \langle \nabla, \ell \rangle$ and $E_2 = \langle \nabla, \tau \rangle$, based on a finite set of atoms A .

Find: An A -based $\mathcal{R}_{\mathcal{F}}$ -lgg of E_1 and E_2 and its generalization degrees over E_1 and E_2 .

Intuitively, the choice of A in generalization problems should be such that all the structural similarities between ℓ and τ can be captured by an A -based generalization. This is guaranteed by selecting $A = A_1 \cup A_2$ where $A_1 = \text{atoms}(\langle \nabla, \ell \rangle) \cup \text{atoms}(\langle \nabla, \tau \rangle)$ and $A_2 \subset A \setminus A_1$ is a set of k atoms where k is the minimum amount of abstraction occurrences in ℓ and in τ . The proof and more details can be found in [5].

4 The algorithm $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ for linear generalizations

We design a rule-based algorithm to solve a nominal $\mathcal{R}_{\mathcal{F}}$ -generalization problem between two given A -based eics. In this section, we consider the *linear variant* of the problem, i.e., when the generalization expressions do not contain multiple occurrences of the same variable.

A *generalization equation* (GEQ) is a triple of the form $\chi: \ell \triangleq \tau$, where χ is a variable and ℓ, τ are expressions. The rules transform *configurations*: tuples $H; T; S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2$, where

- H is a set of not-yet-solved GEQs between expressions (which can be terms);
- T is a set of not-yet-solved GEQs between terms, more precisely, between (the same) atoms or between abstractions. These problems are obtained from H by decomposition according to the rigidity function;
- S (the store) is a set of solved GEQs or GEQs between empty hedges;
- Γ is the freshness context computed so far;
- σ is the generalization substitution computed so far;
- \mathfrak{d}_1 and \mathfrak{d}_2 are the generalization degrees for each of the two input eics, respectively, computed so far.

All these sets are finite. To compute quantitative rigid generalizations of two eics $\langle \nabla, \ell \rangle$ and $\langle \nabla, \tau \rangle$, both based on the same set of atoms A , we create the *initial configuration* $\{X: \ell \triangleq \tau\}; \emptyset; \emptyset; \emptyset; id; 1; 1$, where X is a hedge variable that occurs neither in ℓ, τ nor in ∇ . Then the configuration transformation rules defined below are being applied exhaustively, in all possible ways. A configuration to which no rule applies is called a *final configuration*. The parameters \mathcal{F} , $\mathcal{R}_{\mathcal{F}}$, A , and ∇ are global and unaffected by rule applications. The T-norm is assumed to be fixed. (Recall that our T-norms have no zero-divisors.) A variable is said to be *new* if it occurs neither in the current configuration nor in a global parameter. \cup is the disjoint union operation. Given some number i , we write i^+ for $i + 1$ and i^- for $i - 1$.

Dec: Hedge Decomposition

$$\{X: \tilde{s} \triangleq \tilde{q}\} \cup H; T; S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2 \Longrightarrow \\ H' \cup H; T' \cup T; S' \cup S; \Gamma' \cup \Gamma; \sigma\{X \mapsto Z_0, r_1, Z_1, \dots, r_n, Z_n\}; \mathfrak{d}_1 \otimes \mathfrak{d}'_1; \mathfrak{d}_2 \otimes \mathfrak{d}'_2,$$

where

- $(h_1[i_1, j_1] \cdots h_n[i_n, j_n], \mathfrak{d}'_1, \mathfrak{d}'_2) \in \mathcal{R}_{\mathcal{G}}(\text{head}(\tilde{s}), \text{head}(\tilde{q}))$;
- $H' \cup T' = \{Y_1: \ell_1 \triangleq \tau_1, \dots, Y_n: \ell_n \triangleq \tau_n\}$ in which for each $1 \leq k \leq n$, the Y_k 's are new hedge variables and
 - if $h_k \in \Sigma$, then $r_k = h_k(Y_k)$, and ℓ_k and τ_k come from $\tilde{s}|_{i_k} = f_k(\ell_k)$ and $\tilde{q}|_{i_k} = g_k(\tau_k)$ for some f_k and g_k , and $Y_k: \ell_k \triangleq \tau_k \in H'$,
 - if $h_k \in \mathbf{A}$, then $r_k = Y_k$, $\ell_k = \tilde{s}|_{i_k} = \tau_k = \tilde{q}|_{i_k} = h_k$, and $Y_k: \ell_k \triangleq \tau_k \in T'$,
 - if $h_k = \cdot$ (abstraction), then $r_k = Y_k$, $\ell_k = \tilde{s}|_{i_k}$, $\tau_k = \tilde{q}|_{i_k}$, and $Y_k: \ell_k \triangleq \tau_k \in T'$;
- $S' = \{Z_0: \tilde{s}_1^{i_1^-} \triangleq \tilde{q}_1^{j_1^-}\} \cup \{Z_k: \tilde{s}|_{i_k^+}^{i_{k+1}^-} \triangleq \tilde{q}|_{j_k^+}^{j_{k+1}^-} \mid 1 \leq k \leq n-1\} \cup \{Z_n: \tilde{s}|_{i_n^+}^{|\tilde{s}|} \triangleq \tilde{q}|_{j_n^+}^{|\tilde{q}|}\}$ in which for each $0 \leq k \leq n$, the Z_k 's are new hedge variables;
- $\Gamma' = \{a\#Z \mid Z: \tilde{s}' \triangleq \tilde{q}' \in S', a \in A, \nabla \vdash a\#\tilde{s}', \nabla \vdash a\#\tilde{q}'\}$.

Tri: Trivial Atom Elimination

$$H; \{X: a \triangleq a\} \cup T; S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2 \Longrightarrow H; T; S; \Gamma; \sigma\{X \mapsto a\}; \mathfrak{d}_1; \mathfrak{d}_2.$$

Abs: Abstraction

$$H; \{X: a.t_1 \triangleq b.t_2\} \cup T; S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2 \Longrightarrow \\ \{Y: (c.a).t_1 \triangleq (c.b).t_2\} \cup H; T; S; \Gamma; \sigma\{X \mapsto c.Y\}; \mathfrak{d}_1; \mathfrak{d}_2,$$

where Y is a new hedge variable, $c \in A$, $\nabla \vdash c\#a.t_1$ and $\nabla \vdash c\#b.t_2$.

Sol: Solving

$$\{X: \tilde{s} \triangleq \tilde{q}\} \cup H; T; S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2 \Longrightarrow \\ H; T; \{X: \tilde{s} \triangleq \tilde{q}\} \cup S; \{a\#X \mid a \in A, \nabla \vdash a\#\tilde{s}, \nabla \vdash a\#\tilde{q}\} \cup \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2,$$

if $\mathcal{R}_{\mathcal{G}}(\text{head}(\tilde{s}), \text{head}(\tilde{q})) = \emptyset$.

Nar-H: Hedge Narrowing

$$\emptyset; \emptyset; \{X: (t_1, \dots, t_n) \triangleq (s_1, \dots, s_n)\} \cup S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2 \Longrightarrow \\ \emptyset; \emptyset; \{x_1: t_1 \triangleq s_1, \dots, x_n: t_n \triangleq s_n\} \cup S; \\ \Gamma\{X \mapsto x_1, \dots, x_n\} \cup \{a\#x_i \mid a \in A, \nabla \vdash a\#t_i, \nabla \vdash a\#s_i\}; \sigma\{X \mapsto x_1, \dots, x_n\}; \mathfrak{d}_1; \mathfrak{d}_2,$$

where $n \geq 0$.

Note that as a special case, the Nar-H rule removes GEQs of the form $X: \varepsilon \triangleq \varepsilon$ from S . It also replaces $X: t \triangleq s$ by $x: t \triangleq s$.

When a transformation process starting from the initial configuration $\{X: \ell \triangleq \tau\}; \emptyset; \emptyset; id; 1; 1$ leads to a final configuration $\emptyset; \emptyset; S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2$, then the computed answer is $\langle \Gamma, X\sigma \rangle$ with the corresponding generalization degrees \mathfrak{d}_1 and \mathfrak{d}_2 . We denote this algorithm by $\text{VNAU-lin}(\mathcal{R}_{\mathcal{G}})$, because it computes linear answers, i.e., every variable that appears in the expression $X\sigma$ occurs there only once. This is guaranteed by always introducing new variables for generalizations and never merging them together. The store S contains all the differences of the input expressions ℓ and τ .

Since the decomposition rule introduces branching, all resulting generalizations are collected. Some of the computed generalizations, however, may fail to be least general generalizations. For instance, for crisp \mathcal{F} , consider two eics $\langle \emptyset, f(g, h, h) \rangle$ and $\langle \emptyset, f(g, g, h) \rangle$, and let $\mathcal{R}_{\mathcal{F}}$ compute longest common subsequences. Among the answers computed by $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$, one obtains $G_1 = \langle \emptyset, f(g, X, h, Y) \rangle$ and $G_2 = \langle \emptyset, f(g, z, h) \rangle$, with $G_1 \preceq_{\mathcal{F}} G_2$. Nevertheless, we will show below that the computed set of linear generalizations is complete. This set can subsequently be minimized, which requires a matchability check. The latter problem is decidable (even for nonlinear expressions), for the reasons explained below:

- (i) the proximity class of each term is finite,
- (i) variadic matching can be reduced to finitely many fixed-arity matching problems, and
- (i) standard nominal matching is decidable.

A detailed discussion of matchability in our language lies beyond the scope of this paper.

Next, we show that $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ is terminating, sound, and complete. For termination, we first define the size of an expression and a GEQ:

$$\begin{aligned} \text{size}(a) &= \text{size}(\pi \cdot \chi) = 1, & \text{size}(a.t) &= 1 + \text{size}(t), & \text{size}(f(\tilde{s})) &= 1 + \text{size}(\tilde{s}), \\ \text{size}(r_1, \dots, r_n) &= \text{size}(r_1) + \dots + \text{size}(r_n), & \text{size}(\chi : \ell \triangleq \tau) &= \text{size}(\ell) + \text{size}(\tau). \end{aligned}$$

The complexity measure on configurations, denoted as $\mu(H; T; S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2)$, is defined as a triple (n_1, n_2, n_3) , where $n_1 = \sum_{\text{geq} \in H \cup T} \text{size}(\text{geq})$, $n_2 = \sum_{\text{geq} \in H} \text{size}(\text{geq})$, and n_3 is the number of GEQs of the form $X : (t_1, \dots, t_n) \triangleq (s_1, \dots, s_n)$, $n \geq 0$, in S .

Measures are ordered lexicographically. The ordering, denoted by $>$, is well-founded. The termination theorem below directly implies that $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ terminates on any input:

Theorem 15 (Termination of $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$). *Let \mathcal{C} be an arbitrary configuration. There are finitely many possible rule applications $\mathcal{C} \Longrightarrow \mathcal{C}_i$, $1 \leq i \leq n$ and $\mu(\mathcal{C}) > \mu(\mathcal{C}_i)$ holds for each $1 \leq i \leq n$.*

Proof. The only rule that may transform a configuration in several alternative ways is **Dec**, which uses alignments computed by the rigidity function to produce those alternatives. Rigidity functions return only finite sets, since the proximity class of each symbol is finite. Therefore, every configuration can be transformed in finitely many alternative ways.

As for the measure, the **Dec** rule does not increase its n_1 component while strictly decreasing n_2 . **Tri**, **Abs**, and **Sol** strictly decrease n_1 . **Nar-H** does not change n_1 and n_2 and strictly decreases n_3 . Hence, each rule strictly decreases the measure, leading to $\mu(\mathcal{C}) > \mu(\mathcal{C}_i)$ for each $1 \leq i \leq n$. \square

The soundness theorem shows that $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ indeed computes A -based linear $\mathcal{R}_{\mathcal{F}}$ -generalizations of the input. We use \Longrightarrow^+ to denote a sequence of one or more rule applications.

Lemma 16 (Invariant). *Given \mathcal{F} , $\mathcal{R}_{\mathcal{F}}$, a freshness context ∇ , and a configuration $H; T; S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2$ (all based on a finite set of atoms A), assume that for all $\chi : \ell \triangleq \tau \in H \cup T \cup S$, the eic $\langle \Gamma, \chi \sigma \rangle$ is an A -based linear \mathcal{F} -generalization of $\langle \nabla, \ell \rangle$ and $\langle \nabla, \tau \rangle$.*

If $H; T; S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2 \Longrightarrow H'; T'; S'; \Gamma'; \sigma'; \mathfrak{d}'_1; \mathfrak{d}'_2$ is a step by a $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ rule, then, for all $\chi : \ell \triangleq \tau \in H \cup T \cup S \cup H' \cup T' \cup S'$, the eic $\langle \Gamma', \chi \sigma' \rangle$ is an A -based linear \mathcal{F} -generalization of $\langle \nabla, \ell \rangle$ and $\langle \nabla, \tau \rangle$.

The proof of Lemma 16 uses two substitutions that can be obtained from a set M of GEQs:

$$\sigma_L(M) ::= \{\chi \mapsto \ell \mid \chi : \ell \triangleq \tau \in M\}, \quad \sigma_R(M) ::= \{\chi \mapsto \tau \mid \chi : \ell \triangleq \tau \in M\}.$$

Proof. By inspecting each rule separately, it can be verified that the invariant holds. In particular, given a transformation $H; T; S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2 \Longrightarrow H'; T'; S'; \Gamma'; \sigma'; \mathfrak{d}'_1; \mathfrak{d}'_2$ performed by any rule, we have $\nabla \vdash \ell \approx_{\mathcal{F}} \chi \sigma' \sigma_L(H' \cup T' \cup S') > 0$ and $\nabla \vdash \tau \approx_{\mathcal{F}} \chi \sigma' \sigma_R(H' \cup T' \cup S') > 0$ for all $\chi : \ell \triangleq \tau \in H \cup T \cup S$. Note that symbol proximity is guaranteed by the rigidity function. The second part, namely that $\nabla \vdash \ell \approx \chi \sigma' \sigma_L(H' \cup T' \cup S') > 0$ and $\nabla \vdash \tau \approx_{\mathcal{F}} \chi \sigma' \sigma_R(H' \cup T' \cup S') > 0$ hold for all $\chi : \ell \triangleq \tau \in (H' \cup T' \cup S') \setminus (H \cup T \cup S)$, is straight forward since σ' does not contain assignments for newly introduced generalization variables: it simply reduces to $\nabla \vdash \ell \approx \chi \sigma_L(H' \cup T' \cup S') = 1$ and $\nabla \vdash \tau \approx_{\mathcal{F}} \chi \sigma_R(H' \cup T' \cup S') = 1$. \square

Theorem 17 (Soundness of $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$). *Given \mathcal{F} , $\mathcal{R}_{\mathcal{F}}$, two expressions ℓ, τ , and a freshness context ∇ , all based on a finite set of atoms A . If $\{X : \ell \triangleq \tau\}; \emptyset; \emptyset; \emptyset; id; 1; 1 \xrightarrow{+} \emptyset; \emptyset; S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2$ is a derivation obtained by an execution of $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$, then $\langle \Gamma, X\sigma \rangle$ is an A -based linear $\mathcal{R}_{\mathcal{F}}$ -generalization of $\langle \nabla, \ell \rangle$ and $\langle \nabla, \tau \rangle$ with generalization degrees \mathfrak{d}_1 and \mathfrak{d}_2 , respectively.*

Proof. The proof uses the above invariant lemma (Lemma 16) which considers a single derivation step, i.e., given a configuration, the invariant holds after one rule application.

The first part, namely that $\langle \Gamma, X\sigma \rangle$ is an A -based linear $\mathcal{R}_{\mathcal{F}}$ -generalization of $\langle \nabla, \ell \rangle$ and $\langle \nabla, \tau \rangle$, follows from Lemma 16 by induction on the derivation length, and from the fact that the decomposition rule is guided by the rigidity function $\mathcal{R}_{\mathcal{F}}$.

The second part, that \mathfrak{d}_1 and \mathfrak{d}_2 correspond to the generalization degrees, i.e., $\langle \Gamma, X\sigma \rangle \preceq_{\mathcal{F}} \langle \nabla, \ell \rangle$ has degree \mathfrak{d}_1 and $\langle \Gamma, X\sigma \rangle \preceq_{\mathcal{F}} \langle \nabla, \tau \rangle$ has degree \mathfrak{d}_2 , is true because:

- T-norms are associative, commutative and monotone operations.
- Rigidity functions return the degrees that correspond to the combined (w.r.t. some fixed T-norm) symbol proximity used in the alignments.
- Proximity relations are reflexive, i.e., $\mathcal{F}(s, s) = 1$ for any $s \in \mathbf{A} \cup \Sigma \cup \mathbf{x} \cup \mathbf{X}$. Only decomposition, which is guided by the rigidity function, affects the degrees. \square

The completeness theorem states that, for given \mathcal{F} , $\mathcal{R}_{\mathcal{F}}$, A , two A -based eics and an arbitrary A -based linear $\mathcal{R}_{\mathcal{F}}$ -generalization G of them, $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ computes an A -based linear $\mathcal{R}_{\mathcal{F}}$ -generalization of the given eics that is at most as general as G .

Theorem 18 (Completeness of $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$). *Given \mathcal{F} , $\mathcal{R}_{\mathcal{F}}$, two expressions ℓ, τ , and a freshness context ∇ , all based on a finite set of atoms A . If G is an A -based linear $\mathcal{R}_{\mathcal{F}}$ -generalization of $\langle \nabla, \ell \rangle$ and $\langle \nabla, \tau \rangle$, then there exists a derivation $\{X : \ell \triangleq \tau\}; \emptyset; \emptyset; \emptyset; id; 1; 1 \xrightarrow{+} \emptyset; \emptyset; S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2$ obtained by an execution of $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ such that $G \preceq_{\mathcal{F}} \langle \Gamma, X\sigma \rangle$.*

Proof sketch. The idea is similar to the completeness proofs from [5, 13] and proceeds by induction over the structure of the given generalization. \square

Note that the Completeness Theorem does not mention generalization degree comparison. It is understandable, because the most general generalization $\langle \emptyset, X \rangle$ has the highest possible degrees: 1 and 1. Less general generalizations may provide more reduced degrees.

Example 19 (The importance of Nar-H). Consider a crisp proximity relation \mathcal{F} , the set of atoms $A = \{a, b, c, d\}$, and the eics $E_1 = \langle \emptyset, f(a, b) \rangle$ and $E_2 = \langle \emptyset, f(c, d) \rangle$. The rigidity function $\mathcal{R}_{\mathcal{F}}$ computes all \mathcal{F} -lcs. Without the Nar-H rule, the $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ algorithm would only apply the Dec rule once and solve (apply Sol) to reach the final configuration. It would result in $\langle \emptyset, f(X) \rangle$ as an A -based $\mathcal{R}_{\mathcal{F}}$ -generalization of E_1 and E_2 , where X generalizes the hedges (a, b) and (c, d) . With consecutive term variables, introduced by Nar-H, we get a more precise generalization $\langle \{b\#x, d\#x\}, f(x, (c\ d)(a\ b)\cdot x) \rangle$, which shows not only the fact that f is applied to two term arguments in both input expressions, but also that the second argument is obtained from the first one by swapping a and b , and c and d . Note that there are also two extra freshness constraints. \blacktriangleright

We can optimize the $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ algorithm by modifying the Dec rule in two ways:

- Forbid creating and adding to T the GEQs of the form $Y_k : h_k \triangleq h_k$, when $h_k \in \mathbf{A}$. Instead, directly replace Y_k by h_k in the substitution. It is justified by the fact that such GEQs are anyway removed by the Tri rule later and completeness is not violated. Such a modification would make the Tri rule obsolete.
- Forbid creating and adding to S the GEQs of the form $Z_k : \varepsilon \triangleq \varepsilon$ and the corresponding Z_k 's to the substitution. It is justified by the fact that such GEQs are anyway removed by the Nar-H rule later and completeness is not violated.

For simplicity, we continue to refer to the modified algorithm $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$, and use it in the rest of the paper. Now we turn to the general case, where the same variable may appear several times in the computed expression. Examples that illustrate how our algorithms work, starting with some initial configuration and transforming it until a final configuration is reached, can be found in Section 6.

5 The general case: the algorithm $\text{VNAU}(\mathcal{R}_{\mathcal{F}})$

To extend $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ to the general case, we should equip it with the capability to detect similar GEQs in the store and generalize them with the same variable. It is not a trivial task and doing it naively might lead to unsound answers. An example of such a naive merging would be an adoption of the rule from [5], which we call simple merging:

SMer: Simple Merging

$$\begin{aligned} & \emptyset; \emptyset; \{\chi_1 : \ell_1 \triangleq \tau_1, \chi_2 : \ell_2 \triangleq \tau_2\} \cup S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2 \implies \\ & \emptyset; \emptyset; \{\chi_1 : \ell_1 \triangleq \tau_1\} \cup S; \Gamma\{\chi_2 \mapsto \pi \cdot \chi_1\}; \sigma\{\chi_2 \mapsto \pi \cdot \chi_1\}; \mathfrak{d}_1 \otimes \mathfrak{d}'_1; \mathfrak{d}_2 \otimes \mathfrak{d}'_2, \\ & \pi \text{ is an } A\text{-based permutation such that } \nabla \vdash \pi \cdot \ell_1 \approx_{\mathcal{F}} \ell_2 = \mathfrak{d}'_1 > 0, \nabla \vdash \pi \cdot \tau_1 \approx_{\mathcal{F}} \tau_2 = \mathfrak{d}'_2 > 0. \end{aligned}$$

An example that shows unsoundness of this rule is the following:

Example 20. Let \mathcal{F} be the proximity relation defined as $\mathcal{F}(g_1, g_2) = 0.8$, $\mathcal{F}(g_2, g_3) = 0.7$, $\mathcal{F}(g_3, g_4) = 0.6$. Assume that the rigidity function $\mathcal{R}_{\mathcal{F}}$ computes some common subsequences (what exactly it does, is not really relevant) and consider the $\mathcal{R}_{\mathcal{F}}$ -generalization problem between $f(g_1, g_2, g_3, g_4)$ and $f(a, a, a, a)$ under the empty context. Then $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ extended with the SMer rule would give as one of the answers the eic $\langle \emptyset, f(x, x, x, x) \rangle$ with degrees 0.6 and 1, which is obviously not a correct solution, because no instance of $f(x, x, x, x)$ can be proximal to $f(g_1, g_2, g_3, g_4)$. \blacktriangleright

The problem comes from non-transitive relations and consecutive merging of proximal problems. (Consecutive merging needs transitivity.) Note that the **SMer** rule works correctly for similarity relations, since they are transitive. We make the merging rule respect non-transitivity by introducing yet another component M (for merged) to the configuration, merge several proximal problems in one single step and place the merged GEQ in M to which merging does not apply anymore:

Mer: Merging

$$\begin{aligned} & \emptyset; \emptyset; \{\chi_1 : \ell_1 \triangleq \tau_1, \dots, \chi_n : \ell_n \triangleq \tau_n\} \cup S; M; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2 \implies \\ & \emptyset; \emptyset; S; \{\chi : \ell \triangleq \tau\} \cup M; \Gamma\{\chi_1 \mapsto \pi_1 \cdot \chi, \dots, \chi_n \mapsto \pi_n \cdot \chi\}; \\ & \sigma\{\chi_1 \mapsto \pi_1 \cdot \chi, \dots, \chi_n \mapsto \pi_n \cdot \chi\}; \mathfrak{d}_1 \otimes \mathfrak{d}'_1 \otimes \dots \otimes \mathfrak{d}'_n; \mathfrak{d}_2 \otimes \mathfrak{d}''_1 \otimes \dots \otimes \mathfrak{d}''_n, \end{aligned}$$

where χ is a new variable, $n > 1$, and ℓ and τ are A -based expressions such that for all $1 \leq i \leq n$ there is an A -based permutation π_i with $\nabla \vdash \pi_i \cdot \ell \approx_{\mathcal{F}} \ell_i = \mathfrak{d}'_i > 0$ and $\nabla \vdash \pi_i \cdot \tau \approx_{\mathcal{F}} \tau_i = \mathfrak{d}''_i > 0$.

The $\text{VNAU}(\mathcal{R}_{\mathcal{F}})$ algorithm first uses $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ to reach final configurations where linear generalizations are computed. Then, it modifies each such final configuration $\emptyset; \emptyset; S; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2$ into $\emptyset; \emptyset; S; \emptyset; \Gamma; \sigma; \mathfrak{d}_1; \mathfrak{d}_2$ (i.e., initializes M with \emptyset) and applies the **Mer** rule in all possible ways as long as possible.

Computing the permutations π_1, \dots, π_n together with the expressions ℓ and τ required in the condition of **Mer** corresponds to deciding the equivariance problem. In [5], an algorithm is provided to solve this problem for the crisp fixed-arity case. It can be extended to our setting with some modifications (namely, by permitting decomposition for expressions with proximal heads and the same number of arguments, and tracking the derivation steps taken by the algorithm in order to obtain ℓ and τ), but we do not discuss the details here.

The **Mer** rule introduces two kinds of nondeterminism: (1) the choice of the set of merging GEQs, and (2) the choice of eics ℓ and τ towards which it merges all the selected GEQs. The first of them is a ‘don’t know’ nondeterminism: we should try out all possible sets of mergeable GEQs to guarantee completeness, although it might introduce redundancies: some answers can be more general than the others, or the same answer can be computed multiple times. The second choice is ‘don’t care’: it affects neither completeness nor minimality. However, it can influence the generalization degrees. For instance, if the store consists of three GEQs $y_1 : g_1 \triangleq a$, $y_2 : g_2 \triangleq a$, and $y_3 : g_3 \triangleq a$, where $\mathcal{F}(g_1, g_3) = \mathcal{F}(g_2, g_3) = 0.9$ and $\mathcal{F}(g_1, g_2) = 0.5$, and $\otimes = *$, then merging them into $y : g_1 \triangleq a$ (or into $y : g_2 \triangleq a$) will lead to degrees 0.5 and 1, while merging into $y : g_3 \triangleq a$ would give 0.81 and 1.

Example 21. Consider a modified version of the problem from Example 20, with $A = \{a, b, c, d\}$ and expressions to be generalized being $f(g_1, g_2, g_3, g_4)$ and $f(a, b, c, d)$. The proximity relation \mathcal{F} is the same as in Example 20, $\otimes = *$, and $\mathcal{R}_{\mathcal{F}}$ computes all \mathcal{F} -lcs.

Here we show only the computed answers. Full derivations can be found in Example 24 in Section 6. The $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ algorithm ends with the configuration:

$$\begin{aligned} & \emptyset; \emptyset; \{y_1 : g_1 \triangleq a, y_1 : g_2 \triangleq b, y_3 : g_3 \triangleq c, y_4 : g_4 \triangleq d\}; \\ & \{b\#y_1, c\#y_1, d\#y_1, a\#y_2, c\#y_2, d\#y_2, a\#y_3, b\#y_3, d\#y_3, a\#y_4, b\#y_4, c\#y_4\}; \\ & \{X \mapsto f(y_1, y_2, y_3, y_4), \dots\}; 1; 1. \end{aligned}$$

Further, $\text{VNAU}(\mathcal{R}_{\mathcal{F}})$ adds to it the component for M , initializes it with \emptyset , and applies **Mer** exhaustively in all possible ways. It leads to the derivation tree with seven branches, computing five different answers (two answers are re-computed on different branches). Here we show them

with their degrees, witness substitutions (computed from SUM in the final configuration), and justifications (where the notation $E\sigma = \langle \nabla, t_1 \rangle \approx_{\mathcal{F}} \langle \nabla, t_2 \rangle = \mathfrak{d}$ should be read as $E\sigma = \langle \nabla, t_1 \rangle$ and $\langle \nabla, t_1 \rangle \approx_{\mathcal{F}} \langle \nabla, t_2 \rangle = \mathfrak{d}$):

$$E_1 = \langle \{a\#y, c\#y, d\#y, a\#y_4, b\#y_4, c\#y_4\}, f((a\ b)\cdot y, y, (c\ b)\cdot y, y_4) \rangle,$$

degrees: 0.56 and 1,

$$\text{substitutions: } \sigma_L = \{y \mapsto g_2, y_4 \mapsto g_4\}, \quad \sigma_R = \{y \mapsto b, y_4 \mapsto d\},$$

$$E_1\sigma_L = \langle \emptyset, f(g_2, g_2, g_2, g_4) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(g_1, g_2, g_3, g_4) \rangle = 0.56,$$

$$E_1\sigma_R = \langle \emptyset, f(a, b, c, d) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(a, b, c, d) \rangle = 1.$$

$$E_2 = \langle \{b\#y_1, c\#y_1, d\#y_1, a\#z, b\#z, d\#z\}, f(y_1, (b\ c)\cdot z, z, (d\ c)\cdot z) \rangle,$$

degrees 0.42 and 1,

$$\text{substitutions: } \sigma_L = \{y_1 \mapsto g_1, z \mapsto g_3\}, \quad \sigma_R = \{y_1 \mapsto a, z \mapsto c\},$$

$$E_2\sigma_L = \langle \emptyset, f(g_1, g_3, g_3, g_3) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(g_1, g_2, g_3, g_4) \rangle = 0.42,$$

$$E_2\sigma_R = \langle \emptyset, f(a, b, c, d) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(a, b, c, d) \rangle = 1.$$

$$E_3 = \langle \{b\#u_1, c\#u_1, d\#u_1, a\#u_2, b\#u_2, d\#u_2\}, f(u_1, (b\ a)\cdot u_1, u_2, (d\ c)\cdot u_2) \rangle,$$

degrees 0.48 and 1,

$$\text{substitutions: } \sigma_L = \{u_1 \mapsto g_1, u_2 \mapsto g_3\}, \quad \sigma_R = \{u_1 \mapsto a, u_2 \mapsto c\},$$

$$E_3\sigma_L = \langle \emptyset, f(g_1, g_1, g_3, g_3) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(g_1, g_2, g_3, g_4) \rangle = 0.48,$$

$$E_3\sigma_R = \langle \emptyset, f(a, b, c, d) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(a, b, c, d) \rangle = 1.$$

$$E_4 = \langle \{b\#v_1, c\#v_1, d\#v_1, a\#v_2, c\#v_2, d\#v_2\}, f(v_1, v_2, (c\ a)\cdot v_1, (d\ b)\cdot v_2) \rangle,$$

degrees 0.2352 and 1,

$$\text{substitutions: } \sigma_L = \{v_1 \mapsto g_2, v_2 \mapsto g_3\}, \quad \sigma_R = \{v_1 \mapsto b, v_2 \mapsto c\},$$

$$E_4\sigma_L = \langle \emptyset, f(g_2, g_3, g_2, g_3) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(g_1, g_2, g_3, g_4) \rangle = 0.2352,$$

$$E_4\sigma_R = \langle \emptyset, f(a, b, c, d) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(a, b, c, d) \rangle = 1.$$

$$E_5 = \langle \{b\#y_1, c\#y_1, d\#y_1, a\#w, c\#w, d\#w, a\#y_4, b\#y_4, c\#y_4\}, f(y_1, w, (c\ b)\cdot w, y_4) \rangle,$$

degrees 0.7 and 1,

$$\text{substitutions: } \sigma_L = \{y_1 \mapsto g_1, w \mapsto g_2, y_4 \mapsto g_4\}, \quad \sigma_R = \{y_1 \mapsto a, w \mapsto b, y_4 \mapsto d\},$$

$$E_5\sigma_L = \langle \emptyset, f(g_1, g_2, g_2, g_4) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(g_1, g_2, g_3, g_4) \rangle = 0.7,$$

$$E_5\sigma_R = \langle \emptyset, f(a, b, c, d) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(a, b, c, d) \rangle = 1.$$

The answer E_5 is redundant since it is more general than E_1 and E_2 . If we drop it, the remaining ones form a minimal complete set $\{E_1, E_2, E_3, E_4\}$ of A -based $\mathcal{R}_{\mathcal{F}}$ -generalizations of $\langle \emptyset, f(g_1, g_2, g_3, g_4) \rangle$ and $\langle \emptyset, f(a, b, c, d) \rangle$. \blacktriangleright

Theorem 22. *The algorithm $VNAU(\mathcal{R}_{\mathcal{F}})$ terminates, is sound and complete.*

Proof. Repeated application of the **Mer** rule is terminating, since it generates a finite amount of branches and the size of the store is being strictly reduced. Soundness of **Mer** follows from the condition that ℓ, τ , and the permutations π_i are A -based, such that $\nabla \vdash \pi_i \cdot \ell \approx_{\mathcal{F}} \ell_i > 0$ and $\nabla \vdash \pi_i \cdot \tau \approx_{\mathcal{F}} \tau_i > 0$, for all selected GEQs $\chi_i: \ell_i \triangleq \tau_i$ to be merged, and T-norms do not have zero divisors. Completeness follows from the fact that **Mer** is applied in all possible ways, considering all subsets of any size > 1 from the store. \square

6 Examples

This section contains two examples that illustrate how our algorithms work.

Example 23 (Illustrating $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ and $\text{VNAU}(\mathcal{R}_{\mathcal{F}})$). Let \mathcal{F} be a proximity relation defined as $\mathcal{F}(f, f_1) = 0.8$, $\mathcal{F}(f, f_2) = 0.7$, $\mathcal{F}(h_1, h_2) = 0.9$, A be the set of atoms $\{a, b, c, d\}$, ∇ be the freshness context $\nabla = \{a\#x, b\#X\}$, \otimes be the minimum T-norm, and the rigidity function $\mathcal{R}_{\mathcal{F}}$ compute all \mathcal{F} -lcs. We apply the optimized $\text{VNAU}(\mathcal{R}_{\mathcal{F}})$ algorithm to generalize the eics $E_1 = \langle \nabla, a.u_1 \rangle$ and $E_2 = \langle \nabla, b.u_2 \rangle$, where $u_1 = f_1(g(f_1(c), X), f_2(d), (c d) \cdot X, a, h_1(a))$ and $u_2 = f_2(g(x), (c d) \cdot x, b, h_2(b))$.

Initialize $\{V : a.u_1 \triangleq b.u_2\}; \emptyset; \emptyset; \emptyset; id; 1; 1$
 \implies_{Dec} since $\text{head}(a.u_1) = \text{head}(b.u_2) = .$, and $\mathcal{R}_c F(., .) = \{([1, 1], 1, 1)\}$
 $\emptyset; \{V_1 : a.u_1 \triangleq b.u_2\}; \emptyset; \emptyset; \{V \mapsto V_1\}; 1; 1$
 \implies_{Abs} since $\nabla \vdash a\#a.u_1$, and $\nabla \vdash a\#b.u_2$
 $\{Y : u_1 \triangleq f_2(g((a b) \cdot x), (a b)(c d) \cdot x, a, h_2(a))\}; \emptyset; \emptyset; \emptyset; \{V \mapsto a.Y, \dots\}; 1; 1$
 \implies_{Dec} since $\text{head}(u_1) = f_1$, $\text{head}(u_2) = f_2$ and $\mathcal{R}_{\mathcal{F}}(f_1, f_2) = \{(f[1, 1], 0.8, 0.7)\}$
 $\{Y_1 : g(f_1(c), X), f_2(d), (c d) \cdot X, a, h_1(a) \triangleq g((a b) \cdot x), (a b)(c d) \cdot x, a, h_2(a)\};$
 $\emptyset; \emptyset; \emptyset; \{V \mapsto a.f(Y_1), \dots\}; 0.8; 0.7$
 \implies_{Dec} $\mathcal{R}_{\mathcal{F}}(g f_2 X a h_1, g x a h_2)$ contains two elements: $(g[1, 1] a[4, 3] h_1[5, 4], 1, 0.9)$
and $(g[1, 1] a[4, 3] h_2[5, 4], 0.9, 1)$. On this branch, we choose the first one:
 $\{W : (f_1(c), X) \triangleq (a b) \cdot x\}; \emptyset;$
 $\{Z : (f_2(d), (c d) \cdot X) \triangleq (a b)(c d) \cdot x\}; \{b\#Z\};$
 $\{V \mapsto a.f(g(W), Z, a, h_1(a)), \dots\}; 0.8; 0.7$ (Note that $\min(0.7, 0.9) = 0.7$)
 \implies_{Sol} since $\mathcal{R}_{\mathcal{F}}(f_1 X, x) = \emptyset$, and $\nabla \vdash b\#(f_1(c), X)$ and $\nabla \vdash b\#(a b) \cdot x$ gives $b\#W$
 $\emptyset; \emptyset; \{Z : (f_2(d), (c d) \cdot X) \triangleq (a b)(c d) \cdot x, W : (f_1(c), X) \triangleq (a b) \cdot x\};$
 $\{b\#Z, b\#W\}; \{V \mapsto a.f(g(W), Z, a, h_1(a)), \dots\}; 0.8; 0.7$

The $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ algorithm stops here with the computed linear $\mathcal{R}_{\mathcal{F}}$ -generalization $G_L = \langle \{b\#Z, b\#W\}, a.f(g(W), Z, a, h_1(a)) \rangle$, whose generalization degrees are 0.8 and 0.7. Applying the witness substitutions $\sigma_1 = \{Z \mapsto (f_2(d), (c d) \cdot X), W \mapsto (f_1(c), X)\}$ and $\sigma_2 = \{Z \mapsto (a b)(c d) \cdot x, W \mapsto (a b) \cdot x\}$ gives

$$G_L \sigma_1 = \langle \{b\#X\}, a.f(g(f_1(c), X), f_2(d), (c d) \cdot X, a, h_1(a)) \rangle, \text{ degree } 0.8,$$

$$G_L \sigma_2 = \langle \{a\#x\}, a.f(g((a b) \cdot x), (a b)(c d) \cdot x, a, h_1(a)) \rangle, \text{ degree } 0.7.$$

From $G_L \sigma_1$ and $G_L \sigma_2$, indeed, we get

$$\{b\#X\} \vdash a.f(g(f_1(c), X), f_2(d), (c d) \cdot X, a, h_1(a)) \approx_{\mathcal{F}}$$

$$a.f_1(g(f_1(c), X), f_2(d), (c d) \cdot X, a, h_1(a)) = 0.8$$

$$\{a\#x\} \vdash a.f(g((a b) \cdot x), (a b)(c d) \cdot x, a, h_1(a)) \approx_{\mathcal{F}} b.f_2(g(x), (c d) \cdot x, b, h_2(b)) = 0.7$$

To continue with $\text{VNAU}(\mathcal{R}_{\mathcal{F}})$, we create a new configuration with the empty merged set M and apply the Mer rule with substitution $\{Z \mapsto (c d) \cdot W\}$. (Note f in the merged set instead of f_1 and f_2 .)

$$\text{Initialize } \emptyset; \emptyset; \{Z : (f_2(d), (c d) \cdot X) \triangleq (a b)(c d) \cdot x, W : (f_1(c), X) \triangleq (a b) \cdot x\}; \emptyset;$$

$$\begin{aligned} & \{b\#Z, b\#W\}; \{V \mapsto a.f(g(W), Z, a, h_1(a)), \dots\}; 0.8; 0.7 \\ \Longrightarrow_{\text{Mer}} & \emptyset; \emptyset; \emptyset; \{U : (f(c), X) \triangleq (a b) \cdot x\}; \\ & \{b\#U\}; \{V \mapsto a.f(g(U), (c d) \cdot U, a, h_1(a)), \dots\}; 0.7; 0.7. \end{aligned}$$

From the computed $\mathcal{R}_{\mathcal{F}}$ -generalization $G = \langle \{b\#U\}, a.f(g(U), (c d) \cdot U, a, h_1(a)) \rangle$, we can reach the original eics E_1 and E_2 using the witness substitutions $\vartheta_1 = \{U \mapsto (f(c), X)\}$ and $\vartheta_2 = \{U \mapsto (a b) \cdot x\}$:

$$\begin{aligned} G\vartheta_1 &= \langle \{b\#X\}, a.f(g(f(c), X), f(d), (c d) \cdot X, a, h_1(a)) \rangle, \text{ degree } 0.7. \\ G\vartheta_2 &= \langle \{a\#x\}, a.f(g((a b) \cdot x), (c d)(a b) \cdot x, a, h_1(a)) \rangle, \text{ degree } 0.7. \end{aligned}$$

From $G\vartheta_1$ and $G\vartheta_2$, we get

$$\begin{aligned} \text{for } E_1: & \quad \{b\#X\} \vdash a.f(g(f(c), X), f(d), (c d) \cdot X, a, h_1(a)) \approx_{\mathcal{F}} \\ & \quad a.f_1(g(f_1(c), X), f_2(d), (c d) \cdot X, a, h_1(a))) = \min(0.8, 0.8, 0.7) = 0.7, \\ \text{for } E_2: & \quad \{a\#x\} \vdash a.f(g((a b) \cdot x), (c d)(a b) \cdot x, a, h_1(a)) \approx_{\mathcal{F}} \\ & \quad b.f_2(g(x), (c d) \cdot x, b, h_2(b)) = 0.7. \end{aligned}$$

►

Example 24 (Illustrating Mer). This is a full version of Example 21. Let \mathcal{F} be the proximity relation defined as $\mathcal{F}(g_1, g_2) = 0.8$, $\mathcal{F}(g_2, g_3) = 0.7$, $\mathcal{F}(g_3, g_4) = 0.6$. Let $\otimes = *$. Assume that the rigidity function $\mathcal{R}_{\mathcal{F}}$ computes some common subsequences (what exactly it does, is not really relevant) and consider the $\mathcal{R}_{\mathcal{F}}$ -generalization problem between $f(g_1, g_2, g_3, g_4)$ and $f(a, b, c, d)$ under the empty context. First, the VNAU-lin($\mathcal{R}_{\mathcal{F}}$) algorithm produces the following derivation:

$$\begin{aligned} & \text{Initialize } \{X : f(g_1, g_2, g_3, g_4) \triangleq f(a, b, c, d)\}; \emptyset; \emptyset; \emptyset; id; 1; 1 \\ \Longrightarrow_{\text{Dec}} & \quad \{Y_1 : g_1 \triangleq a, Y_1 : g_2 \triangleq b, Y_3 : g_3 \triangleq c, Y_4 : g_4 \triangleq d\}; \emptyset; \emptyset; \emptyset; \\ & \quad \{X \mapsto f(Y_1, Y_2, Y_3, Y_4)\}; 1; 1 \\ \Longrightarrow_{\text{Sol}}^4 & \quad \emptyset; \emptyset; \{Y_1 : g_1 \triangleq a, Y_1 : g_2 \triangleq b, Y_3 : g_3 \triangleq c, Y_4 : g_4 \triangleq d\}; \\ & \quad \{b\#Y_1, c\#Y_1, d\#Y_1, a\#Y_2, c\#Y_2, d\#Y_2, a\#Y_3, b\#Y_3, d\#Y_3, a\#Y_4, b\#Y_4, c\#Y_4\}; \\ & \quad \{X \mapsto f(Y_1, Y_2, Y_3, Y_4), \dots\}; 1; 1 \\ \Longrightarrow_{\text{Nar-H}}^4 & \quad \emptyset; \emptyset; \{y_1 : g_1 \triangleq a, y_1 : g_2 \triangleq b, y_3 : g_3 \triangleq c, y_4 : g_4 \triangleq d\}; \\ & \quad \{b\#y_1, c\#y_1, d\#y_1, a\#y_2, c\#y_2, d\#y_2, a\#y_3, b\#y_3, d\#y_3, a\#y_4, b\#y_4, c\#y_4\}; \\ & \quad \{X \mapsto f(y_1, y_2, y_3, y_4), \dots\}; 1; 1. \end{aligned}$$

Now, VNAU($\mathcal{R}_{\mathcal{F}}$) adds to the final configuration obtained the component for M and initializes it with \emptyset :

$$\begin{aligned} & \emptyset; \emptyset; \{y_1 : g_1 \triangleq a, y_2 : g_2 \triangleq b, y_3 : g_3 \triangleq c, y_4 : g_4 \triangleq d\}; \emptyset; \\ & \quad \{b\#y_1, c\#y_1, d\#y_1, a\#y_2, c\#y_2, d\#y_2, a\#y_3, b\#y_3, d\#y_3, a\#y_4, b\#y_4, c\#y_4\}; \\ & \quad \{X \mapsto f(y_1, y_2, y_3, y_4), \dots\}; 1; 1 \end{aligned}$$

Then it applies Mer exhaustively in all possible ways. The obtained derivation tree has the following branches:

Branch 1: Merging the GEQs for y_1 , y_2 , and y_3 , using the substitution

- $\{y_1 \mapsto (a b) \cdot y, y_2 \mapsto y, y_3 \mapsto (c b) \cdot y\}$ results in
 $\implies_{\text{Mer}} \emptyset; \emptyset; \{y_4 : g_4 \triangleq d\}; \{y : g_2 \triangleq b\}; \{a\#y, c\#y, d\#y, a\#y_4, b\#y_4, c\#y_4\};$
 $\{X \mapsto f((a b) \cdot y, y, (c b) \cdot y, y_4), \dots\}; 0.56; 1$
- Branch 2: Merging the GEQs for $y_2, y_3,$ and $y_4,$ using the substitution
 $\{y_2 \mapsto (b c) \cdot z, y_3 \mapsto z, y_4 \mapsto (d c) \cdot z\}$ results in
 $\implies_{\text{Mer}} \emptyset; \emptyset; \{y_1 : g_1 \triangleq a\}; \{z : g_3 \triangleq c\}; \{b\#y_1, c\#y_1, d\#y_1, a\#z, b\#z, d\#z\};$
 $\{X \mapsto f(y_1, (b c) \cdot z, z, (d c) \cdot z), \dots\}; 0.42; 1.$
- Branch 3: First, merging the GEQs for y_1 and $y_2,$ using the substitution
 $\{y_1 \mapsto u_1, y_2 \mapsto (b a) \cdot u_1\}$ leads to
 $\implies_{\text{Mer}} \emptyset; \emptyset; \{y_3 : g_3 \triangleq c, y_4 : g_4 \triangleq d\}; \{u_1 : g_1 \triangleq a\};$
 $\{b\#u_1, c\#u_1, d\#u_1, a\#y_3, b\#y_3, d\#y_3, a\#y_4, b\#y_4, c\#y_4\};$
 $\{X \mapsto f(u_1, (b a) \cdot u_1, y_3, y_4), \dots\}; 0.8; 1$
 the merging substitution $\{y_1 \mapsto (b a) \cdot u_1, y_2 \mapsto u_1\}.$
- Second, merging the GEQs for y_3 and $y_4,$ using the substitution
 $\{y_3 \mapsto u_2, y_4 \mapsto (d c) \cdot u_2\}$ results in
 $\implies_{\text{Mer}} \emptyset; \emptyset; \emptyset; \{u_1 : g_1 \triangleq a, u_2 : g_3 \triangleq c\}; \{b\#u_1, c\#u_1, d\#u_1, a\#u_2, b\#u_2, d\#u_2\};$
 $\{X \mapsto f(u_1, (b a) \cdot u_1, u_2, (d c) \cdot u_2), \dots\}; 0.48; 1.$
- (Alternatively, we could have chosen to merge towards the GEQ for y_2
 in the first step or towards the GEQ for y_4 in the second step, but the
 obtained generalization would be equi-general to the one above.
 Moreover, these generalizations would even have the same degree.)
- Branch 4: First, merging the GEQs for y_1 and $y_3,$ using the substitution
 $\{y_1 \mapsto v_1, y_3 \mapsto (c a) \cdot v_1\}$ leads to
 $\implies_{\text{Mer}} \emptyset; \emptyset; \{y_2 : g_2 \triangleq b, y_4 : g_4 \triangleq d\}; \{v_1 : g_2 \triangleq a\};$
 $\{b\#v_1, c\#v_1, d\#v_1, a\#y_2, c\#y_2, d\#y_2, a\#y_4, b\#y_4, c\#y_4\};$
 $\{X \mapsto f(v_1, y_2, (c a) \cdot v_1, y_4), \dots\}; 0.56; 1$
- Second, merging the GEQs for y_2 and $y_4,$ using the substitution
 $\{y_2 \mapsto v_2, y_4 \mapsto (d b) \cdot v_2\}$ results in
 $\implies_{\text{Mer}} \emptyset; \emptyset; \emptyset; \{v_1 : g_2 \triangleq a, v_2 : g_3 \triangleq b\}; \{b\#v_1, c\#v_1, d\#v_1, a\#v_2, c\#v_2, d\#v_2\};$
 $\{X \mapsto f(v_1, v_2 (c a) \cdot v_1, (d b) \cdot v_2), \dots\}; 0.2352; 1.$
- Branch 5: Merging the GEQs for $y_2,$ and $y_3,$ using the substitution
 $\{y_2 \mapsto w, y_3 \mapsto (c b) \cdot w\}$ results in

$$\begin{aligned} \Longrightarrow_{\text{Mer}} \quad & \emptyset; \emptyset; \{y_1 : g_1 \triangleq a, y_4 : g_4 \triangleq d\}; \\ & \{w : g_2 \triangleq b\}; \{b\#y_1, c\#y_1, d\#y_1, a\#w, c\#w, d\#w, a\#y_4, b\#y_4, c\#y_4\}; \\ & \{X \mapsto f(y_1, w, (c\,b)\cdot w, y_4), \dots\}; 0.7; 1. \end{aligned}$$

Note that the computed answer here is redundant, because it is more general than those computed in branches 1 and 2.

The same would happen if we merged towards the GEQ for y_3 .

Branch 6: In this branch, the algorithm computes the same answer as in branch 3, but in a different order: first, merging the GEQs for y_3 and y_4 , and then for y_1 and y_2 .

Branch 7: In this branch, the algorithm computes the same answer as in branch 4, but in a different order: first, merging the GEQs for y_2 and y_4 , and then for y_1 and y_3 .

Hence, we computed five different answers (two of them even twice). Here we show them with their degrees, witness substitutions (computed from $S \cup M$ in the final configuration), and justifications (where the notation $E\sigma = \langle \nabla, t_1 \rangle \approx_{\mathcal{F}} \langle \nabla, t_2 \rangle = \mathfrak{d}$ should be read as $E\sigma = \langle \nabla, t_1 \rangle$ and $\langle \nabla, t_1 \rangle \approx_{\mathcal{F}} \langle \nabla, t_2 \rangle = \mathfrak{d}$):

$$E_1 = \langle \{a\#y, c\#y, d\#y, a\#y_4, b\#y_4, c\#y_4\}, f((a\,b)\cdot y, y, (c\,b)\cdot y, y_4) \rangle,$$

degrees: 0.56 and 1,

$$\text{substitutions: } \sigma_L = \{y \mapsto g_2, y_4 \mapsto g_4\}, \quad \sigma_R = \{y \mapsto b, y_4 \mapsto d\},$$

$$E_1\sigma_L = \langle \emptyset, f(g_2, g_2, g_2, g_4) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(g_1, g_2, g_3, g_4) \rangle = 0.56,$$

$$E_1\sigma_R = \langle \emptyset, f(a, b, c, d) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(a, b, c, d) \rangle = 1.$$

$$E_2 = \langle \{b\#y_1, c\#y_1, d\#y_1, a\#z, b\#z, d\#z\}, f(y_1, (b\,c)\cdot z, z, (d\,c)\cdot z) \rangle,$$

degrees 0.42 and 1,

$$\text{substitutions: } \sigma_L = \{y_1 \mapsto g_1, z \mapsto g_3\}, \quad \sigma_R = \{y_1 \mapsto a, z \mapsto c\},$$

$$E_2\sigma_L = \langle \emptyset, f(g_1, g_3, g_3, g_3) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(g_1, g_2, g_3, g_4) \rangle = 0.42,$$

$$E_2\sigma_R = \langle \emptyset, f(a, b, c, d) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(a, b, c, d) \rangle = 1.$$

$$E_3 = \langle \{b\#u_1, c\#u_1, d\#u_1, a\#u_2, b\#u_2, d\#u_2\}, f(u_1, (b\,a)\cdot u_1, u_2, (d\,c)\cdot u_2) \rangle,$$

degrees 0.48 and 1,

$$\text{substitutions: } \sigma_L = \{u_1 \mapsto g_1, u_2 \mapsto g_3\}, \quad \sigma_R = \{u_1 \mapsto a, u_2 \mapsto c\},$$

$$E_3\sigma_L = \langle \emptyset, f(g_1, g_1, g_3, g_3) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(g_1, g_2, g_3, g_4) \rangle = 0.48,$$

$$E_3\sigma_R = \langle \emptyset, f(a, b, c, d) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(a, b, c, d) \rangle = 1.$$

$$E_4 = \langle \{b\#v_1, c\#v_1, d\#v_1, a\#v_2, c\#v_2, d\#v_2\}, f(v_1, v_2, (c\,a)\cdot v_1, (d\,b)\cdot v_2) \rangle,$$

degrees 0.2352 and 1,

$$\text{substitutions: } \sigma_L = \{v_1 \mapsto g_2, v_2 \mapsto g_3\}, \quad \sigma_R = \{v_1 \mapsto b, v_2 \mapsto c\},$$

$$E_4\sigma_L = \langle \emptyset, f(g_2, g_3, g_2, g_3) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(g_1, g_2, g_3, g_4) \rangle = 0.2352,$$

$$E_4\sigma_R = \langle \emptyset, f(a, b, c, d) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(a, b, c, d) \rangle = 1.$$

$$\begin{aligned}
E_5 &= \langle \{b\#y_1, c\#y_1, d\#y_1, a\#w, c\#w, d\#w, a\#y_4, b\#y_4, c\#y_4\}, f(y_1, w, (c\ b)\cdot w, y_4) \rangle, \\
&\text{degrees } 0.7 \text{ and } 1, \\
&\text{substitutions: } \sigma_L = \{y_1 \mapsto g_1, w \mapsto g_2, y_4 \mapsto g_4\}, \quad \sigma_R = \{y_1 \mapsto a, w \mapsto b, y_4 \mapsto d\}, \\
E_5\sigma_L &= \langle \emptyset, f(g_1, g_2, g_2, g_4) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(g_1, g_2, g_3, g_4) \rangle = 0.7, \\
E_5\sigma_R &= \langle \emptyset, f(a, b, c, d) \rangle \approx_{\mathcal{F}} \langle \emptyset, f(a, b, c, d) \rangle = 1.
\end{aligned}$$

However, if we drop E_5 , which is redundant since it is more general than E_1 and E_2 , the remaining ones form a minimal complete set $\{E_1, E_2, E_3, E_4\}$ of A -based $\mathcal{R}_{\mathcal{F}}$ -generalizations of $\langle \emptyset, f(g_1, g_2, g_3, g_4) \rangle$ and $\langle \emptyset, f(a, b, c, d) \rangle$. \blacktriangleright

7 Final remarks

Our approach relies on the expressive capabilities of nominal expressions, variadic syntax, and fuzzy relations. It subsumes several existing generalization algorithms and extends both nominal and (rigid) variadic algorithms by integrating fuzzy relations. On the other hand, it allows for richer structures in quantitative generalization problems. Other dimensions of comparison include the use of proximity versus similarity relations, as well as linear versus unrestricted variants.

The motivation behind considering the rigid variant was to address practical challenges such as computational overhead and limited control over generalization precision. This refinement enhances efficiency and offers greater flexibility, enabling the algorithm to be tuned for specific applications. In particular, rigidity functions support hybrid approaches that combine the speed of string-based similarity techniques with the structural precision of tree-based methods. Their design also facilitates the integration of approximate and learnable similarity measures, allowing for the incorporation of AI-driven techniques.

Linear generalizations focus on detecting common structure, whereas the general case also accounts for “similar differences.” In practical applications, particularly those related to program analysis, linear generalizations are often sufficient. The length of each derivation performed by the $\text{VNAU-lin}(\mathcal{R}_{\mathcal{F}})$ algorithm is bounded by the size (more precisely, by the depth) of the problem. Moreover, when the rigidity function returns a singleton set, the derivation contains only a single branch.

When the given fuzzy relation is a similarity, the alignments computed by rigidity functions contain only symbols taken from the given hedges to be generalized; the only remaining choice is from which hedge each symbol is selected. In contrast, for proximity relations, such symbols need not belong to the given hedges: they may also arise as common neighbors of symbols appearing in those hedges.

Merging under similarity relations can be done consecutively and, therefore, the SMer rule can be used, which does not cause branching. For proximity, we have to merge several GEQs at once as done in the Mer rule and need to consider all possible alternatives.

Various versions of crisp generalization algorithms have found applications in several areas of software science, e.g., in automated program repair [3, 17, 23, 26] or clone detection [8, 9, 25]. Incorporating quantitative features, such as proximity or similarity relations, can further extend the applicability of these methods by enabling the capture of approximately common code fragments that would remain undetected by crisp generalization techniques.

Acknowledgments. This work was partially supported by the Austrian Science Fund (FWF), project P 35530.

References

- [1] Hassan Aït-Kaci and Gabriella Pasi. Fuzzy lattice operations on first-order terms over signatures with similar constructors: A constraint-based approach. *Fuzzy Sets Syst.*, 391:1–46, 2020. doi:[10.1016/J.FSS.2019.03.019](https://doi.org/10.1016/J.FSS.2019.03.019).
- [2] Mauricio Ayala-Rincón, Washington de Carvalho Segundo, Maribel Fernández, Daniele Nantes-Sobrinho, and Ana Cristina Rocha Oliveira. A formalisation of nominal α -equivalence with A, C, and AC function symbols. *Theor. Comput. Sci.*, 781:3–23, 2019. doi:[10.1016/j.tcs.2019.02.020](https://doi.org/10.1016/j.tcs.2019.02.020).
- [3] Johannes Bader, Andrew Scott, Michael Pradel, and Satish Chandra. Getafix: learning to fix bugs automatically. *Proc. ACM Program. Lang.*, 3(OOPSLA):159:1–159:27, 2019. doi:[10.1145/3360585](https://doi.org/10.1145/3360585).
- [4] Alexander Baumgartner and Temur Kutsia. Generalization of variadic structures with binders: A tool for structural code comparison, 2025. URL: <https://arxiv.org/abs/2509.25023>, arXiv:2509.25023.
- [5] Alexander Baumgartner, Temur Kutsia, Jordi Levy, and Mateu Villaret. Nominal anti-unification. In Maribel Fernández, editor, *26th International Conference on Rewriting Techniques and Applications, RTA 2015, Warsaw, Poland, June 29 - July 1, 2015*, volume 36 of *LIPICs*, pages 57–73. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015. doi:[10.4230/LIPICs.RTA.2015.57](https://doi.org/10.4230/LIPICs.RTA.2015.57).
- [6] Alexander Baumgartner, Temur Kutsia, Jordi Levy, and Mateu Villaret. Higher-order pattern anti-unification in linear time. *J. Autom. Reasoning*, 58(2):293–310, 2017. doi:[10.1007/s10817-016-9383-3](https://doi.org/10.1007/s10817-016-9383-3).
- [7] Alexander Baumgartner and Daniele Nantes-Sobrinho. Nominal anti-unification modulo equational theories. *J. Log. Algebraic Methods Program.*, 149:101100, 2026. doi:[10.1016/J.JLAMP.2025.101100](https://doi.org/10.1016/J.JLAMP.2025.101100).
- [8] Peter E. Bulychev, Egor V. Kostylev, and Vladimir A. Zakharov. Anti-unification algorithms and their applications in program analysis. In Amir Pnueli, Irina B. Virbitskaite, and Andrei Voronkov, editors, *Perspectives of Systems Informatics, 7th International Andrei Ershov Memorial Conference, PSI 2009, Novosibirsk, Russia, June 15-19, 2009. Revised Papers*, volume 5947 of *Lecture Notes in Computer Science*, pages 413–423. Springer, 2009. doi:[10.1007/978-3-642-11486-1_35](https://doi.org/10.1007/978-3-642-11486-1_35).
- [9] Petr Bulychev and Marius Minea. An evaluation of duplicate code detection using anti-unification. In *Proc. 3rd International Workshop on Software Clones*, 2009.
- [10] David M. Cerna and Temur Kutsia. Anti-unification and generalization: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI '23*, 2023. doi:[10.24963/ijcai.2023/736](https://doi.org/10.24963/ijcai.2023/736).
- [11] Besik Dundua, Temur Kutsia, and Mikheil Rukhaia. Unranked nominal unification. In Aybüke Özgün and Yulia Zinova, editors, *Language, Logic, and Computation - 13th International Tbilisi Symposium, TbiLLC 2019, Batumi, Georgia, September 16-20, 2019, Revised Selected Papers*, volume 13206 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2019. doi:[10.1007/978-3-030-98479-3_14](https://doi.org/10.1007/978-3-030-98479-3_14).
- [12] Murdoch J. Gabbay and Aad Mathijssen. One-and-a-halfth-order logic. In *Proceedings of the 8th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming, PPDP '06*, page 189–200, New York, NY, USA, 2006. Association for Computing Machinery. doi:[10.1145/1140335.1140359](https://doi.org/10.1145/1140335.1140359).
- [13] Temur Kutsia, Jordi Levy, and Mateu Villaret. Anti-unification for unranked terms and hedges. *J. Autom. Reason.*, 52(2):155–190, 2014. doi:[10.1007/S10817-013-9285-6](https://doi.org/10.1007/S10817-013-9285-6).
- [14] Temur Kutsia and Cleo Pau. Matching and generalization modulo proximity and tolerance relations. In Aybüke Özgün and Yulia Zinova, editors, *Language, Logic, and Computation - 13th International Tbilisi Symposium, TbiLLC 2019, Batumi, Georgia, September 16-20, 2019, Revised Selected Papers*, volume 13206 of *Lecture Notes in Computer Science*, pages 323–342. Springer, 2019. doi:[10.1007/978-3-030-98479-3_16](https://doi.org/10.1007/978-3-030-98479-3_16).

- [15] Temur Kutsia and Cleo Pau. A framework for approximate generalization in quantitative theories. In Jasmin Blanchette, Laura Kovács, and Dirk Pattinson, editors, *Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8-10, 2022, Proceedings*, volume 13385 of *Lecture Notes in Computer Science*, pages 578–596. Springer, 2022. doi:10.1007/978-3-031-10769-6_34.
- [16] Jordi Levy and Mateu Villaret. Nominal unification from a higher-order perspective. *ACM Trans. Comput. Log.*, 13(2):10:1–10:31, 2012. doi:10.1145/2159531.2159532.
- [17] Sonu Mehta, Ranjita Bhagwan, Rahul Kumar, Chetan Bansal, Chandra Shekhar Maddila, Balasubramanyan Ashok, Sumit Asthana, Christian Bird, and Aditya Kumar. Rex: Preventing bugs and misconfiguration in large services using correlated change analysis. In Ranjita Bhagwan and George Porter, editors, *17th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2020, Santa Clara, CA, USA, February 25-27, 2020*, pages 435–448. USENIX Association, 2020. URL: <https://www.usenix.org/conference/nsdi20/presentation/mehta>.
- [18] Dale Miller. A logic programming language with lambda-abstraction, function variables, and simple unification. *J. Log. Comput.*, 1(4):497–536, 1991. doi:10.1093/logcom/1.4.497.
- [19] Cleo Pau. *Symbolic Techniques for Approximate Reasoning*. PhD thesis, RISC, Johannes Kepler University, 2022.
- [20] Andrew M. Pitts. Nominal logic, a first order theory of names and binding. *Inf. Comput.*, 186(2):165–193, 2003. doi:10.1016/S0890-5401(03)00138-X.
- [21] Gordon D. Plotkin. A note on inductive generalization. *Machine Intell.*, 5(1):153–163, 1970.
- [22] John C. Reynolds. Transformational systems and the algebraic structure of atomic formulas. *Machine Intell.*, 5(1):135–151, 1970.
- [23] Reudismam Rolim de Sousa, Gustavo Soares, Rohit Gheyi, Titus Barik, and Loris D’Antoni. Learning quick fixes from code repositories. In Cristiano D. Vasconcellos, Karina Girardi Roggia, Vanessa Collere, and Paulo Bousfield, editors, *35th Brazilian Symposium on Software Engineering, SBES 2021, Joinville, Santa Catarina, Brazil, 27 September 2021 - 1 October 2021*, pages 74–83. ACM, 2021. doi:10.1145/3474624.3474650.
- [24] Manfred Schmidt-Schauß and Daniele Nantes-Sobrinho. Nominal anti-unification with atom-variables. In Amy P. Felty, editor, *7th International Conference on Formal Structures for Computation and Deduction, FSCD 2022, August 2-5, 2022, Haifa, Israel*, volume 228 of *LIPICs*, pages 7:1–7:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.FSCD.2022.7.
- [25] Wim Vanhoof and Gonzague Yernaux. Generalization-driven semantic clone detection in CLP. In Maurizio Gabbrielli, editor, *Logic-Based Program Synthesis and Transformation - 29th International Symposium, LOPSTR 2019, Porto, Portugal, October 8-10, 2019, Revised Selected Papers*, volume 12042 of *Lecture Notes in Computer Science*, pages 228–242. Springer, 2019. doi:10.1007/978-3-030-45260-5_14.
- [26] Emily Rowan Winter, Vesna Nowack, David Bowes, Steve Counsell, Tracy Hall, Sæmundur Óskar Haraldsson, John R. Woodward, Serkan Kirbas, Etienne Windels, Olayori McBello, Abdurahman Atakishiyev, Kevin Kells, and Matthew W. Pagano. Towards developer-centered automatic program repair: findings from bloomberg. In Abhik Roychoudhury, Cristian Cadar, and Miryung Kim, editors, *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2022, Singapore, Singapore, November 14-18, 2022*, pages 1578–1588. ACM, 2022. doi:10.1145/3540250.3558953.