Fuzzy Lattice–Theoretic Operations over Data and Knowledge Structures

Hassan Aït-Kaci and Gabriella Pasi

January 8, 2019

DRAFT—PLEASE DO NOT DISTRIBUTE

What is this about?

This explores the lattice-theoretic properties of the fuzzy processing of data and knowledge structures, such as First-Order Terms ($\mathcal{FOT}$s) and Order-Sorted Feature ($\mathcal{OSF}$) graphs. These objects are ordered with endomorphic structure subsumption which characterizes consistent instantiation of subterms, and more generally inheritance of features from a sort to its subsorts. Constraint systems consisting of declarative rules and axioms formalizing these operations as syntax-driven constraint normalization also provide effective operational interpretations. Extending these notions to assimilate distinct but similar objects can then be done by loosening semantic congruence among constructors. This is achieved by fuzzifying the constraint system. For example, fuzzy $\mathcal{FOT}$ unification is used in Fuzzy Logic Programming ($\mathcal{FLP}$), thus adding flexibility and expressivity to operations on $\mathcal{FOT}$s. This kind of fuzzification can be defined and used as well for approximate data and knowledge representation and processing thanks to richer attributed object and concept structures such as $\mathcal{OSF}$ graphs.

Why Lattice Theory?

Because it is the mathematics of consistent approximation. Indeed, when (fuzzy) approximation of $\mathcal{FOT}$ terms or $\mathcal{OSF}$ graphs is defined as (fuzzy) structural subsumption, (fuzzy) unification is the Greatest Lower Bound (**glb**) operation, and (fuzzy) generalization is the Least Upper Bound (**lub**) operation. This provides a (fuzzy) **glb** operation over $\mathcal{FOT}$s and $\mathcal{OSF}$ object structures to act as, *e.g.*, a kind of "*(fuzzy) object join*" to specify approximate retrieval patterns over an object database. Dually, the **lub** operation is the computation of the most specific $\mathcal{FOT}$ or $\mathcal{OSF}$ graph up to a fuzzy threshold that is their most specific approximate generalization for that truth threshold. Such could be used, *e.g.*, for fuzzy object schema inference or Machine Learning by fuzzy inductive reasoning. Therefore, understanding the formal operational aspects of (fuzzy) structure unification and its dual (fuzzy) structure generalization are invaluable pursuits most suitably formalized using Lattice Theory.

Our objective?

It is to extend to fuzzy operations (to "*fuzzify*") both lattice operations on $\mathcal{FOT}$s and $\mathcal{OSF}$ graphs. Calibrating such structures with fuzzy truth levels as approximation degrees can then exploit more expressive lattice-theoretic operations (fuzzy unification, but also fuzzy generalization). Our pragmatic motivation is that such fuzzy lattice operations on $\mathcal{FOT}$s and $\mathcal{OSF}$ graphs are very convenient in structured data and knowledge representation and processing, such as *approximate* Information Mining and Retrieval.

Still too fuzzy?

Be that as it may, we—*the authors*—hope that you—*reading this*—will be prompted to muse further into this document's contents to understand what the foregoing techno-gibberish actually means. Who knows? You may just share the frizzy fuzzy fun we felt defrizzing tangled fuzz, and perhaps even be enticed to use, or extend, our results.

Keywords: Approximate Information Processing; Lattices; First-Order Terms; Feature Terms; Fuzzy Unification; Fuzzy Generalization; Fuzzy Pattern-directed Reasoning; Fuzzy Pattern Abduction; Fuzzy Pattern Induction; Fuzzy Machine Learning.

Version of January 8, 2019

Preamble

Our model of the world is, at best, fuzzy. What we hold to be true or false, as far as reasoning is concerned, is a partial approximation of ideal concepts and relations among such. Yet, far from suffering from this seeming imperfection, we are actually quite clever at many tasks involving reasoning (including recognition and learning) precisely *because* we can efficiently make sense out of approximate knowledge. It is therefore natural to wish that all AI tools be given the capability of approximate reasoning as a practical means to make efficient pragmatic sense out of the abundance of knowledge fed by the current supernova-size explosion of data—whether extracting knowledge learning from it or using what is learned to render intelligent services for useful aims; or both.

The ideas reported in this document are some mathematical and computational reflections on approximate reasoning with data and knowledge represented as algebraic trees and, more generally, labeled rooted graphs (*i.e.*, most models). For us, approximation is interpreted as a partial order on object structures composed of symbols, where some symbols may denote more or less similar concepts. The ideas we have undertaken to develop here came to be out of the visit of the first author to Milan, Italy, in the Fall of 2016, at the invitation of the second author. It is the consequence of a congruence of minds intrigued at fuzzifying the power of lattice theory for data and knowledge processing. It is the result of our discussions trying to give simple answers to simple questions. The initial question was simple indeed: "*What happens to the Reynolds-Plotkin lattice of $\mathcal{FOT}$s with fuzzy unification and generalization?*" But since this lattice is itself only a special case of the lattice of rooted order-sorted feature graphs: "*What happens then when we fuzzify $\mathcal{OSF}$ lattice operations?*" We were demanding, however, as we strove to give intuitive, formal, and operational, answers to these questions. This has led, after some methodic and laborious research and a few initial but unsatisfying answers, to this present collection of technical thoughts. These constitute, in our opinion, just *a start*.

We wish to share this, should there be anyone interested in the same or related topics. We hope that the reader will join us, even if only partially, in the satisfaction of seeing exposed a comprehensive, but simple and coherent, family of algebraic structures for fuzzy deduction and induction. It is our further wish that these ideas beget new ones in the reader's mind since, as we try to illustrate in closing, we believe that there is a high potential for further work and applications.

We also took up the challenge of making this work destined to a wide audience, yet be self-contained. All such notions that we use and/or build upon are recalled, the needed background and vocabulary, summed-up in an appendix, where we cover all essential notions needed to understand the rest of this book. In a second appendix, we also elaborate a few detailed examples. Although this work may first appear as just one more theoretical niche for idle mathematicians, it has in fact a universal pragmatic purpose. A much longer but more accurate title should have

perhaps been worded as, " ***Lattice-Theoretic Operations for Fuzzy Inference by Deduction and Induction over Similar Data and Knowledge Structures***." Indeed, it leads to a universal model for efficiently implementing a powerful fuzzy reasoning algebra over approximate subsumption-ordered object and concept structures.

Acknowledgments

Each author is indebted to the other for being foolish enough to discuss the issues in sufficient detail as to be formally convincing. It became all the more intriguing when we realized that this may have further consequences over variant structures and operations.

And of course, each author is more indebted to their respective families as they each were making even lesser sense than usual while rambling about crazy fuzzy lattices when working in the garden tending sprouting ivy crawling up and down a criss-crossing mesh of guiding trellis.[1]

It is hoped that our ideas be the seeds of several creative sprawling upshots.

Version of: January 8, 2019

[1] Like trying to parse, let alone make sense out of, this very sentence!

Table of Contents

List of Figures

List of Examples

Chapter 1

Version of January 8, 2019

Generalities

Authors' comment: ***N.B.****: Please note that several parts of this document are in a state of draft which is of course* ***not to be distributed****. The parts that have been published are indicated and duly referenced. The parts still in a draft state are essentially sets of notes and are meant to evolve into finished form in the near future. Hence, much of their current contents is likely to be modified, or could be synthesized more succintly, or may diseappear altogether in a later update. Such parts are incomplete and/or possibly inconsistent in their current form. New parts may appear later.*

Our expected readership is assumed to be at ease with, or at least not averse to, advanced senior-level or graduate-level Symbolic Logic and Algebra;[1] more specifically, basic Lattice Theory.[2] Although not required, familiarity with some software specification and/or programming languages should be a plus. In particular, one should be comfortable with the syntax, data structures, and operations of Logic Programming (*e.g.*, Prolog).[3] Also, one should not mind our making use of formal notation as we find it helpful in conveying accurately what we mean. However, we strive to keep this notation simple and intuitive, and also "easy to program." Indeed, our *leit motiv* is deriving implementable methods from declarative specifications.

What follows in this introduction explains our motivation, overviews the ideas we discuss, and gives a snapshot of the rest of the document's organization.

1.1 Motivation

In this work, we do not mean to deal with fuzzy concepts or fuzzy properties thereof. Such would require to encode existing knowledge and data bases to be populated with fuzzy entities. This would not only be a formidable task to undertake but also an unrealistic assumption. Rather, we wish to take the world of knowledge and data as it exists with possibly some additional information relating various concept or data constructor symbols. This information consists in a fuzzy measure of approximation among the meaning of symbols. For example, in a knowledge

[1] https://www.encyclopediaofmath.org/index.php/Algebra_of_logic
[2] https://en.wikipedia.org/wiki/Lattice_(order)
[3] https://en.wikipedia.org/wiki/Prolog

base dealing with people, an approximate pattern for fuzzy pattern information retrieval could tolerate considering that the symbols "*person*" and "*individual*" denote the same concepts with a .9 approximation degree.[4] As well, relating some attributes of so similar concepts could identify, at a given approximation degree, which attributes of a concept correspond to what attributes of a similar concept.

Authors' comment:

We need some general comparative discussion about structurally vs. numerically assessed approximation—viz., First-Order Term ($\mathcal{FOT}$) or Order-Sorted Feature ($\mathcal{OSF}$) structure vs. Fuzzy Set ($\mathcal{FS}$)—lattice-theoretic calculi, and why it would be interesting to combine both.

[To be completed. . .]

Other points to discuss/elaborate:

- *Use fuzzy $\mathcal{OSF}$ generalization to aid, for example, in Machine Learning as a precious initial focusing step prior to exploiting number-analytical techniques such as Bayesian Nets or SVMs [111].*
- *While Propositional Logic ($\mathcal{PL}$) is a Boolean lattice, Fuzzy Propositional Logic ($\mathcal{FL}$) is a Brouwer lattice (also referred to as a Heyting algebra) [93].*[5]
  $\mathcal{OSF}$ Logic is also a lattice algebra: it has an infimum operation ($\mathcal{OSF}$ unification) and a supremum operation ($\mathcal{OSF}$ generalization). However, it is not a Boolean lattice. It is not even distributive [3], [5]. Seeing an $\mathcal{OSF}$ term as a logical constraint, $\mathcal{OSF}$ unification corresponds to Boolean conjunction, but $\mathcal{OSF}$ generalization is more approximate than Boolean disjunction. This is true as well for the lattice of first-order terms ordered by subsumption,[6] *as was first shown by Reynolds [104] and, simultaneously and independently, by Plotkin [101]. The lattice of ψ-terms (i.e., $\mathcal{OSF}$ terms in normal form) can be extended with a disjunction operation. So-extended ψ-terms, called ϵ-terms in [3] and [5], form a distributive lattice. It is not Boolean as it does not have complements. Further extending ϵ-terms with a restricted (constructive) form of complementation can provide a structure of Brouwer lattice.*[7]
- *cite relevant references and give BibTEX format and with a public pdf link (see to enrich existing file `main.bib` — see BibTEX templates in `bibtex-templates.bib`). N.B.: Not all current citations in this file are to be used, nor appropriate. They will be eventually reviewed and cleaned.*
- *potential applications (approximate information retrieval, ...)*

1.2 Objective

We seek to explore the "*fuzzification*" of operations on $\mathcal{FOT}$s and $\mathcal{OSF}$ constraints as used in Logic Programming (*e.g.*, [50, 83], and [15]). We proceed by conjugating the lattice-theoretic

[4] Degree 0 meaning distinct; degree 1.0 meaning identical.
[5] See Appendix Section A.3.1.
[6] That is, by $\mathcal{FOT}$ matching modulo variable renaming.
[7] See [5], Section 6.1, Page 336: "*Negative information*."

properties and $\mathcal{FOT}$s and $\mathcal{OSF}$-constraint graph algebras ordered by structure subsumption with a fuzzy interpretation of equality. We start with the Reynolds-Plotkin original characterization of first-order algebraic term subsumption as a lattice ordering, further extended into an $\mathcal{OSF}$ constraint subsumption lattice, and further enhanced with fuzzy lattice operations.

We approach this study from the perspective of information approximation. In this context, we focus essentially on fuzzifying lattice-theoretic operations on $\mathcal{FOT}$s and $\mathcal{OSF}$ constraints. These formal (strict) structures and the fuzzified lattice operations thereon may then be used in, *e.g.*, approximate Data and Knowledge representation and processing. As it has been demonstrated in all areas they have been applied to, Fuzzy Logic and Algebra offer greater flexibility and expressivity for performing approximate deduction (inference) and induction (abstraction). Fuzzy inference and abstraction operations over $\mathcal{FOT}$s and $\mathcal{OSF}$ constraints are therefore bound to offer an appreciated improvement of "smarts" in approximate pattern-based retrieval, mining, and learning. In addition, these fuzzy operations are effective, efficient, and conservative extensions of their crisp versions.

We shall always insist on formulating formal lattice-theoretic operations following *declarative*, as opposed to *procedural*, specifications in the form of syntax-driven transformational rules which can be applied in any order. Besides greatly simplifying proving their correctness by structural induction, this eliminates irrelevant control issues and side-effectable environments which typically clutter procedural specifications.

We also review some literature we felt relevant to our pursuit (even if only as potential topics for further research) dealing with related formal notions from general correspondances between arguments (positional or keyword) attributed structures based on syntactic terms, graph data structures, finite-state automata, and their fuzzification. We also provide a few examples of how these structures may be put to use for approximate Knowledge Representation as they offer a more flexible means to perform fuzzy deduction and induction over abstract attributed objects and concepts represented as order-sorted feature constraints.

1.3 Organization of contents

The rest of this document is organized as follows.

Chapter 2 focuses on first-order terms and fuzzifying their lattice operations. Section 2.1 covers the necessary background on $\mathcal{FOT}$s and Section 2.2 on $\mathcal{FOT}$ substitutions. Section 2.3 overviews background on the lattice of first-order terms, and offers an original declarative approach to $\mathcal{FOT}$ generalization that will later ease for us the task of fuzzifying this operation. Section 2.4 recalls the rules for $\mathcal{FOT}$ unification, while Section 2.5 develops an original approach for declarative $\mathcal{FOT}$ generalization. In Section 2.6, we proceed to fuzzify the Reynolds-Plotkin $\mathcal{FOT}$ subsumption lattice. We start with a specific formal fuzzification of $\mathcal{FOT}$ unification due to Maria Sessa. We then show how to make it more expressive by extending it to tolerate arity and/or argument-order mismatch in addition to just similar functors, and proceed to define their respective dual fuzzy generalization operations.[8] Finally, Section 2.7 recapitulates the contents of this chapter.

[8] Parts of this section appeared in [16]; see presentation slides in [17].

Chapter 3 focuses on order-sorted feature constraints and fuzzifying their lattice operations. Section 3.1 presents basic vocabulary and properties of order-sorted feature terms describing data and knowledge structures, and exposes the $\mathcal{OSF}$ term lattice operations. Section 3.2 pursues with the fuzzification of the lattice operations on $\mathcal{OSF}$ terms.

Chapter 4 puts this work in context. We review extant work that has some potential relation to the work presented here. Section 4.1 looks at other fuzzy unification work. Section 4.2 overviews some related work: Section 4.2.1 looks at similarity measures among order-sorted feature graphs, Section 4.2.2 discusses potential ties of our work with other topics, while Section 4.2.3 mentions even more general possible connections. Section 4.3 looks at implementations. Section 4.4 discusses pragmatic upshots illustrating some of these principles as proof-of-concept software realizations. It also goes into some details regarding the implementation of a fuzzy partial order's lattice operations. Section 4.5 speculates about potential applications. Section 4.6 is a proposal for a convincing use case in the area of intelligent information retrieval.

Chapter 5 concludes with some comments on the usefulness and future evolution of this work: Section 5.1 recapitulates, Section 5.2 indicates further work.

We have also adjoined a substantial appendix to (1) recall quickly some background material defining basic notions useful to understand more easily the notions exposed in this document, and (2) detail examples for some of this material.

Chapter 2

Version of January 8, 2019

First-Order Terms

The first-order term ($\mathcal{FOT}$) was introduced as a data structure in software programming by the Prolog language.[1] The $\mathcal{FOT}$ is Prolog's universal data structure in exactly the same way as the S-expression is that of LISP.[2] In this chapter, we formalize the $\mathcal{FOT}$ data structure as it is used in Logic Programming, then we expose in this formal setting its lattice-theoretic characterization before studying fuzzy extensions thereof.[3]

2.1 First-Order Term

Using formal algebra notation, we write $\mathcal{T}_{\Sigma,\mathcal{V}}$ for the set of $\mathcal{FOT}$s on an operator signature $\Sigma \stackrel{\text{def}}{=} \biguplus_{n\geq 0} \Sigma_n$ where Σ_n is a set of n-ary operator symbols.[4] The set $\mathcal{V}$ is a countably infinite set of variables. Also following Prolog's tradition, we shall designate an element f in Σ as a *functor*, with $\mathbf{arity}(f)$ denoting its number of arguments.[5] This set $\mathcal{T}_{\Sigma,\mathcal{V}}$ can then be defined inductively as:

$$\mathcal{T}_{\Sigma,\mathcal{V}} \stackrel{\text{def}}{=} \mathcal{V} \cup \{\, f(t_1,\ldots,t_n) \mid f \in \Sigma_n,\ n \geq 0, \text{ and } t_i \in \mathcal{T}_{\Sigma,\mathcal{V}},\ 0 \leq i \leq n \,\}.$$

Technically, an additional condition of well-foundedness requires that $\Sigma_0 \neq \emptyset$. We write c instead of $c()$ for a constant $c \in \Sigma_0$. Also, when the set Σ of functor symbols and the set $\mathcal{V}$ of variables are implicit from the context, we simply write $\mathcal{T}$ instead of $\mathcal{T}_{\Sigma,\mathcal{V}}$.

The set $\mathbf{var}(t)$ of variables occurring in a $\mathcal{FOT}$ $t \in \mathcal{T}$ is defined as:[6]

$$\mathbf{var}(t) \stackrel{\text{def}}{=} \begin{cases} \{\, X \,\} & \text{if } t = X \in \mathcal{V} \\ \bigcup_{i=1}^{n} \mathbf{var}(t_i) & \text{if } t = f(t_1,\ldots,t_n). \end{cases}$$

[1] https://en.wikipedia.org/wiki/Prolog
[2] https://en.wikipedia.org/wiki/Lisp_(programming_language)
[3] Parts of this chapter have appeared in [16]. For presentation slides, see [17].
[4] We shall use the notation "$\stackrel{\text{def}}{=}$" to mean "*is defined as*."
[5] When $\mathbf{arity}(f) = n$, this is sometimes denoted by writing f/n.
[6] We shall use Prolog's convention of writing variables with capitalized symbols.

A term t such that $\mathbf{var}(t) = \emptyset$ is called a *ground term*. We call $\mathcal{T}_\emptyset$ the subset of $\mathcal{T}$ of ground terms. The *depth* of a $\mathcal{FOT}$ t is a value in $\mathbb{N}$ defined inductively as:

$$\mathbf{depth}(t) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } t \in \mathcal{V} \cup \Sigma_0; \\ 1 + \max_{i=1}^{n} \mathbf{depth}(t_i) & \text{if } t = f(t_1, \ldots, t_n) \text{ with } n > 0. \end{cases}$$

The **var** and **depth** notation is extended to a set of terms $T \subset \mathcal{T}$ as $\mathbf{var}(T) \stackrel{\text{def}}{=} \bigcup_{t \in T} \mathbf{var}(t)$ and $\mathbf{depth}(T) \stackrel{\text{def}}{=} \max_{t \in T} \mathbf{depth}(t)$.

2.2 Substitution

In order to express the notion of instance of a term, the concept of variable substitution σ is formalized as a functional mapping $\sigma : \mathcal{V} \to \mathcal{T}$ that is the identity function everywhere on $\mathcal{V}$ except on a finite set of n variables, $n \in \mathbb{N}$, written $\mathbf{dom}(\sigma) \stackrel{\text{def}}{=} \{ X_k \mid X_k \neq \sigma(X_k) \}_{k=1}^{k=n}$, and called the domain of σ. The range of a substitution σ is the set of terms in $\mathcal{T}$ defined as $\mathbf{ran}(\sigma) \stackrel{\text{def}}{=} \{ t \in \mathcal{T} \mid \exists X \in \mathbf{dom}(\sigma) \text{ s.t. } \sigma(X) = t \}$.

Such a mapping σ from V to $\mathcal{T}$ is then extended homomorphically to a mapping $\bar{\sigma}$ from $\mathcal{T}$ to $\mathcal{T}$ as follows:

$$\bar{\sigma}(t) \stackrel{\text{def}}{=} \begin{cases} \sigma(X) & \text{if } t = X \in \mathcal{V} \\ f(\bar{\sigma}(t_1), \ldots, \bar{\sigma}(t_n)) & \text{if } t = f(t_1, \ldots, t_n) \end{cases} \tag{2.1}$$

which, because it coincides with σ on $\mathcal{V}$, will be written simply σ rather than $\bar{\sigma}$ even when applied to non-variable terms. In a similar fashion, substitutions may be applied to equations, as well as to sets of terms or equations in the obvious manner.

We shall denote as $\textsc{Subst}_{\mathcal{T}}$ the set of functions in $\mathcal{V} \to \mathcal{T}$ that are substitutions. Because it is non-identical only on a finite number of variables, we can express a substitution σ in $\textsc{Subst}_{\mathcal{T}}$ as a finite set of "*term/variable*" pairs of the form $\{ \sigma(X_k)/X_k \mid X_k \neq \sigma(X_k),\ k = 1, \ldots, n \}$ associating each of a finite set of n variables with a term not equal to it. When the number n of variables is equal to 0, this set is empty, giving the identity on $\mathcal{V}$, which we shall call the empty substitution. Each pair t/X in a substitution's set notation is read "*term t is substituted for all occurrences of variable X*."

By tradition, rather than the prefix parenthesized notation usually used for functional application, substitution application to a term is written in postfix notation; *viz.*, $t\sigma$ instead of $\sigma(t)$. Thus, as defined by Expression (2.1), a substitution σ is a function in $\mathcal{T} \to \mathcal{T}$ mapping a term t into another one noted $t\sigma$, called its (σ-)*instance*, obtained after replacing all occurrences in t (if any) of variables in $\mathbf{dom}(\sigma)$, the domain of the substitution, by the term associated with this variable by σ. If $\mathbf{var}(\mathbf{ran}(\sigma)) = \emptyset$, σ is called a *ground substitution*, and for any term t in $\mathcal{T}$, $t\sigma \in \mathcal{T}$, and is called a *ground instance* of t.

We define the composition of two substitutions $\sigma \in \textsc{Subst}_{\mathcal{T}}$ and $\theta \in \textsc{Subst}_{\mathcal{T}}$ seen as finite sets of non-identical term/variable pairs as the set of pairs written as $\sigma\theta$ and defined in terms of σ

and θ as:

$$\begin{aligned}\sigma\theta \quad \stackrel{\text{def}}{=} \quad & \big(\{\, t\theta/X \mid t/X \in \sigma \,\} \setminus \{\, X/X \mid X \in \mathbf{dom}(\sigma) \,\}\big) \\ & \cup \\ & \big(\theta \setminus \{\, u/Y \mid Y \in \mathbf{dom}(\sigma) \,\}\big).\end{aligned} \tag{2.2}$$

For terminology and proofs of formal properties of $\mathcal{FOT}$ substitutions as defined above and used in the remainder of this chapter, please refer to A.4.

2.3 $\mathcal{FOT}$ Subsumption Lattice

The lattice-theoretic properties of $\mathcal{FOT}$s as data structures were initially and independently studied by Reynolds (in [104]) and Plotkin (in [100] and [101]). They both noted that the set $\mathcal{T}$ is preordered by term subsumption (denoted as '$\preceq$'); *viz.*, $t \preceq t'$ (and we say: "t' *subsumes* t") iff there exists a variable substitution $\sigma \in \textbf{SUBST}_{\mathcal{T}}$ such that $t'\sigma = t$. Two $\mathcal{FOT}$s t and t' are considered "*equal up to variable renaming*" (denoted as $t \simeq t'$) whenever both $t \preceq t'$ and $t' \preceq t$. Then, the quotient set of first-order terms modulo variable renaming augmented with a bottom element $\mathcal{T}_{/\simeq} \cup \{\bot_{\mathcal{T}}\}$ has a lattice structure for subsumption. It has a least element $\bot_{\mathcal{T}}$ that corresponds to no term in $\mathcal{T}$, since there exists no term that is an instance of all terms. It has a top element which is the set of all variables $\mathcal{V}$, since $\mathcal{V}$ is the class of any variable modulo renaming.

Unification corresponds to the greatest lower bound (**glb**) operation. This is the case also for failure of unification as in this case the **glb** operation results in $\bot_{\mathcal{T}}$. Given two $\mathcal{FOT}$s t_1 and t_2, unifying them is seeking to compute a most general substitution of their variables σ such that: $t_1\sigma = t_2\sigma$.[7] Such a substitution, when one exists, is not unique since any more general substitution verifies the equation; indeed, then $t_1\sigma\theta = t_2\sigma\theta$ for any $\theta \in \textbf{SUBST}_{\mathcal{T}}$. We want only the most general such substitution. That is, for any other substitution $\theta \in \textbf{SUBST}_{\mathcal{T}}$ such that $t_1\theta = t_2\theta$, then necessarily $\theta \preceq \sigma$. This is why it is called the Most General Unifier (**mgu**) of t_1 and t_2 [105]. If no such substitution exists, unification fails and returns $\bot_{\mathcal{T}}$ as the **glb** of t_1 and t_2, and no substitution. Formally, this is equivalent to instantiating the two terms with a bottom substitution $\bot_{\textbf{SUBST}_{\mathcal{T}}}$ that is added to $\textbf{SUBST}_{\mathcal{T}}$. This new substitution is a zero element in the quotient monoid of substitutions with composition. Namely, for all $\sigma \in \textbf{SUBST}_{\mathcal{T}}$, $\sigma\bot_{\textbf{SUBST}_{\mathcal{T}}} = \bot_{\textbf{SUBST}_{\mathcal{T}}}\sigma = \bot_{\textbf{SUBST}_{\mathcal{T}}}$; which implies that $\bot_{\textbf{SUBST}_{\mathcal{T}}} \preceq \sigma$ for all $\sigma \in \textbf{SUBST}_{\mathcal{T}}$. From this, it follows necessarily that, for all $t \in \mathcal{T}$, $t\bot_{\textbf{SUBST}_{\mathcal{T}}} = \bot_{\mathcal{T}}$. Thus, when t_1 and t_2 are not unifiable, $\mathbf{mgu}(t_1, t_2) = \bot_{\textbf{SUBST}_{\mathcal{T}}}$.

The dual operation, generalization of two terms, yields a term that is their least upper bound (**lub**) for subsumption. That is, it finds the most specific term t, and two most general substitutions σ_1 and σ_2 such that $t_i = t\sigma_i$ for $i = 1, 2$. Importantly, unlike unification, generalization cannot fail. This is because two term structures having different functors, or two unequal terms one of which is a variable, are always generalizable into a new variable (which may be construed as "*anything*"). Also, generalization yields *two* substitutions rather than just one like for unification. Ths is because a variable in the generalizing term t may correspond to two different instantiations

[7] See A.4, *First-Order Term Substitutions.*

in t_1 and t_2. Unification, on the other hand, seeks the *same* instantiation for all the variables in t_1 and t_2 to compute their most general common instance.

This can be summarized as the lattice diagram shown in Figure 2.1. In this diagram, given a pair of terms $\langle t_1, t_2 \rangle$, the pair of substitutions $\langle \sigma_1, \sigma_2 \rangle$ are their respective most general generalizers, and the substitution σ is the pair's most general unifier (**mgu**).

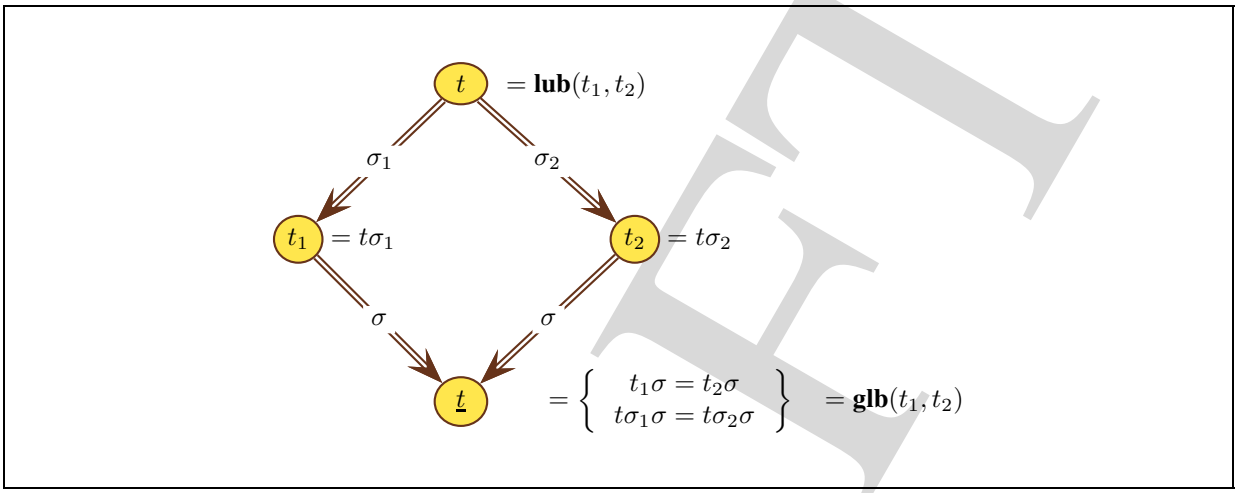


Figure 2.1: Subsumption lattice operations

Example 2.1 $\mathcal{FOT}$ **lattice operations** — Consider the terms t_1 and t_2 defined as:

$$t_1 \stackrel{\text{def}}{=} f(a, g(X_1, b), Y_1, g(a, Y_1)),$$
$$t_2 \stackrel{\text{def}}{=} f(X_2, Y_2, g(X_2, g(X_2, b)), g(X_2, g(a, Z_2))).$$

Their most general unifier $\mathbf{mgu}(t_1, t_2)$ is the substitution σ given by:

$$\sigma = \{\, a/X_2, g(X_1, b)/Y_2, g(a, g(a, b))/Y_1, g(a, b)/Z_2 \,\}$$

and so their greatest lower bound $\mathbf{glb}(t_1, t_2) = \underline{t}$ is given by:

$$\underline{t} = t_1\sigma = t_2\sigma = f(a, g(X_1, b), g(a, g(a, b)), g(a, g(a, g(a, b)))).$$

Dually, their least upper bound $\mathbf{lub}(t_1, t_2) = t$ is given by $t = f(X, Y, Z, g(U, V))$, with their most general generalizers $\langle \sigma_1, \sigma_2 \rangle$ such that:

$$t_1 = t\sigma_1 \text{ with } \sigma_1 = \{\, a/X, g(X_1, b)/Y, Y_1/Z, a/U, Y_1/V \,\}$$
$$t_2 = t\sigma_2 \text{ with } \sigma_2 = \{\, X_2/X, Y_2/Y, g(X_2, g(X_2, b))/Z, X_2/U, g(a, Z_2)/V \,\}.$$

Next, we formalize these lattice operations on $\mathcal{FOT}$s by specifiying them as declarative constraint normalization.

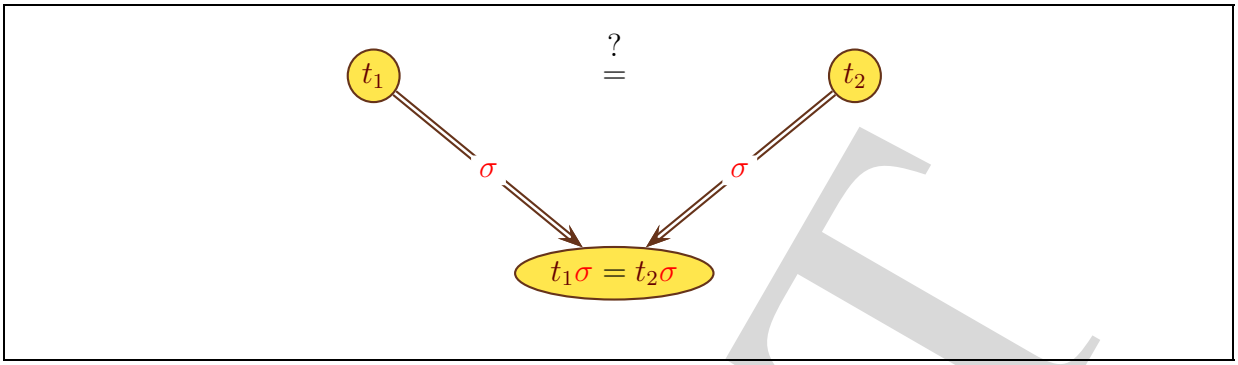


Figure 2.2: $\mathcal{FOT}$ unification as a constraint

TERM DECOMPOSITION:

$$\frac{E \cup \{\, f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n) \,\}}{E \cup \{\, s_1 \doteq t_1, \ldots, s_n \doteq t_n \,\}} \; [n \geq 0]$$

VARIABLE ERASURE:

$$\frac{E \cup \{\, X \doteq X \,\}}{E}$$

VARIABLE ELIMINATION:

$$\frac{E \cup \{\, X \doteq t \,\}}{E[X \leftarrow t] \cup \{\, X \doteq t \,\}} \left[\begin{array}{l} X \notin \mathbf{var}(t) \\ X \text{ occurs in } E \end{array} \right]$$

EQUATION ORIENTATION:

$$\frac{E \cup \{\, t \doteq X \,\}}{E \cup \{\, X \doteq t \,\}} \; [t \notin \mathcal{V}]$$

Figure 2.3: Herbrand-Martelli-Montanari unification rules

2.4 $\mathcal{FOT}$ Unification Rules

Figure 2.2 illustrates $\mathcal{FOT}$ unification as a commutative diagram constraint. Solving such a constraint is done by a system of equation-normalization rules that we shall call Herbrand-Martelli-Montanari [68], [92]. These rules are given in Figure 2.3. Each rule can be proven correct as a solution-preserving transformation of a set of equations. In Rule **VARIABLE ELIMINATION**, the notation $E[X \leftarrow t]$ denotes the set of equations E in which all occurrences of variable X have been replaced with the term t.

Thus, we can use these rules to unify two $\mathcal{FOT}$s t_1 and t_2, starting with the singleton set of equations $E \stackrel{\text{def}}{=} \{\, t_1 \doteq t_2 \,\}$.[8] Then, we transform this set of equations using any applicable rule in any order until none applies. This always terminates into a finite set of equations E'. If all the equations in E' are of the form $X \doteq t$ with X occurring nowhere else in E', then this is a most general unifying substitution (up to consistent variable renaming) $\sigma \stackrel{\text{def}}{=} \{\; t/X \mid X \doteq t \in E' \;\}$ solving the original equation (*i.e.*, $t_1\sigma = t_2\sigma$); otherwise, there is no solution.

In the rules of Figure 2.3, Rule **VARIABLE ELIMINATION** has the side condition $X \notin \mathbf{var}(t)$ to prevent cyclic terms (such as, *e.g.*, $X = f(X)$) whose presence indicates no $\mathcal{FOT}$ solutions.

[8] In such equations, we use the notation $t_1 \doteq t_2$ not to confuse it with the equality symbol "=" (at the meta-level).

This condition could be omitted if wished, thus extending the set of $\mathcal{FOT}$s and solutions of equations to rational $\mathcal{FOT}$s—also called "*infinite trees*" (see, *e.g.*, [122], [78], [51]).

Example 2.2 $\mathcal{FOT}$ **unification** — Consider the equation set $\{ t_1 \doteq t_2 \}$ for the terms t_1 and t_2 of Example 2.1:

$$\{ f(a, g(X_1, b), Y_1, g(a, Y_1)) \doteq f(X_2, Y_2, g(X_2, g(X_2, b)), g(X_2, g(a, Z_2))) \}$$

and let us apply the rules of Figure 2.3:

- Rule **Term Decomposition**:

$$\{ a \doteq X_2, g(X_1, b) \doteq Y_2, Y_1 \doteq g(X_2, g(X_2, b)), g(a, Y_1) \doteq g(X_2, g(a, Z_2)) \};$$

- Rule **Equation Orientation** to $a \doteq X_2$:

$$\{ X_2 \doteq a, g(X_1, b) \doteq Y_2, Y_1 \doteq g(X_2, g(X_2, b)), g(a, Y_1) \doteq g(X_2, g(a, Z_2)) \};$$

- Rule **Variable Elimination** to $X_2 \doteq a$:

$$\{ X_2 \doteq a, g(X_1, b) \doteq Y_2, Y_1 \doteq g(a, g(a, b)), g(a, Y_1) \doteq g(a, g(a, Z_2)) \};$$

- Rule **Equation Orientation** to $g(X_1, b) \doteq Y_2$:

$$\{ X_2 \doteq a, Y_2 \doteq g(X_1, b), Y_1 \doteq g(a, g(a, b)), g(a, Y_1) \doteq g(a, g(a, Z_2)) \};$$

- Rule **Variable Elimination** to $Y_1 \doteq g(a, g(a, b))$:

$$\{ X_2 \doteq a, Y_2 \doteq g(X_1, b), Y_1 \doteq g(a, g(a, b)), g(a, g(a, g(a, b))) \doteq g(a, g(a, Z_2)) \};$$

- Rule **Term Decomposition** to $g(a, g(a, g(a, b))) \doteq g(a, g(a, Z_2))$:

$$\{ X_2 \doteq a, Y_2 \doteq g(X_1, b), Y_1 \doteq g(a, g(a, b)), a \doteq a, g(a, g(a, b)) \doteq g(a, Z_2) \};$$

- Rule **Term Decomposition** to $a \doteq a$:

$$\{ X_2 \doteq a, Y_2 \doteq g(X_1, b), Y_1 \doteq g(a, g(a, b)), g(a, g(a, b)) \doteq g(a, Z_2) \};$$

- Rule **Term Decomposition** to $g(a, g(a, b)) \doteq g(a, Z_2)$:

$$\{ X_2 \doteq a, Y_2 \doteq g(X_1, b), Y_1 \doteq g(a, g(a, b)), a \doteq a, g(a, b) \doteq Z_2 \};$$

- Rule **Term Decomposition** to $a \doteq a$:

$$\{ X_2 \doteq a, Y_2 \doteq g(X_1, b), Y_1 \doteq g(a, g(a, b)), g(a, b) \doteq Z_2 \};$$

- Rule **Equation Orientation** to $g(a, b) \doteq Z_2$:

$$\{ X_2 \doteq a, Y_2 \doteq g(X_1, b), Y_1 \doteq g(a, g(a, b)), Z_2 \doteq g(a, b) \}.$$

This last equation set is in normal form defining the substitution

$$\sigma = \{\, a/X_2, g(X_1, b)/Y_2, g(a, g(a, b))/Y_1, g(a, b)/Z_2 \,\}$$

. So the greatest lower bound $\underline{t} \stackrel{\text{def}}{=} \mathbf{glb}(t_1, t_2)$ of:

$$t_1 \stackrel{\text{def}}{=} f(a, g(X_1, b), Y_1, g(a, Y_1))$$

and:

$$t_2 \stackrel{\text{def}}{=} f(X_2, Y_2, g(X_2, g(X_2, b)), g(X_2, g(a, Z_2)))$$

is given by:

$$\underline{t} = t_1\sigma = t_2\sigma = f(a, g(X_1, b), g(a, g(a, b)), g(a, g(a, g(a, b)))).$$

2.5 $\mathcal{FOT}$ Generalization Rules

Next, we present a set of constraint normalization rules for $\mathcal{FOT}$ generalization which are equivalent to the procedural method of Reynolds and Plotkin. The advantage of specifying this operation in this manner rather than procedurally as done originally by Reynolds and Plotkin is that each rule or axiom relates a pair of prior substitutions to a pair of posterior substitutions based only on local syntactic-pattern properties of the terms to generalize, and this without resorting to side-effects on global structures. In this way, the terms and substitutions involved are derived as solutions of logical syntactic constraints. In addition, correctness of the so-specified operation is made much easier to establish since we only need to prove each rule's correctness independently of that of the others. Finally, the rules also provide an effective means for the derivation of an operational semantics for the so-specified operation by constraint solving, without need for control specification as any applicable rule may be invoked in any order.[9]

DEFINITION 2.1 (GENERALIZATION JUDGMENT) *A generalization judgment is an expression of the form:*

$$\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} \vdash \begin{pmatrix}t_1\\ t_2\end{pmatrix} t \begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix} \tag{2.3}$$

where $\sigma_i \in$ **SUBST** $_{\mathcal{T}}$, $\theta_i \in$ **SUBST** $_{\mathcal{T}}$, $t_i \in \mathcal{T}$ $(i = 1, 2)$, *and* $t \in \mathcal{T}$.

Informally, it reads: "*given two prior substitutions* σ_1 *and* σ_2, *the term* t *is the least generalization of terms* $t_1\sigma_1$ *and* $t_2\sigma_2$ *with posterior substitutions* θ_1 *and* θ_2." How all the constituents of such a generalization judgment must be related to constitute what we shall consider a valid judgment, is defined next.

[9] Such as the Herbrand-Martelli-Montanari rules *w.r.t.* to Robinson's procedural unification algorithm.

DEFINITION 2.2 (GENERALIZATION JUDGMENT VALIDITY) *A $\mathcal{FOT}$ generalization judgment such as (2.3) is said to be valid whenever, for $i = 1, 2$:*

1. $t_i\sigma_i = t\theta_i$; *and,*
2. $\exists\, \delta_i \in$ **SUBST**$_{\mathcal{T}}$ *s.t.* $t_i = t\delta_i$ *and* $\theta_i = \delta_i\sigma_i$ *(i.e.,* $t_i \preceq t$ *and* $\theta_i \preceq \sigma_i$*).*

Figure 2.4 illustrates the validity of a $\mathcal{FOT}$ generalization judgment as a commutative diagram constraint.

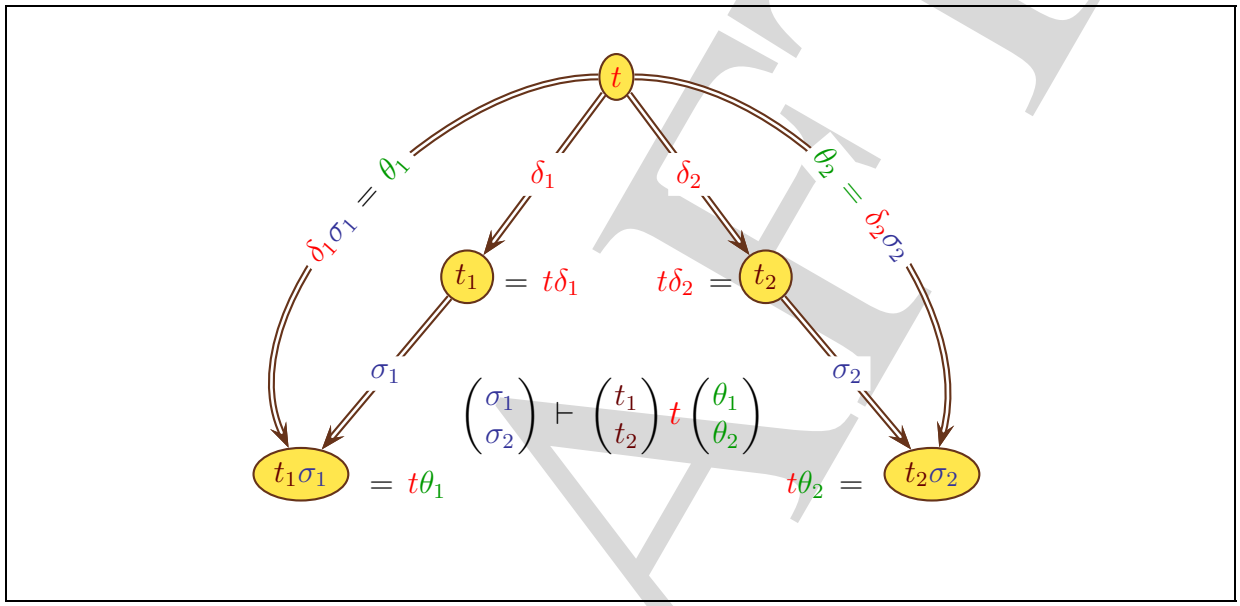


Figure 2.4: $\mathcal{FOT}$ generalization judgment validity as a constraint

DEFINITION 2.3 (TRIVIAL $\mathcal{FOT}$ GENERALIZATION JUDGMENT) *The $\mathcal{FOT}$ generalization judgment:*

$$\texttt{true} \stackrel{def}{=} \begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix} \vdash \begin{pmatrix}t\\ t\end{pmatrix} t \begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix} \tag{2.4}$$

where t is an arbitrary term in $\mathcal{T}$ is called a "trivial $\mathcal{FOT}$ generalization judgment."

LEMMA 2.1 (TRIVIAL $\mathcal{FOT}$ GENERALIZATION JUDGMENT VALIDITY) *The trivial $\mathcal{FOT}$ generalization judgment* true *is always valid.*

> **PROOF** This follows from Definition 2.2 since in this particular case the equations of the first condition of Definition (2.2) becomes $t = t$, which is trivially true for any term $t \in \mathcal{T}$. □

Contrary to unification normalization rules which are expressed as conditional rewrite rules whereby a prior form (the "numerator") is related to a posterior form (the "denominator"), these normalization rules are more naturally rendered as (conditional) Horn clauses of judgments (*i.e.*,

Prolog clauses of judgments). This is as convenient as rewrite rules since a Prolog-like operational semantics can then readily provide an effective interpretation.[10] Thus, a generalization rule is of the form:

$$\frac{[\phi] \quad J_1 \quad \ldots \quad J_n}{J} \tag{2.5}$$

where ϕ is an optional side meta-condition, and $J, J_1, \ldots, J_n$ are judgments, and it reads, "*whenever the side condition ϕ holds, if all the n antecedent judgments $J_1, \ldots, J_n$ are valid, then the consequent judgment J is also valid.*" Such a generalization rule without a specified antecedent (a "numerator") is called a "*generalization axiom.*" Such an axiom is said to be valid iff its consequent (the "denominator") is valid whenever its optional side condition holds. It is equivalent to a rule where the only antecedent is the trivial generalization judgment $\mathtt{true}$.

DEFINITION 2.4 (GENERALIZATION RULE CORRECTNESS) *A generalization rule such as Rule (2.5) is correct iff J_k is a valid judgment for all $k = 1, \ldots, n$ implies that J is a valid judgment, whenever the side condition ϕ holds.*

Given t_1 and t_2 two $\mathcal{FOT}$s, in order to find the most specific term t and most general substitutions σ_i, $i = 1, 2$, such that $t\sigma_i = t_i$, $i = 1, 2$, one needs to establish the generalization judgment:

$$\begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix} \vdash \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} t \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}. \tag{2.6}$$

In other words, this expresses the upper half of Figure 2.1 whereby $t = \mathbf{lub}(t_1, t_2)$, with most general substitutions σ_1 and σ_2. We give a complete set of normalization axioms and rule for generalization for all syntactic patterns in Figure 2.5.

Rule "**EQUAL FUNCTORS**" specifies a sequence of judgments constrained as a sequence. It does so exactly as a so-called "Definite Clause Grammar" (or DCG) rule does with a Prolog clause.[11] This rule uses an "*unapply*" operation ('$\uparrow$') on a pair of terms (t_1, t_2) given a pair of substitutions (σ_1, σ_2). It may be conceived as (and in fact is) the result of simultaneously "*unapplying*" σ_i from t_i into a common variable X only if such X is bound to t_i by σ_i, for $i = 1, 2$. If there is no such a variable, it is the identity. This operation avoids the introduction of

[10] This operational semantics is also efficient because it does not need backtracking as long as the complete set of conditions of a ruleset covers all but mutually exclusive syntactic patterns.

[11] A DCG rule (see https://www.metalevel.at/prolog/dcg) is a Horn rule expressing constraints on a sequence of words constituting a sentence. The judgment sequencing in the rules we define uses exactly the same kind of constraint: the posterior pair of substitutions of a judgment must match the prior pair of substitutions of the judgment following it. However, contrary to a DCG rule that constrains an *ordered* sequence of constituents, the order of constraints on the antecedent judgments on the arguments is arbitrary. We choose the same order as that of the arguments as it is the most natural, but it could be any of its permutations as long as the sequence's posterior/prior constraints are consistent with the chosen argument ordering.

EQUAL VARIABLES

$$\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} \vdash \begin{pmatrix}X\\ X\end{pmatrix} X \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}$$

VARIABLE-TERM

$[t_1 \in \mathcal{V} \text{ or } t_2 \in \mathcal{V};\ \ t_1 \neq t_2;\ \ X \text{ is new}]$

$$\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} \vdash \begin{pmatrix}t_1\\ t_2\end{pmatrix} X \begin{pmatrix}\sigma_1\{\, t_1/X \,\}\\ \sigma_2\{\, t_2/X \,\}\end{pmatrix}$$

UNEQUAL FUNCTORS

$[m \geq 0, n \geq 0;\ \ m \neq n \text{ or } f \neq g;\ \ X \text{ is new}]$

$$\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} \vdash \begin{pmatrix}f(s_1,\ldots,s_m)\\ g(t_1,\ldots,t_n)\end{pmatrix} X \begin{pmatrix}\sigma_1\{\, f(s_1,\ldots,s_m)/X \,\}\\ \sigma_2\{\, g(t_1,\ldots,t_n)/X \,\}\end{pmatrix}$$

EQUAL FUNCTORS

$[n \geq 0]$

$$\frac{\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} \vdash \begin{pmatrix}s_1\\ t_1\end{pmatrix} \uparrow \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} u_1 \begin{pmatrix}\sigma_1^1\\ \sigma_2^1\end{pmatrix} \quad \ldots \quad \begin{pmatrix}\sigma_1^{n-1}\\ \sigma_2^{n-1}\end{pmatrix} \vdash \begin{pmatrix}s_n\\ t_n\end{pmatrix} \uparrow \begin{pmatrix}\sigma_1^{n-1}\\ \sigma_2^{n-1}\end{pmatrix} u_n \begin{pmatrix}\sigma_1^n\\ \sigma_2^n\end{pmatrix}}{\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} \vdash \begin{pmatrix}f(s_1,\ldots,s_n)\\ f(t_1,\ldots,t_n)\end{pmatrix} f(u_1,\ldots,u_n) \begin{pmatrix}\sigma_1^n\\ \sigma_2^n\end{pmatrix}}$$

Figure 2.5: Generalization axioms and rule

a new variable when generalizing two already generalized terms. Formally, this is defined as:

$$\begin{pmatrix}t_1\\ t_2\end{pmatrix} \uparrow \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} \stackrel{\text{def}}{=} \begin{cases} \begin{pmatrix}X\\ X\end{pmatrix} & \text{if } \exists\, X \in \mathcal{V},\ t_i = X\sigma_i,\ \text{for } i = 1,2; \\ \begin{pmatrix}t_1\\ t_2\end{pmatrix} & \text{otherwise.} \end{cases} \tag{2.7}$$

Note also that Rule "**EQUAL FUNCTORS**" is defined for $n \geq 0$. For $n = 0$, it becomes the following axiom for any constant c and any two substitutions σ_i, $i = 1, 2$:

$$\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} \vdash \begin{pmatrix}c\\ c\end{pmatrix} c \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}. \tag{2.8}$$

Referring to the axioms (seen as rules with no antecedent) and the rule of Figure 2.5, we first establish the following fact.

LEMMA 2.2 *In Rule* **EQUAL FUNCTORS** *of Figure 2.5, taking* $\sigma_i^0 \stackrel{\text{def}}{=} \sigma_i$, *for* $i = 1, 2$, *the substitutions* $\sigma_i^0, \ldots, \sigma_i^n$ *are such that, for all* k, $1 \leq k \leq n$, $\sigma_i^k \preceq \sigma_i^{k-1}$, *for* $i = 1, 2$.

PROOF We proceed by induction on the depth d of the terms; *i.e.*, we consider only terms of depth less than or equal to d.

1. $d = 0$: This limits terms to constants and variables. The inequality between prior and posterior substitutions is verified for the three first axioms: the posterior substitutions are all either equal to the corresponding prior substitutions or of the form $\theta = \sigma\{t/X\}$ where X is a new variable and σ is the corresponding prior substitution; that is, $\theta \preceq \sigma$. As well, when limited to terms of 0 depth, Rule **EQUAL FUNCTORS** becomes the single judgment Axiom (2.8), which preserves the substitutions.
2. $d > 0$: Let us assume that this is true for all terms of depth strictly less than d. We now consider two terms at least one of which is of depth d. For Axiom **EQUAL VARIABLES**, the same argument given above for the case $d = 0$ justifies concluding that $\theta \preceq \sigma$, since then $\theta = \sigma$. For Axiom **VARIABLE-TERM** and Axiom **UNEQUAL FUNCTORS**, this true for terms t_1 and t_2 of any depth since the posterior substitutions are both less general then the corresponding prior substitutions. As for Rule **EQUAL FUNCTORS**, there are two possible cases for the generalized terms in its consequent (the "denominator"):
   (a) $n = 0$: then, the conclusion follows true by Axiom (2.8).
   (b) $n \geq 0$: since the unapply operation (2.7) yields either a pair of terms having the same depth as the corresponding terms it is applied to, or 0 (because it can only be a new variable), we can say that all the terms of unapplied pairs of arguments in the judgments of the rule's antecedent (the "numerator") are of depth at most $d - 1$. Therefore, all the terms in the n antecedent judgments verify our inductive hypothesis; namely: $\sigma_i^k \preceq \sigma_i^{k-1}$, for all $k = 1, \ldots, n$. Then, by transitivity of the "more general" ordering on substitutions, the conclusion follows.

Hence, this establishes that, for both $i = 1, 2$, σ_i^k is monotonically refined from more general to less as k increases from 1 to n. □

From this lemma, the following corollary follows by transitivity of the $\preceq$ preorder on substitutions.

COROLLARY 2.1 *In Rule* **EQUAL FUNCTORS**, *the substitutions* σ_i^k *are such that, for all* k, $1 \leq k \leq n$, $\sigma_i^n \preceq \sigma_i^{n-1} \preceq \ldots \preceq \sigma_i^1 \preceq \sigma_i^0$, *for* $i = 1, 2$.

It is also verified in the proof of the following theorem.

THEOREM 2.1 *The axioms and the rule of Figure 2.5 are correct.*

PROOF We must show that they verify the conditions of Definition 2.4. For each of the three axioms of Figure 2.5 this means that they must be always valid as judgments, verifying the conditions of Definition 2.2, which are:

- *Condition 1*: $t_i\sigma_i = t\theta_i$,
- *Condition 2*: $t_i \preceq t$ and $\theta_i \preceq \sigma_i$

for $i = 1, 2$, for a generalization judgment such as (2.3) in Definition 2.1. These conditions for the axioms and the rule of Figure 2.5 translate as the following.

Condition 1.

- **EQUAL VARIABLES**: it amounts to the two identities $X\sigma_i = X\sigma_i$, $i = 1, 2$;

– **VARIABLE-TERM**: it amounts to the two identities $t_i\sigma_i = t_i\sigma_i,\ i = 1, 2$;

– **UNEQUAL FUNCTORS**: it amounts to the two equations:

$$\begin{array}{rcl} f(s_1, \ldots, s_n)\sigma_1 & = & X\sigma_1\{\, f(s_1, \ldots, s_n)/X\,\}, \\ g(t_1, \ldots, t_n)\sigma_2 & = & X\sigma_2\{\, g(t_1, \ldots, t_n)/X\,\}, \end{array}$$

which, because X is a new variable that does not occurs in either σ_1 or σ_2, can be simplified to the identities:

$$\begin{array}{rcl} f(s_1, \ldots, s_n) & = & f(s_1, \ldots, s_n), \\ g(t_1, \ldots, t_n) & = & g(t_1, \ldots, t_n). \end{array}$$

Condition 2. All threes cases are tautologies:

– **EQUAL VARIABLES**: $X \preceq X$ and $\sigma_i \preceq \sigma_i,\ i = 1, 2$;

– **VARIABLE-TERM**: $t_i \preceq X$ and $\sigma_i\{t_i/X\} \preceq \sigma_i,\ i = 1, 2$;

– **UNEQUAL FUNCTORS**:

$$\begin{array}{l} f(s_1, \ldots, s_n) \preceq X \text{ and } \sigma_1\{\, f(s_1, \ldots, s_n)/X\,\} \preceq \sigma_1, \\ g(s_1, \ldots, s_n) \preceq X \text{ and } \sigma_2\{\, g(s_1, \ldots, s_n)/X\,\} \preceq \sigma_2. \end{array}$$

As for Rule **EQUAL FUNCTORS**, as required by Definition 2.4, we must show that if all the judgments in the numerator are valid, then the judgment in the denominator must be valid too. Let us proceed by induction on the argument-position number k, for $k = 1, \ldots, n$.

For $n = 0$, this rule becomes Axiom (2.8), a judgment that is trivially valid since the conditions of Definition 2.2 become the identity $c = c$, the term inequality $c \preceq c$, and the substitution inequalities $\sigma_i \preceq \sigma_i$, for $i = 1, 2$.

For $n > 0$, a fuzzy judgment in the rule's antecedent, for each argument-position $k = 1, \ldots, n$, is of the form:

$$\begin{pmatrix} \sigma_1^{k-1} \\ \sigma_2^{k-1} \end{pmatrix} \vdash \begin{pmatrix} s_k \\ t_k \end{pmatrix} \uparrow \begin{pmatrix} \sigma_1^{k-1} \\ \sigma_2^{k-1} \end{pmatrix} u_k \begin{pmatrix} \sigma_1^{k} \\ \sigma_2^{k} \end{pmatrix}$$

that is, the form given by Definition 2.1, whose formal validity conditions are given by Definition 2.2, which in the above case is equivalent to:

$$\begin{pmatrix} v_1^k \\ v_2^k \end{pmatrix} \stackrel{\text{def}}{=} \begin{pmatrix} s_k \\ t_k \end{pmatrix} \uparrow \begin{pmatrix} \sigma_1^{k-1} \\ \sigma_2^{k-1} \end{pmatrix} \text{ and } \begin{pmatrix} \sigma_1^{k-1} \\ \sigma_2^{k-1} \end{pmatrix} \vdash \begin{pmatrix} v_1^k \\ v_2^k \end{pmatrix} u_k \begin{pmatrix} \sigma_1^{k} \\ \sigma_2^{k} \end{pmatrix}.$$

Let us now assume that all the judgments in the rule's antecedent are valid. That is, for $k = 1, \ldots, n$, for $i = 1, 2$ (taking $\sigma_i^0 \stackrel{\text{def}}{=} \sigma_i$):

– Condition 1 of Definition 2.2 holds:

$$u_k\sigma_i^k = v_i^k\sigma_i^{k-1}; \tag{2.9}$$

– Condition 2 of Definition 2.2 holds:

$$v_i^k \preceq u_k \text{ and } \sigma_i^k \preceq \sigma_i^{k-1}. \tag{2.10}$$

Condition 1. By Equation (2.7), this means that for all $k = 1, \ldots, n$:

$$\binom{v_1^k}{v_2^k} = \begin{cases} \binom{X}{X} & \text{if } s_k = X\sigma_1^{k-1} \text{ and } t_k = X\sigma_2^{k-1} \text{ for some variable} X; \\ \binom{s_k}{t_k} & \text{otherwise.} \end{cases}$$

In other words, for each $k = 1, \ldots, n$, there are two cases:

1. $s_k = X\sigma_1^{k-1}$ and $t_k = X\sigma_2^{k-1}$ for some variable X; then, by Axiom **EQUAL VARIABLES**, we must have $u_k = X$, and $\sigma_i^k = \sigma_i^{k-1}$ for $i = 1, 2$; and therefore Equations (2.9) entail:

$$\begin{aligned} s_k\sigma_1^{k-1} &= X\sigma_1^{k-1}\sigma_1^{k-1} = X\sigma_1^{k-1} = X\sigma_1^k = u_k\sigma_1^k \\ t_k\sigma_2^{k-1} &= X\sigma_2^{k-1}\sigma_2^{k-1} = X\sigma_2^{k-1} = X\sigma_2^k = u_k\sigma_2^k. \end{aligned}$$

2. There is no such variable X; in which case, Equations (2.9) also become:

$$\begin{aligned} s_k\sigma_1^{k-1} &= u_k\sigma_1^k \\ t_k\sigma_2^{k-1} &= u_k\sigma_2^k. \end{aligned}$$

Thus, by the only non-identical transformation relating prior and posteriors substitutions in the axioms, for any argument position k, $1 \leq k \leq n$, we have:

$$\sigma_i^k = \sigma_i^0\{\, \tau_1/X_1 \,\} \ldots \{\, \tau_\ell/X_\ell \,\}$$

where each of the variables $X_1 \ldots X_\ell$, with $0 \leq \ell$, is a variable possibly introduced in the validity of the judgment corresponding to some argument preceding position k. Therefore, for any argument position k, $1 \leq k \leq n$:

$$\begin{aligned} s_k\sigma_1^0 = s_k\sigma_1^1 = \ldots = s_k\sigma_1^{k-1} \\ t_k\sigma_2^0 = t_k\sigma_2^1 = \ldots = t_k\sigma_2^{k-1} \end{aligned}$$

as well as:

$$\begin{aligned} u_k\sigma_1^k = u_k\sigma_1^{k+1} = \ldots = u_k\sigma_1^n \\ u_k\sigma_2^k = u_k\sigma_2^{k+1} = \ldots = u_k\sigma_2^n \end{aligned}$$

because σ_i^k affects only new variables introduced in some axioms verifying the validity of a subterm of argument at position k; and because the same variable in u_k is always instantiated by the same term, and thus as well all at higher argument positions.

This means that in both cases we have, for all $k = 1, \ldots, n$:

$$\begin{aligned} s_k\sigma_1^0 &= u_k\sigma_1^n, \\ t_k\sigma_2^0 &= u_k\sigma_2^n. \end{aligned}$$

Therefore, for $k = n$:

$$f(s_1, \ldots, s_n)\sigma_1^0 = f(u_1, \ldots, u_n)\sigma_1^n,$$
$$f(t_1, \ldots, t_n)\sigma_2^0 = f(u_1, \ldots, u_n)\sigma_2^n.$$

This proves Condition 1.

Condition 2. By transitivity of the $\leq$ ordering on approximation degrees and that of the $\preceq$ preorder on $\textbf{SUBST}_{\mathcal{T}}$, both parts of Condition 2 of our induction hypothesis (2.10) implies that $f(s_1, \ldots, s_n) \preceq f(u_1, \ldots, u_n)$, $f(t_1, \ldots, t_n) \preceq f(u_1, \ldots, u_n)$ and $\sigma_i \preceq \sigma_i^n$, for $i = 1, 2$, which completes the proof. □

In particular, with empty prior substitutions, we obtain the following corollary.

COROLLARY 2.2 ($\mathcal{FOT}$ GENERALIZATION) *Whenever the judgment*

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix} \vdash \begin{pmatrix}t_1\\ t_2\end{pmatrix} t \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}$$

is valid, then $t\sigma_i = t_i$, *for* $i = 1, 2$.

Example 2.3 $\mathcal{FOT}$ **generalization** — Consider the terms $f(a, a, a)$ and $f(b, c, c)$.

- Let us find term t and substitutions σ_1 and σ_2 such that $t\sigma_1 = f(a, a, a)$ and $t\sigma_2 = f(b, c, c)$; that is, let us try to solve the following constraint problem:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix} \vdash \begin{pmatrix}f(a,a,a)\\ f(b,c,c)\end{pmatrix} t \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}$$

- By Rule **EQUAL FUNCTORS**, we must have $t = f(u_1, u_2, u_3)$ since:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix} \vdash \begin{pmatrix}f(a,a,a)\\ f(b,c,c)\end{pmatrix} f(u_1,u_2,u_3) \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}$$

  where:

  - u_1 is the generalization of $\begin{pmatrix}a\\ b\end{pmatrix} \uparrow \begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}$; that is, of a and b; and by Rule **UNEQUAL FUNCTORS**:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix} \vdash \begin{pmatrix}a\\ b\end{pmatrix} X \begin{pmatrix}\{\,a/X\,\}\\ \{\,b/X\,\}\end{pmatrix}$$

    therefore $u_1 = X$;

  - u_2 is the generalization of $\begin{pmatrix}a\\ c\end{pmatrix} \uparrow \begin{pmatrix}\{\,a/X\,\}\\ \{\,b/X\,\}\end{pmatrix}$; that is, of a and c; and by Rule **UNEQUAL FUNCTORS**:

$$\begin{pmatrix}\{\,a/X\,\}\\ \{\,b/X\,\}\end{pmatrix} \vdash \begin{pmatrix}a\\ c\end{pmatrix} Y \begin{pmatrix}\{\,a/X, a/Y\,\}\\ \{\,b/X, c/Y\,\}\end{pmatrix}$$

    so $u_2 = Y$;

– u_3 is the generalization of $\begin{pmatrix} a \\ c \end{pmatrix} \uparrow \begin{pmatrix} \{\, a/X, a/Y \,\} \\ \{\, b/X, c/Y \,\} \end{pmatrix}$; that is, of Y and Y; and by Rule **EQUAL VARIABLES**:

$$\begin{pmatrix} \{\, a/X, a/Y \,\} \\ \{\, b/X, c/Y \,\} \end{pmatrix} \vdash \begin{pmatrix} Y \\ Y \end{pmatrix} Y \begin{pmatrix} \{\, a/X, a/Y \,\} \\ \{\, b/X, c/Y \,\} \end{pmatrix}$$

so $u_3 = Y$;

- therefore, this yields:

$$\begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix} \vdash \begin{pmatrix} f(a,a,a) \\ f(b,c,c) \end{pmatrix} f(X,Y,Y) \begin{pmatrix} \{\, a/X, a/Y \,\} \\ \{\, b/X, c/Y \,\} \end{pmatrix}$$

that is $t = f(X,Y,Y)$ with $\sigma_1 = \{\, a/X, a/Y \,\}$ such that $t\sigma_1 = f(a,a,a)$, and $\sigma_2 = \{\, b/X, c/Y \,\}$ such that $t\sigma_2 = f(b,c,c)$.

Example 2.4 Generalization of ground $\mathcal{FOT}$s — Let us now consider the terms $f(a, g(b,a), b))$ and $f(b, g(a,b), a))$;

- let us find term t and substitutions σ_1 and σ_2 s.t. $t\sigma_1 = f(a, g(b,a), b))$ and $t\sigma_2 = f(b, g(a,b), a)$; *i.e.*, let us try to solve the following constraint problem:

$$\begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix} \vdash \begin{pmatrix} f(a, g(b,a), b) \\ f(b, g(a,b), a) \end{pmatrix} t \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}$$

- By Rule **EQUAL FUNCTORS**, we must have $t = f(u_1, u_2, u_3)$ since:

$$\begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix} \vdash \begin{pmatrix} f(a, g(b,a), b) \\ f(b, g(a,b), a) \end{pmatrix} f(u_1, u_2, u_3) \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix} \quad \text{where:}$$

– u_1 is the generalization of $\begin{pmatrix} a \\ b \end{pmatrix} \uparrow \begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix}$; that is of a and b; and by Rule **UNEQUAL FUNCTORS**:

$$\begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix} \vdash \begin{pmatrix} a \\ b \end{pmatrix} X \begin{pmatrix} \{\, a/X \,\} \\ \{\, b/X \,\} \end{pmatrix} \quad \text{and therefore} \quad u_1 = X;$$

– $u_2 = g(v_1, v_2)$ is the generalization of $\begin{pmatrix} g(b,a) \\ g(a,b) \end{pmatrix} \uparrow \begin{pmatrix} \{\, a/X \,\} \\ \{\, b/X \,\} \end{pmatrix}$; that is, of $g(b,X)$ and $g(a,X)$; and by Rule **EQUAL FUNCTORS**:

∗ v_1 is the generalization of $\begin{pmatrix} b \\ a \end{pmatrix} \uparrow \begin{pmatrix} \{\, a/X \,\} \\ \{\, b/X \,\} \end{pmatrix}$; that is, of b and a; and by Rule **UNEQUAL FUNCTORS**:

$$\begin{pmatrix} \{\, a/X \,\} \\ \{\, b/X \,\} \end{pmatrix} \vdash \begin{pmatrix} b \\ a \end{pmatrix} Y \begin{pmatrix} \{\, a/X, b/Y \,\} \\ \{\, b/X, a/Y \,\} \end{pmatrix} \quad \text{so} \quad v_1 = Y;$$

* v_2 is the generalization of $\binom{a}{b} \uparrow \binom{\{\, a/X, b/Y \,\}}{\{\, b/X, a/Y \,\}}$; that is, of X and X; and by Rule **EQUAL VARIABLES**:

$$\binom{\{\, a/X, b/Y \,\}}{\{\, b/X, a/Y \,\}} \vdash \binom{X}{X} X \binom{\{\, a/X, b/Y \,\}}{\{\, b/X, a/Y \,\}} \text{ so } v_2 = X;$$

therefore:

$$\binom{\{\, a/X \,\}}{\{\, b/X \,\}} \vdash \binom{g(b,X)}{g(a,X)} g(Y,X) \binom{\{\, a/X, b/Y \,\}}{\{\, b/X, a/Y \,\}} \text{ so } u_2 = g(Y,X);$$

– u_3 is the generalization of $\binom{b}{a} \uparrow \binom{\{\, a/X, b/Y \,\}}{\{\, b/X, a/Y \,\}}$; that is, of Y and Y; and by Rule **EQUAL VARIABLES**:

$$\binom{\{\, a/X, b/Y \,\}}{\{\, b/X, a/Y \,\}} \vdash \binom{Y}{Y} Y \binom{\{\, a/X, b/Y \,\}}{\{\, b/X, a/Y \,\}} \text{ so } u_3 = Y;$$

- therefore, this yields:

$$\binom{\emptyset}{\emptyset} \vdash \binom{f(a, g(b,a), b)}{f(b, g(a,b), a)} f(X, g(Y,X), Y) \binom{\{\, a/X, b/Y \,\}}{\{\, b/X, a/Y \,\}}$$

that is $t = f(X, g(Y,X), Y)$ with $\sigma_1 = \{\, a/X, b/Y \,\}$ such that $t\sigma_1 = f(a, g(b,a), b)$, and $\sigma_2 = \{\, b/X, a/Y \,\}$ such that $t\sigma_2 = f(b, g(a,b), a)$.

Example 2.5 Generalization of non-ground $\mathcal{FOT}$s — Let us apply the $\mathcal{FOT}$ generalization axioms and rules of Figure 2.5 to the following $\mathcal{FOT}$s:

$$t_1 \stackrel{\text{def}}{=} h(f(a, X_1), g(X_1, b), f(Y_1, Y_1)), \text{ and } t_2 \stackrel{\text{def}}{=} h(X_2, X_2, g(c,d)).$$

- Let us find term t and substitutions σ_1 and σ_2 such that $t\sigma_1 = h(f(a, X_1), g(X_1, b), f(Y_1, Y_1))$ and $t\sigma_2 = h(X_2, X_2, g(c,d))$; that is, let us try to solve the constraint problem:

$$\binom{\emptyset}{\emptyset} \vdash \binom{h(f(a, X_1), g(X_1, b), f(Y_1, Y_1))}{h(X_2, X_2, g(c,d))} t \binom{\sigma_1}{\sigma_2}.$$

- By Rule **EQUAL FUNCTORS**, we must have $t = h(u_1, u_2, u_3)$ since:

$$\binom{\emptyset}{\emptyset} \vdash \binom{h(f(a, X_1), g(X_1, b), f(Y_1, Y_1))}{h(X_2, X_2, g(c,d))} h(u_1, u_2, u_3) \binom{\sigma_1}{\sigma_2} \text{ where:}$$

– u_1 is the generalization of $\binom{f(a, X_1)}{X_2} \uparrow \binom{\emptyset}{\emptyset}$; that is of $f(a, X_1)$ and X_2; and by Rule **VARIABLE-TERM**:

$$\binom{\emptyset}{\emptyset} \vdash \binom{f(a, X_1)}{X_2} X \binom{\{\, f(a, X_1)/X \,\}}{\{\, X_2/X \,\}} \text{ so } u_1 = X;$$

– u_2 is the generalization of $\begin{pmatrix} g(X_1,b) \\ X_2 \end{pmatrix} \uparrow \begin{pmatrix} \{\, f(a,X_1)/X \,\} \\ \{\, X_2/X \,\} \end{pmatrix}$; that is, of $g(X_1,b)$ and X_2; and by Rule **VARIABLE-TERM**:

$$\begin{pmatrix} \{\, f(a,X_1)/X \,\} \\ \{\, X_2/X \,\} \end{pmatrix} \vdash \begin{pmatrix} g(X_1,b) \\ X_2 \end{pmatrix} Y \begin{pmatrix} \{\ldots, g(X_1,b)/Y \,\} \\ \{\ldots, X_2/Y \,\} \end{pmatrix} \text{ so } u_2 = Y;$$

– u_3 is the generalization of $\begin{pmatrix} f(Y_1,Y_1) \\ g(c,d) \end{pmatrix} \uparrow \begin{pmatrix} \{\, f(a,X_1)/X, g(X_1,b)/Y \,\} \\ \{\, X_2/X, X_2/Y \,\} \end{pmatrix}$; that is, of $f(Y_1,Y_1)$ and $g(c,d)$; and by Rule **UNEQUAL FUNCTORS**:

$$\begin{pmatrix} \{\, f(a,X_1)/X, g(X_1,b)/Y \,\} \\ \{\, X_2/X, X_2/Y \,\} \end{pmatrix} \vdash \begin{pmatrix} f(Y_1,Y_1) \\ g(c,d) \end{pmatrix} Z \begin{pmatrix} \{\ldots, f(Y_1,Y_1)/Z \,\} \\ \{\ldots, g(c,d)/Z \,\} \end{pmatrix}$$

and so $u_3 = Z$;

- therefore, this yields:

$$\begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix} \vdash \begin{pmatrix} h(f(a,X_1), g(X_1,b), f(Y_1,Y_1)) \\ h(X_2,X_2,g(c,d)) \end{pmatrix} h(X,Y,Z) \begin{pmatrix} \{\ldots, f(Y_1,Y_1)/Z \,\} \\ \{\ldots, g(c,d)/Z \,\} \end{pmatrix}$$

that is, $t = h(X,Y,Z)$ with $\sigma_1 = \{\, f(a,X_1)/X,\ g(X_1,b)/Y,\ f(Y_1,Y_1)/Z \,\}$ s.t. $t\sigma_1 = h(f(a, X_1), g(X_1,b), f(Y_1,Y_1))$, and: $\sigma_2 = \{\, X_2/X,\ X_2/Y,\ g(c,d)/Z \,\}$ s.t. $t\sigma_2 = h(X_2, X_2, g(c,d))$.

2.6 Fuzzy Lattice Operations on $\mathcal{FOT}$s

For the formal Fuzzy Algebra notation and terminology that we use in the remainder of this work, see A.3.

2.6.1 Fuzzy $\mathcal{FOT}$ unification

Sessa's weak unification

A fuzzy unification operation on $\mathcal{FOT}$s, dubbed "*weak unification*," was proposed by Maria Sessa in [117] which consists in normalizing fuzzy equations between conventional $\mathcal{FOT}$s modulo a similarity relation $\sim$ over functor symbols [56]. This similarity relation is then homomorphically extended to one over all $\mathcal{FOT}$s.

Example 2.6 Functor similarity matrix — Given a similarity (*i.e.*, a fuzzy equivalence) relation $\sim$ on a finite signature $\Sigma = \cup_n \Sigma_n$, as explained in [56], this information is represented as a matrix in $\Sigma \times \Sigma \to [0,1]$. For example, if the signature Σ is the union of $\Sigma_0 = \{\, a,b,c,d \,\}$, $\Sigma_2 = \{\, f,g \,\}$, $\Sigma_3 = \{\, h \,\}$, and $\Sigma_n = \emptyset$ otherwise ($n = 1$ or $n \geq 4$), and with a similarity that is the reflexive, symmetric,

and transitive closure of the pairs $a \sim_{0.7} b$, $c \sim_{0.6} d$, and $f \sim_{0.9} g$. This corresponds to the similarity matrix whose rows and columns are indexed by elements of Σ:

$$\sim \stackrel{\text{def}}{=} \begin{array}{c|ccccccc} & a & b & c & d & f & g & h \\ \hline a & 1 & 0.7 & 0 & 0 & 0 & 0 & 0 \\ b & 0.7 & 1 & 0 & 0 & 0 & 0 & 0 \\ c & 0 & 0 & 1 & 0.6 & 0 & 0 & 0 \\ d & 0 & 0 & 0.6 & 1 & 0 & 0 & 0 \\ f & 0 & 0 & 0 & 0 & 1 & 0.9 & 0 \\ g & 0 & 0 & 0 & 0 & 0.9 & 1 & 0 \\ h & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

Following Maria Sessa's formal setting [117], we assume given such a similarity relation between functors of equal arity (*i.e.*, which admit the same number of arguments). Upon this basis, this similarity can be extended homomorphically from functors to $\mathcal{FOT}$s as follows. Let $\sim$ be a similarity on functors of equal arity in a signature Σ.

DEFINITION 2.5 (SESSA'S $\mathcal{FOT}$ SIMILARITY) *The fuzzy relation $\sim^{\mathcal{T}}$ on $\mathcal{T}_{\Sigma,\mathcal{V}}$ is defined inductively as follows:*

1. $\forall X \in \mathcal{V},\ \ X \sim_1^{\mathcal{T}} X$;
2. $\forall X \in \mathcal{V}, \forall t \in \mathcal{T}$ *such that* $X \neq t$, $X \sim_0^{\mathcal{T}} t$ *and* $t \sim_0^{\mathcal{T}} X$;
3. *if* $f \in \Sigma_n$ *and* $g \in \Sigma_n$ *with* $f \sim_\alpha g$, *and if* $s_i \in \mathcal{T}$ *and* $t_i \in \mathcal{T}$ *are such that* $s_i \sim_{\alpha_i}^{\mathcal{T}} t_i$ *for* $i = 1, \ldots, n$, *then:*

$$f(s_1, \ldots, s_n) \sim^{\mathcal{T}}_{\alpha \wedge \bigwedge_{i=1}^n \alpha_i} g(t_1, \ldots, t_n). \tag{2.11}$$

THEOREM 2.2 *The relation $\sim^{\mathcal{T}}$ defined by Definition 2.5 is a similarity relation on the set of $\mathcal{FOT}$s $\mathcal{T}_{\Sigma,\mathcal{V}}$.*

> **PROOF** See proof of more general Theorem 2.3 below, as this is a particular case of that theorem where every similar pairs of functors have same arity and every argument position mapping is the identity. □

Since from the above definition of similarity $\sim^{\mathcal{T}}$ extends homomorphically a similarity $\sim$ on the functors to all $\mathcal{FOT}$s in $\mathcal{T}$, we shall also assimilate $\sim^{\mathcal{T}}$ to $\sim$. This allows to define formally fuzzy subsumption among $\mathcal{FOT}$s as the fuzzy relation $\preceq$ on $\mathcal{T}$ that can be verified to be a preorder (modulo variable renaming) as a corollary of Theorem 2.2.

DEFINITION 2.6 (FUZZY $\mathcal{FOT}$ SUBSUMPTION) *For all terms $t_1 \in \mathcal{T}$ and $t_2 \in \mathcal{T}$, t_1 is said to be subsumed by t_2 for some α in $[0,1]$ (and this is written $t_1 \preceq_\alpha t_2$) if and only if there exists a substitution $\sigma \in$ **SUBST**$_{\mathcal{T}}$ such that $t_1 \sim_\alpha t_2\sigma$.*

Note that, for the identity similarity on the signature and $\alpha = 1$, this reduces to the classical definition of term subsumption, as expected.

In Definition 2.6, the more specific term t_1 is then called a fuzzy instance of term t_2 realized with subsititution σ at approximation degree α. It comes also that the "more general" relation on $\mathcal{FOT}$ substitutions extends to a "fuzzy more general" fuzzy relation on substitutions, which can also readily be verified to be a fuzzy preorder on $\mathbf{SUBST}_{\mathcal{T}}$ as a corollary of Theorem 2.2. It is formally equivalent to the relation defined in [117].[12]

DEFINITION 2.7 (FUZZY "MORE GENERAL" ORDERING ON $\mathcal{FOT}$ SUBSTITUTIONS) *If σ_1 and σ_2 are two substitutions in $\mathbf{SUBST}_{\mathcal{T}}$ and α in $[0,1]$, we say that σ_1 is less general than σ_2 at approximation degree α (and this is written $\sigma_1 \preceq_\alpha \sigma_2$), if and only if for any term $t \in \mathcal{T}$, it is true that $t\sigma_1 \preceq_\alpha t\sigma_2$ as terms.*

Also as expected, note that for the identity similarity on the signature and $\alpha = 1$, this reduces to the classical "more general than" ordering on substitutions.

The following fuzzy relation defined on $\mathbf{SUBST}_{\mathcal{T}}$ can also be verified to be a similarity as a corollary of Theorem 2.2.[13]

DEFINITION 2.8 ($\mathcal{FOT}$ SUBSTITUTION SIMILARITY) *Given an approximation degree α in $[0,1]$, two substitutions σ and θ in $\mathbf{SUBST}_{\mathcal{T}}$ are said to be α-similar (written $\sigma \sim_\alpha \theta$) iff $t\sigma \sim_\alpha t\theta$ for all $\mathcal{FOT}$ t in $\mathcal{T}$.*

Therefore, referring to Definition 2.6 of fuzzy $\mathcal{FOT}$ subsumption, it comes as a fact that:

LEMMA 2.3 *For any two substitutions σ and θ in $\mathbf{SUBST}_{\mathcal{T}}$ and approximation degree α in $[0,1]$, $\sigma \preceq_\alpha \theta$ iff $\sigma \sim_\alpha \theta\delta$ for some substitution δ.*

> **PROOF** Stating that $\sigma \preceq_\alpha \theta$, by Definition 2.7, is equivalent to stating that $t\sigma \preceq_\alpha t\theta$, for any $t \in \mathcal{T}$. By Definition 2.6, this is equivalent to stating that for all term t, $t\sigma \sim_\alpha t\theta\delta$, for some substitution δ; namely, again by Definition 2.7, that $\sigma \sim_\alpha \theta\delta$. □

The following two facts regarding the fuzzy term subsumption relation on terms and the fuzzy "more general" relation on substitutions will be useful later in a proof arguments.

LEMMA 2.4 *For any two approximation degrees α and β in $[0,1]$, for any terms t_1, t_2, and t_3 in $\mathcal{T}$, if $t_1 \preceq_\alpha t_2$ and $t_2 \preceq_\beta t_3$, then $t_1 \preceq_{\alpha\wedge\beta} t_3$.*

> **PROOF** Let $t_1 \preceq_\alpha t_2$ and $t_2 \preceq_\beta t_3$; this is, by definition, equivalent to $t_1 \sim_\alpha t_2\sigma$, for some $\sigma \in \mathbf{SUBST}_{\mathcal{T}}$, and $t_2 \sim_\beta t_3\theta$, for some $\theta \in \mathbf{SUBST}_{\mathcal{T}}$. However, for any set S, any pair $\langle x, y\rangle$ in $S \times S$, and any similarity $\sim$: $S \times S \to [0,1]$, if $x \sim_\alpha y$ for some α in$[0,1]$, then $x \sim_\beta y$ for all $\beta \in [0,\alpha]$.[14] This, the fact that $\alpha \wedge \beta \leq \alpha$ and $\alpha \wedge \beta \leq \beta$, together with our assumption, entail then that $t_1 \sim_{\alpha\wedge\beta} t_2\sigma$ and $t_2 \sim_{\alpha\wedge\beta} t_3\sigma$; which, by transitivity of $\sim_{\alpha\wedge\beta}$, implies that $t_1 \sim_{\alpha\wedge\beta} t_3\sigma$; that is, $t_1 \preceq_{\alpha\wedge\beta} t_3$. □

[12] *Op. cit.*, Page 410, Definition 6.2
[13] A equivalent definition is given in [75].
[14] See A.3.2.

COROLLARY 2.3 *For any two approximation degrees α and β in $[0,1]$, for any substitutions σ_1, σ_2, and σ_3 in* **SUBST** $_{\mathcal{T}}$*, if $\sigma_1 \preceq_\alpha \sigma_2$ and $\sigma_2 \preceq_\beta \sigma_3$, then $\sigma_1 \preceq_{\alpha\wedge\beta} \sigma_3$.*

PROOF Let $\sigma_1 \preceq_\alpha \sigma_2$ and $\sigma_2 \preceq_\beta \sigma_3$; this is, by definition, equivalent to stating that for any term $t \in \mathcal{T}$, $t\sigma_1 \preceq_\alpha t\sigma_2$ and $t\sigma_2 \preceq_\beta t\sigma_3$. By Lemma 2.4, it follows that for any term $t \in \mathcal{T}$, $t\sigma_1 \preceq_{\alpha\wedge\beta} t\sigma_3$; that is, $\sigma_1 \preceq_{\alpha\wedge\beta} \sigma_3$. □

Using the definition of similarity between terms in $\mathcal{T}$ extending one on functors of equal arity, Sessa proposes to extend the $\mathcal{FOT}$ unification problem to the following fuzzy unification problem: given two $\mathcal{FOT}$s t_1 and t_2 in $\mathcal{T}$, find the most general substitution $\sigma \in$ **SUBST** $_{\mathcal{T}}$ and maximum approximation degree α in $[0,1]$ such that $t_1\sigma \sim_\alpha t_2\sigma$. Figure 2.6 expresses fuzzy unification as a commutative diagram constraint.

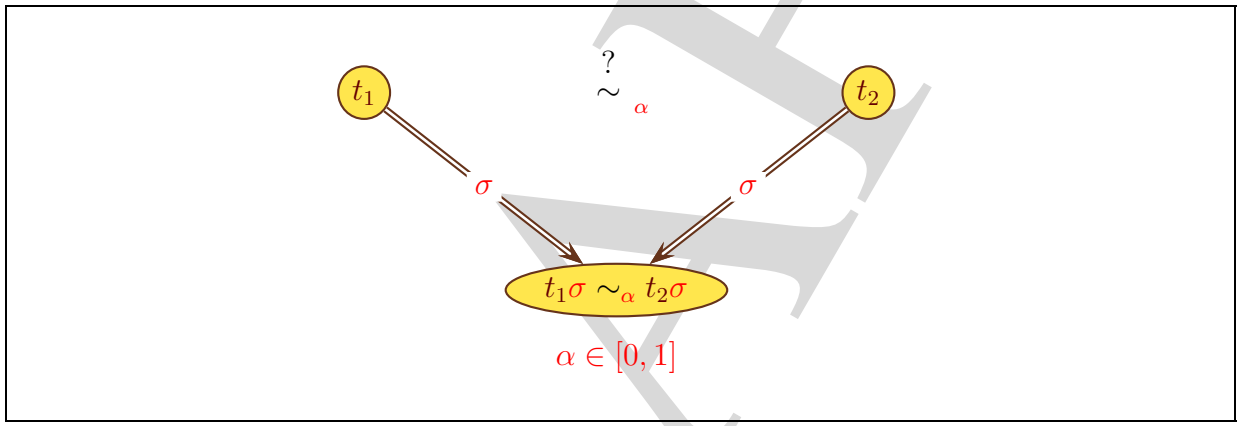


Figure 2.6: Fuzzy unification as a constraint

In Figure 2.7, we provide a set of declarative rewrite rules for fuzzy unfication equivalent to Sessa's case-based "weak unification algorithm" [117]. To simplify the presentation of these rules while remaining faithful to Sessa's weak unification algorithm, it is assumed for now that functor symbols f/m and g/n of different arities $m \neq n$ are never similar. This follows Sessa's assumption for weak unification, which fails on term structures of different arities. (See Case (2) of the weak unification algorithm given in [117], Page 413.) Later, we will relax this and allow functors of different arities to be similar.

The rules of Figure 2.7 transform E_α, a finite conjunctive set E of equations among $\mathcal{FOT}$s along with an associated approximation degree α in $[0,1]$, into $E'_{\alpha'}$, another set of equations E' at approximation degree α' in $[0,\alpha]$. Given to solve a fuzzy unification equation $s \doteq t$ between two $\mathcal{FOT}$s s and t, we start by forming the set $\{ s \doteq t \}_1$ (*i.e.*, a singleton equation set at approximation degree 1), then transform it using any applicable rules in Figure 2.7 until either the approximation degree of the transformed set of equations is 0 (in which case there is no solution to the original equation, not even a fuzzy one), or the final resulting set E_α is a solution at approximation degree α in the form of a variable substitution $\sigma \stackrel{\text{def}}{=} \{ t/X \mid X \doteq t \in E \}$ such that $s\sigma \sim_\alpha t\sigma$.

WEAK TERM DECOMPOSITION:

$$\frac{(E \cup \{\, f(s_1,\ldots,s_n) \doteq g(t_1,\ldots,t_n) \,\})_\alpha}{(E \cup \{\, s_1 \doteq t_1,\ldots,s_n \doteq t_n \,\})_{\alpha\wedge\beta}} \left[\begin{smallmatrix} f \sim_\beta g \\ n \geq 0 \end{smallmatrix}\right]$$

VARIABLE ERASURE:

$$\frac{(E \cup \{\, X \doteq X \,\})_\alpha}{E_\alpha}$$

VARIABLE ELIMINATION:

$$\frac{(E \cup \{\, X \doteq t \,\})_\alpha}{(E[X \leftarrow t] \cup \{\, X \doteq t \,\})_\alpha} \left[\begin{smallmatrix} X \notin \mathbf{var}(t) \\ X \text{ occurs in } E \end{smallmatrix}\right]$$

EQUATION ORIENTATION:

$$\frac{(E \cup \{\, t \doteq X \,\})_\alpha}{(E \cup \{\, X \doteq t \,\})_\alpha} [t \notin \mathcal{V}]$$

Figure 2.7: Normalization rules corresponding to Maria Sessa's "weak unification"

In [117],[15] a transformation rule of a set of equation at approximation degree is considered to be correct when all the solutions of the posterior set are also solutions of the anterior set but with a possibly lesser similarity degree, which is also our Definition 2.10.[16]

Example 2.7 *$\mathcal{FOT}$* **fuzzy unification** — Taking the functor signature of Example 2.6, let us consider the fuzzy equation set:

$$\{\, h(f(a,X_1),g(X_1,b),f(Y_1,Y_1)) \doteq h(X_2,X_2,g(c,d)) \,\}_1 \tag{2.12}$$

and let us apply the rules of Figure 2.7:

- Rule **WEAK TERM DECOMPOSITION** with $\alpha = 1$ and $\beta = 1$:

  $$\{\, f(a,X_1) \doteq X_2, g(X_1,b) \doteq X_2, f(Y_1,Y_1) \doteq g(c,d) \,\}_1 ;$$

- Rule **EQUATION ORIENTATION** to $f(a,X_1) \doteq X_2$ with $\alpha = 1$:

  $$\{\, X_2 \doteq f(a,X_1), g(X_1,b) \doteq X_2, f(Y_1,Y_1) \doteq g(c,d) \,\}_1 ;$$

- Rule **VARIABLE ELIMINATION** to $X_2 \doteq f(a,X_1)$ with $\alpha = 1$:

  $$\{\, X_2 \doteq f(a,X_1), g(X_1,b) \doteq f(a,X_1), f(Y_1,Y_1) \doteq g(c,d) \,\}_1 ;$$

- Rule **WEAK TERM DECOMPOSITION** to $g(X_1,b) \doteq f(a,X_1)$ with $\alpha = 1$ and $\beta = .9$:

  $$\{\, X_2 \doteq f(a,X_1), X_1 \doteq a, b \doteq X_1, f(Y_1,Y_1) \doteq g(c,d) \,\}_{.9} ;$$

- Rule **VARIABLE ELIMINATION** to $X_1 \doteq a$ with $\alpha = .9$:

  $$\{\, X_2 \doteq f(a,a), X_1 \doteq a, b \doteq a, f(Y_1,Y_1) \doteq g(c,d) \,\}_{.9} ;$$

[15] *Op. cit.*, Page 410.

[16] Note that in [117], no explicit proof for of formal correctness of "weak unification algorithm" is given: it is just mentioned that "it can be proven following the same line of the proof" for crisp unification in classible Logic Programmingin [27].

- Rule **WEAK TERM DECOMPOSITION** to $b \doteq a$ with $\alpha = .9$ and $\beta = .7$:

$$\{ X_2 \doteq f(a,a), X_1 \doteq a, f(Y_1, Y_1) \doteq g(c,d) \}_{.7};$$

- Rule **WEAK TERM DECOMPOSITION** to $f(Y_1, Y_1) \doteq g(c,d)$ with $\alpha = .7$ and $\beta = .9$:

$$\{ X_2 \doteq f(a,a), X_1 \doteq a, Y_1 \doteq c, Y_1 \doteq d \}_{.7};$$

- Rule **VARIABLE ELIMINATION** to $Y_1 \doteq c$ with $\alpha = .7$:

$$\{ X_2 \doteq f(a,a), X_1 \doteq a, Y_1 \doteq c, c \doteq d \}_{.7};$$

- Rule **WEAK TERM DECOMPOSITION** to $c \doteq d$ with $\alpha = .7$ and $\beta = .6$:

$$\{ X_2 \doteq f(a,a), X_1 \doteq a, Y_1 \doteq c \}_{.6}.$$

This last equation set is in normal form with similarity degree .6 and defines the substitution σ given by:

$$\sigma = \{ a/X_1, c/Y_1, f(a,a)/X_2 \} \tag{2.13}$$

so that:

$$t_1\sigma = h(f(a,a), g(a,b), f(c,c)) \sim_{.6} h(f(a,a), f(a,a), g(c,d)) = t_2\sigma. \tag{2.14}$$

Generic fuzzy $\mathcal{FOT}$ unification

From our perspective, a fuzzy unification operation ought to be able to fuzzify *full* $\mathcal{FOT}$ unification: whether (1) functor symbol mistmatch, and/or (2) arity mismatch, and/or (3) in which order subterms correspond. Sessa's fuzzification of unification as weak unification misses on the last two items. This is unfortunate as this can turn out to be quite useful. In real life, there is indeed no such garantee that argument positions of different functors match similar information in data and knowledge bases, hence the need for alignment [86].

Still, it has several qualities:

- *It is simple*—specified as a straightforward extension of crisp unification: only one rule (Rule "**FUZZY TERM DECOMPOSITION**") may alter the fuzziness of an equation set by tolerating similar functors.
- *It is conservative*—neither $\mathcal{FOT}$s nor $\mathcal{FOT}$ substitutions *per se* need be fuzzified; so conventional crisp representations and operations can be used; if restricted to only 0 or 1 similarity degrees, it is equivalent to crisp $\mathcal{FOT}$ unification.

We now give an extension of Sessa's weak unification which can tolerate such similarity among functors of different arities. We are given a similarity relation $\approx : \Sigma \times \Sigma \to [0,1]$ on a ranked signature $\Sigma \stackrel{\text{def}}{=} \biguplus_{n \geq 0} \Sigma_n$. Unlike M. Sessa's equal-arity condition, we now allow similar symbols with distinct arities, or equal arities but different argument orders.

Example 2.8 Similar functors with different arities — Consider `person`/3, a functor of arity 3, and `individual`/4, a functor of arity 4 with:

- $\texttt{person}/3 \approx_{.9} \texttt{individual}/4$; and,
- one-to-one position mapping $p : \{1, 2, 3\} \to \{1, 2, 3, 4\}$:

  from $\texttt{person}/3$ to $\texttt{individual}/4$ with $p : \{1 \to 1, 2 \to 3, 3 \to 4\}$

  so that:

$$\texttt{person}(Name, SSN, Address) \approx^{p}_{.9} \texttt{individual}(Name, DoB, SSN, Address)$$

where we write $f \approx^{p}_{\alpha} g$ to denote a pair in the similarity relation $\approx$ consisting of a functor f and a functor g, with similarity degree α and f-to-g argument-position mapping p; in our example, this is rendered as: $\texttt{person} \approx^{\{1\to1,2\to3,3\to4\}}_{.9} \texttt{individual}$.

With this kind of specification, we can tolerate not only fuzzy mismatching of terms with distinct functors `person` and `individual` up to a realigning correspondence of argument positions from `person` to `individual` specified as p, all with a similarity degree of .9.

We formalize this by requiring that the fuzzy equivalence relation $\approx$ on Σ be such that:

- for each pair of functors $\langle f, g\rangle \in \Sigma_m \times \Sigma_n$ where $0 \leq m \leq n$ and $f \approx g$, and any approximation degree α, there exists a one-to-one (*i.e.*, injective) mapping $\mu^{\alpha}_{fg} : \{1, \ldots, m\} \to \{1, \ldots, n\}$ associating each of the m argument positions of f with a unique position among the n arguments of g, which we shall express as $f \approx^{\mu^{\alpha}_{fg}} g$;
- argument-position alignment mappings between similar functors must be *consistent* at any approximation level; namely, they must verify the following four conditions:
  - *approximation consistency*: for any functors $f \in \Sigma$ and $g \in \Sigma$, and any approximation degrees $\alpha \in [0, 1]$ and $\beta \in [0, 1]$:

$$\alpha \leq \beta \implies \mu^{\alpha}_{fg} \subseteq \mu^{\beta}_{fg} \quad \text{(as sets of pairs)}; \tag{2.15}$$

  - *reflexive consistency*: for any functor f/n and any degree $\alpha \in [0, 1]$:

$$\mu^{\alpha}_{ff} = \mathbb{1}_{\{1,\ldots,n\}}; \tag{2.16}$$

  - *symmetric consistency*: for any two equal-arity functors f/n and g/n and any degree $\alpha \in [0, 1]$:

$$\mu^{\alpha}_{fg} \circ \mu^{\alpha}_{gf} = \mathbb{1}_{\{1,\ldots,n\}}; \tag{2.17}$$

  - *transitive consistency*: for any three functors f/m, g/n, h/ℓ s.t. $0 \leq m \leq n \leq \ell$ and any degree $\alpha \in [0, 1]$:

$$\mu^{\alpha}_{fh} = \mu^{\alpha}_{gh} \circ \mu^{\alpha}_{fg}. \tag{2.18}$$

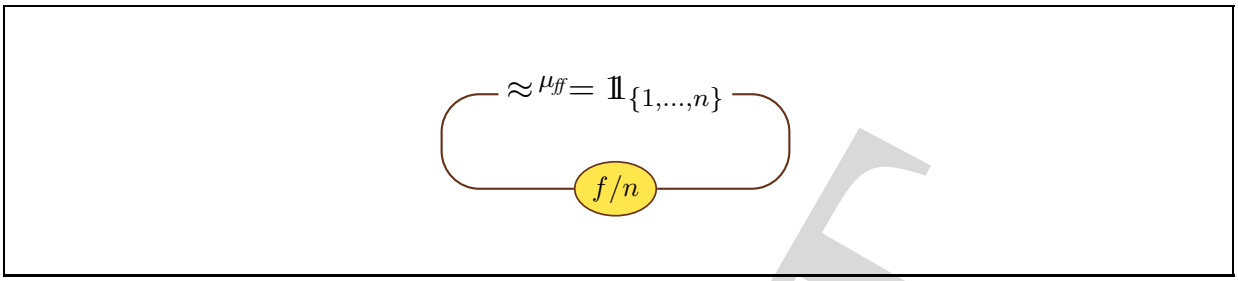


Figure 2.8: Identity consistency for $\mathcal{FOT}$ argument mapping

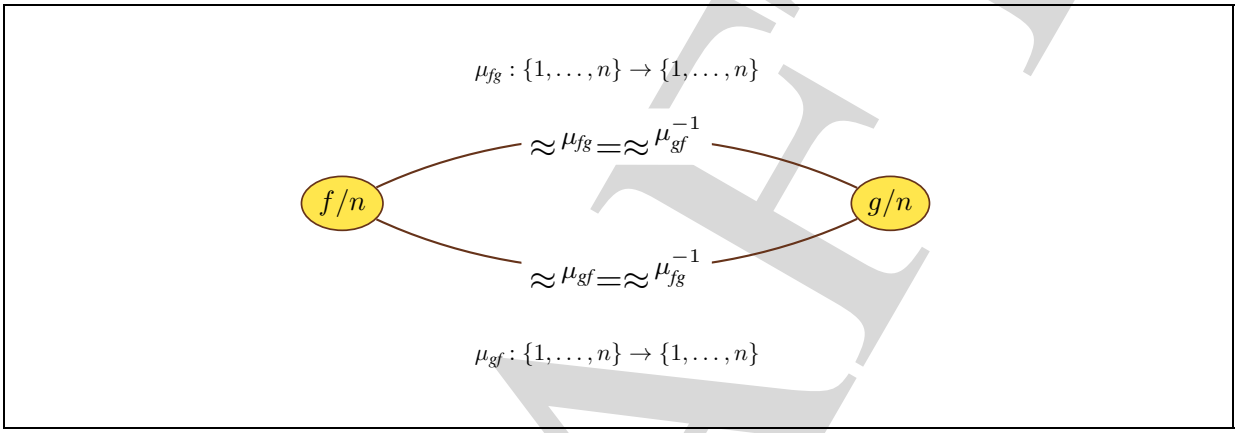


Figure 2.9: Invertibility consistency for equal-arity $\mathcal{FOT}$ argument mapping

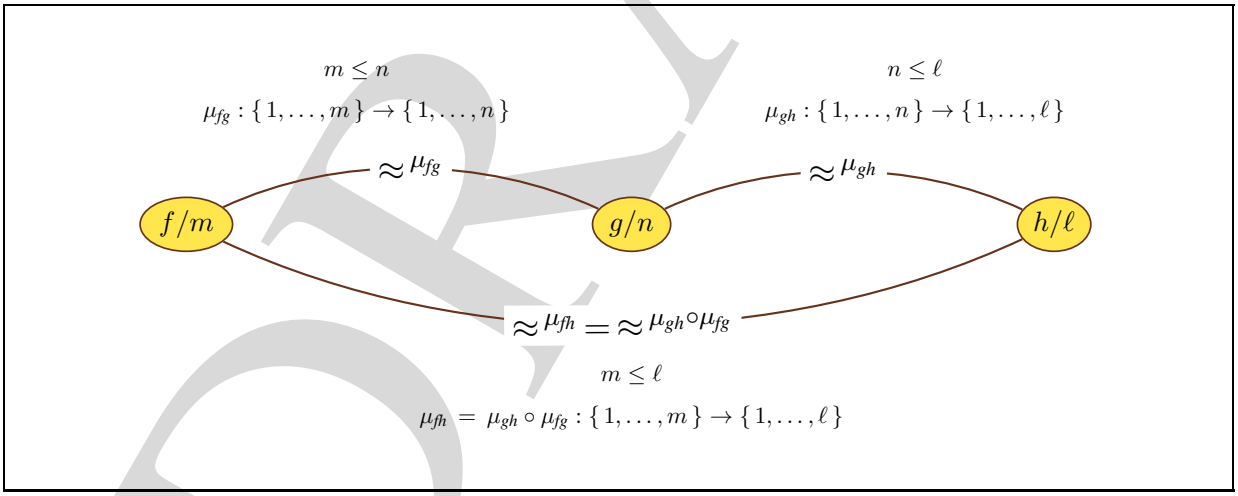


Figure 2.10: Compositional consistency for non-aligned $\mathcal{FOT}$ argument mapping

Figure 2.8 illustrates Condition (2.16), Figure 2.9 illustrates Condition (2.17), and Figure 2.10 illustrates Condition (2.18). Note that Condition (2.18) applies when $0 \leq m \leq n \leq \ell$; so the one-to-one argument-position mappings always go from a smaller set to a larger set. There is no loss of generality with this assumption as this will be taken into account in the definition of

non-aligned $\mathcal{FOT}$ similarity,[17] and in the normalization rules.[18] This amounts to systematically taking a $\mathcal{FOT}$ with functor of least arity as similarity class representative. Finally, note also that such a class representative is not unique because for similar functors of equal arity, it can be either terms due to Condition (2.17). Indeed, then the set of positions are equal and there are two injections from this set to itself in each direction which are mutually inverse bijections; *i.e.*, inverse permutations in the order of arguments realigning one's with the other's in either direction. The similarity degrees in both directions are always equal due to symmetry of similarity.

Fuzzy unification with similar functors and arity mismatch

As in the case of similarity restricted to functors of equal arities only, the similarity with argument position alignment mapping on functors can be extended homomorphically to a similarity on $\mathcal{FOT}$s. Let $\approx$ be a similarity on functors of any arity in a signature Σ. To lighten notation, rather than writing systematically $f \approx^{\mu_{fg}} g$ for two functors f and g such that $\textbf{arity}(f) \leq \textbf{arity}(g)$, we shall sometimes simply write $f \approx^{p}_{\alpha} g$, with p standing for the injective argument realignment mapping μ_{fg}.

DEFINITION 2.9 *The fuzzy relation $\approx^{\mathcal{T}}$ on $\mathcal{T}_{\Sigma,\mathcal{V}}$ is defined inductively as:*

1. $\forall X \in \mathcal{V},\ \ X \approx^{\mathcal{T}}_{1} X$;
2. $\forall X \in \mathcal{V}, \forall t \in \mathcal{T}$ *such that* $X \neq t$, $X \sim^{\mathcal{T}}_{0} t$ *and* $t \sim^{\mathcal{T}}_{0} X$;
3. *if* $s = f(s_1, \ldots, s_m)$ *and* $t = g(t_1, \ldots, t_n)$ *with* $n < m$, *then* $s \approx^{\mathcal{T}} t = t \approx^{\mathcal{T}} s$;
4. *if* $f \in \Sigma_m$ *and* $g \in \Sigma_n$ *with* $m \leq n$ *and* $f \approx^{p}_{\alpha} g$, *and if* $s_i \in \mathcal{T}$, $i = 1, \ldots, m$, *and* $t_j \in \mathcal{T}$, $j = 1, \ldots, n$, *are such that* $s_i \approx^{\mathcal{T}}_{\alpha_i} t_{p(i)}$ *for all* $i \in \{1, \ldots, m\}$, *then:*

$$f(s_1, \ldots, s_m) \approx^{\mathcal{T}}_{\alpha \wedge \bigwedge_{i=1}^{m} \alpha_i} g(t_1, \ldots, t_n). \tag{2.19}$$

THEOREM 2.3 (NON-ALIGNED $\mathcal{FOT}$ SIMILARITY) *The fuzzy relation $\approx^{\mathcal{T}}$ on the set $\mathcal{T}$ of $\mathcal{FOT}$s specified in Definition 2.9 is a similarity.*

PROOF We must establish that $\approx^{\mathcal{T}}$ is reflexive, symmetric, and transitive.

Reflexivity: we must show that $t \approx^{\mathcal{T}}_{1} t$, for all $t \in \mathcal{T}$. We proceed by induction on the depth of the term. *Base case:* either $t = X \in \mathcal{V}$, in which case, by the first condition of Definition 2.9, $X \approx^{\mathcal{T}}_{1} X$; or, $t = c \in \Sigma_0$, in which case the fourth condition of Definition 2.9 and the fact that $c \approx_1 c$ implies that $c \approx^{\mathcal{T}}_{1} c$, for all $c \in \Sigma_0$. *Inductive case:* let us assume that $\approx^{\mathcal{T}}$ is reflexive for all terms of depth less than or equal to d, and consider the term $t = f(t_1, \ldots, t_n)$ of depth $d + 1$; then, the fourth condition of Definition 2.9 implies also that $t \approx^{\mathcal{T}} t$ since, by Condition (2.16) and the fact that $\approx$ is a similarity, $f \approx^{\mathbb{1}_{\{1,\ldots,n\}}}_{1} f$ for all $f \in \Sigma_n$, for any arity $n > 0$.

Symmetry: we must show that $s \approx^{\mathcal{T}} t = t \approx^{\mathcal{T}} s$ for all s and t in $\mathcal{T}$. When either of the terms is a variable, this is so by the two first cases of Definition 2.9. When $s = f(s_1, \ldots, s_m)$

[17] *Cf.*, Definition 2.9 below.
[18] *Cf.*, Figure 2.11 below, Rule **FUZZY EQUATION ORIENTATION**.

and $t = g(t_1, \ldots, t_n)$, it is is always the case that $\approx^{\mathcal{T}}$ is symmetric on such pairs since the third condition of Definition 2.9, states precisely that in this case $\approx^{\mathcal{T}}$ is symmetric.

Transitivity: we must show that $(s \approx^{\mathcal{T}} t \wedge t \approx^{\mathcal{T}} u) \leq s \approx^{\mathcal{T}} u$ for all terms s, t, u. There are eight possibilities:

(1) $s \in \mathcal{V}$ and $t \in \mathcal{V}$ and $u \in \mathcal{V}$;
(2) $s \in \mathcal{V}$ and $t \in \mathcal{V}$ and $u \notin \mathcal{V}$;
(3) $s \in \mathcal{V}$ and $t \notin \mathcal{V}$ and $u \in \mathcal{V}$;
(4) $s \in \mathcal{V}$ and $t \notin \mathcal{V}$ and $u \notin \mathcal{V}$;
(5) $s \notin \mathcal{V}$ and $t \in \mathcal{V}$ and $u \in \mathcal{V}$;
(6) $s \notin \mathcal{V}$ and $t \in \mathcal{V}$ and $u \notin \mathcal{V}$;
(7) $s \notin \mathcal{V}$ and $t \notin \mathcal{V}$ and $u \in \mathcal{V}$;
(8) $s \notin \mathcal{V}$ and $t \notin \mathcal{V}$ and $u \notin \mathcal{V}$.

– Case (1): $s \in \mathcal{V}, t \in \mathcal{V}, u \in \mathcal{V}$. In this case, there are five possiblities. Using different variable names to denote different variables, the corresponding similarity degrees for $s \approx^{\mathcal{T}} t$, $t \approx^{\mathcal{T}} u$, and $s \approx^{\mathcal{T}} u$, for each possibility do indeed verify the inequality. Namely:

s	t	u	$s \approx^{\mathcal{T}} t \wedge t \approx^{\mathcal{T}} u \leq s \approx^{\mathcal{T}} u$
X	Y	Z	$0 \wedge 0 \leq 0$
X	Y	Y	$0 \wedge 1 \leq 1$
X	Y	X	$0 \wedge 0 \leq 1$
X	X	Y	$1 \wedge 0 \leq 0$
X	X	X	$1 \wedge 1 \leq 1$

– Case (2): $s \in \mathcal{V}, t \in \mathcal{V}, u \notin \mathcal{V}$. There are two possibilities, each verifying the inequality:

s	t	u	$s \approx^{\mathcal{T}} t \wedge t \approx^{\mathcal{T}} u \leq s \approx^{\mathcal{T}} u$
X	X	u	$1 \wedge 0 \leq 0$
X	Y	u	$0 \wedge 0 \leq 0$

– Case (3): $s \in \mathcal{V}, t \notin \mathcal{V}, u \in \mathcal{V}$. There are two possibilities, and each verifies the inequality:

s	t	u	$s \approx^{\mathcal{T}} t \wedge t \approx^{\mathcal{T}} u \leq s \approx^{\mathcal{T}} u$
X	t	X	$0 \wedge 0 \leq 1$
X	t	Y	$0 \wedge 0 \leq 0$

– Case (4): $s \in \mathcal{V}, t \notin \mathcal{V}, u \notin \mathcal{V}$. There is only one possibility, for any $\alpha \in [0, 1]$, which verifies the inequality:

s	t	u	$s \approx^{\mathcal{T}} t \wedge t \approx^{\mathcal{T}} u \leq s \approx^{\mathcal{T}} u$
X	t	u	$0 \wedge \alpha \leq 0$

– Case (5): $s \notin \mathcal{V}, t \in \mathcal{V}, u \in \mathcal{V}$. There are two possibilities, and each verifies the inequality:

s	t	u	$s \approx^{\mathcal{T}} t \wedge t \approx^{\mathcal{T}} u \leq s \approx^{\mathcal{T}} u$
s	X	X	$0 \wedge 1 \leq 0$
s	X	Y	$0 \wedge 0 \leq 0$

– Case (6): $s \notin \mathcal{V}, t \in \mathcal{V}, u \notin \mathcal{V}$. There is only one possibility, for any $\alpha \in [0, 1]$, which verifies the inequality:

s	t	u	$s \approx^{\mathcal{T}} t \wedge t \approx^{\mathcal{T}} u \leq s \approx^{\mathcal{T}} u$
s	X	u	$0 \wedge 0 \leq \alpha$

– Case (7) : $s \notin \mathcal{V}, t \notin \mathcal{V}, u \in \mathcal{V}$. There is only one possibility, for any $\alpha \in [0,1]$, which verifies the inequality:

$$\begin{array}{cccccccc} s & t & u & s \approx^{\mathcal{T}} t \wedge t \approx^{\mathcal{T}} u & \leq & s \approx^{\mathcal{T}} u \\ \hline s & t & X & \alpha & \wedge & 0 & \leq & 0 \end{array}$$

– Case (8): $s \notin \mathcal{V}, t \notin \mathcal{V}, u \notin \mathcal{V}$. In this case, we must have $s = f(s_1, \ldots, s_m)$, $t = g(t_1, \ldots, t_n)$, and $u = h(u_1, \ldots, u_\ell)$. We detail this case below.

We must then show that:

$$\begin{array}{c} f(s_1, \ldots, s_m) \approx^{\mathcal{T}} g(t_1, \ldots, t_n) \ \wedge \ g(t_1, \ldots, t_n) \approx^{\mathcal{T}} h(u_1, \ldots, u_\ell) \\ \leq \\ f(s_1, \ldots, s_m) \approx^{\mathcal{T}} h(u_1, \ldots, u_\ell). \end{array}$$

By symmetry of $\approx^{\mathcal{T}}$, all cases are equivalent to when $0 \leq m \leq n \leq \ell$, so we assume that this is so, with $f \approx_\alpha^{\mu_{fg}} g$ and $g \approx_\beta^{\mu_{gh}} h$. By the fourth condition of Definition 2.9, the above inequality is the same as the following one:

$$\begin{array}{c} f \approx g \ \wedge \ \bigwedge_{i=1}^m s_i \approx^{\mathcal{T}} t_{\mu_{fg}(i)} \ \wedge \ g \approx h \ \wedge \ \bigwedge_{j=1}^n t_j \approx^{\mathcal{T}} u_{\mu_{gh}(j)} \\ \leq \\ f \approx h \ \wedge \ \bigwedge_{i=1}^m s_i \approx^{\mathcal{T}} u_{\mu_{fh}(i)}. \end{array}$$

Using commutativity of $\wedge$, let us rearrange the different factors of the conjunction in the lefthand-side of this inequality as:

$$\begin{array}{c} f \approx g \ \wedge \ g \approx h \ \wedge \ \left(\bigwedge_{i=1}^m s_i \approx^{\mathcal{T}} t_{\mu_{fg}(i)} \ \wedge \ t_{\mu_{fg}(i)} \approx^{\mathcal{T}} u_{\mu_{gh}(\mu_{fg}(i))}\right) \ \wedge \ \Delta \\ \leq \\ f \approx h \ \wedge \ \bigwedge_{i=1}^m s_i \approx^{\mathcal{T}} u_{\mu_{fh}(i)} \end{array}$$

where Δ stands for the remaining conjunction $\bigwedge_{j \in \{1,\ldots,n\}}^{j \notin \mathbf{ran}(\mu_{fg})} t_j \approx^{\mathcal{T}} u_{\mu_{gh}(j)}$. Let us now proceed by induction on the depth of the terms to verify this inequality. For terms of depth 0, it is verified since it reduces to the transitivity inequality of $\approx$ on Σ_0. Let us assume that it holds for terms of depth less than d, and that at least one of the terms s, t, or u, is of depth d. By transitivity of $\approx$ on Σ, we have $(f \approx g) \wedge (g \approx h) \leq f \approx h$. Also, by the inductive hypothesis, the transitivity inequality for $\approx^{\mathcal{T}}$ holds for all similar subterms of depth less than or equal to d. Therefore, this assumption entails that for all $i \in \{1, \ldots, m\}$:

$$s_i \approx^{\mathcal{T}} t_{\mu_{fg}(i)} \ \wedge \ t_{\mu_{fg}(i)} \approx^{\mathcal{T}} u_{\mu_{gh}(\mu_{fg}(i))} \ \leq \ s_i \approx^{\mathcal{T}} u_{\mu_{gh}(\mu_{fg}(i))};$$

thus, since, by Condition (2.18), it is required that the mappings be consistent and verify $\mu_{fh} = \mu_{gh} \circ \mu_{fg}$, and by isotonicity of $\wedge$ *w.r.t.* $\leq$, this is equivalent to:

$$\bigwedge_{i=1}^m (s_i \approx^{\mathcal{T}} t_{\mu_{fg}(i)}) \wedge (t_{\mu_{fg}(i)} \approx^{\mathcal{T}} u_{\mu_{fh}(i)}) \ \leq \ \bigwedge_{i=1}^m s_i \approx^{\mathcal{T}} u_{\mu_{fh}(i)}.$$

In summary, the inequality we seek to establish is of the form $A \wedge B \wedge \Delta \leq A' \wedge B'$, and we have shown that $A \leq A'$ and $B \leq B'$. From this, the inequality follows by isotonicity of $\wedge$ *w.r.t.* $\leq$. □

Since we have just formally defined a new notion of similarity $\approx^{\mathcal{T}}$ on $\mathcal{T}$ extending Sessa's similarity $\sim^{\mathcal{T}}$ to non-aligned functors, all the properties we covered for $\sim^{\mathcal{T}}$ carry over to corresponding extensions for terms with non-aligned functors. Namely, Definitions 2.6–2.8 and Lemmas 2.3–2.4, as well as Corollary 2.3, where the term similarity $\sim^{\mathcal{T}}$ is replaced with any similarity on $\mathcal{T}$ such as $\approx^{\mathcal{T}}$ (or $\approxeq^{\mathcal{T}}$ that we shall define later and prove also to be a similarity on $\mathcal{T}$ extending $\approx^{\mathcal{T}}$). Indeed, it is easy to see that all these notions are valid algebraically when parameterized with any relation on $\mathcal{FOT}$ proven to be a similarity on $\mathcal{T}$.

Weak unification with fuzzy functor/arity mismatch

Starting with the Herbrand-Martelli-Montanari ruleset of Figure 2.3, fuzziness is introduced in Sessa's weak unification by relaxing "**TERM DECOMPOSITION**" to make it also tolerate possible arity or argument-order mismatch in two structures being unified. It is the only rule that does not preserve the equation set's similarity degree. In the same manner, Rule **FUZZY NON-ALIGNED-ARGUMENT TERM DECOMPOSITION** in Figure 2.11 is the only one that may possibly alter (decrease) the equation sets' similarity degree. Also, the given functor similarity relation $\approx$ on Σ is adjoined a position mapping from argument positions of a functor f to those of a functor g when $f \approx_\alpha g$ with $f \neq g$, for some α in $(0, 1]$. This is then taken into account in tolerating a fuzzy mismatch between two term structures $s \stackrel{\text{def}}{=} f(s_1, \ldots, s_m)$ and $t \stackrel{\text{def}}{=} g(t_1, \ldots, t_n)$. This may involve a mismatch between the terms' functor symbols (f and g), their arities (m and n), subterm ordering, or a combination. We first reorient all such equations by flipping sides so that the left-hand side is the one wih lesser or equal arity. In this manner, assuming $f \approx^p_\beta g$ and $0 \leq \alpha, \beta \leq 1$, an equation set of the form: $\{ \ \ldots \ , f(s_1, \ldots, s_m) \doteq g(t_1, \ldots, t_n), \ \ldots \ \}_\alpha$ for $0 \leq m \leq n$ acquires its new similarity degree $\alpha \wedge \beta$ due to functor and arity mismatch when equated. Thus, a fully fuzzified term-decomposition rule should proceed by replacing a structure equation by the conjunction of equations between their respective subterms at corresponding indices given by the one-to-one argument mapping $p : \{ 1, \ldots, m \} \to \{ 1, \ldots, n \}$, but (possibly) decreasing the original equation set similarity degree by conjoining it with that of the decomposed terms' functor pair; that is, $\{ \ldots, s_1 \doteq t_{p(1)}, \ \ldots, \ s_m \doteq t_{p(m)}, \ldots \}_{\alpha \wedge \beta}$. Note that all the subterms in the right-hand side term that are arguments at indices which are not p-images are ignored as they have no counterparts in the left-hand side. These terms are simply dropped as part of the approximation. This generic rule is shown in Figure 2.11 along with another rule needed to make it fully effective: a rule reorienting a term equation into one with a lesser-arity term on the left.

DEFINITION 2.10 (FUZZY UNIFICATION RULE CORRECTNESS) *A fuzzy unification rule that transforms E_α, a set of equations E at a prior approximation degree α, into E'_β, a set of equations E' at a posterior approximation degree β, is said to be correct iff β is the largest degree such that $\beta \leq \alpha$ and all the solutions of E' are also solutions of E at approximation degree β.*

Note that this notion of correcteness, contrary to that of crisp unification, does not require that all solutions of the posterior sets of equations be the same as those of of the prior set. It only states that this be so as a possibly lesser posterior approximation degree.

FUZZY NON-ALIGNED-ARGUMENT TERM DECOMPOSITION

$$\frac{(E \cup \{\, f(s_1, \ldots, s_m) \doteq g(t_1, \ldots, t_n) \,\})_\alpha}{(E \cup \{\, s_1 \doteq t_{p(1)}, \ldots, s_m \doteq t_{p(m)} \,\})_{\alpha \wedge \beta}} \left[0 \leq m \leq n;\ f \approx^p_\beta g\right]$$

FUZZY EQUATION ORIENTATION

$$\frac{(E \cup \{\, g(t_1, \ldots, t_n) \doteq f(s_1, \ldots, s_m) \,\})_\alpha}{(E \cup \{\, f(s_1, \ldots, s_m) \doteq g(t_1, \ldots, t_n) \,\})_\alpha} [0 \leq m < n]$$

Figure 2.11: Fuzzy $\mathcal{FOT}$ unification's non-aligned decomposition and orientation rules

THEOREM 2.4 *The fuzzy unification rules of Figure 2.7 where Rule "***WEAK TERM DECOMPOSITION***" is replaced by the rules of Figure 2.11 are correct.*

PROOF Rules **VARIABLE ELIMINATION**, **VARIABLE ERASURE**, and **EQUATION ORIENTATION** are those, unchanged, of Maria Sessa's weak unification. Their correctness follows from those of the corresponding Herbrand-Martelli-Montanari rules since all three rules keep their similarity degree α unchanged under the same side conditions as their crisp versions. As for Rule **FUZZY EQUATION ORIENTATION**, it is also correct as it simply uses the symmetry of equality or similarity denoted by the $\doteq$ relation and it leaves the similarity degree unchanged.

The correctness of Rule **FUZZY NON-ALIGNED-ARGUMENT TERM DECOMPOSITION** follows from the fact that it tolerates equations between two distinct but similar functors, f on the left and g on the right, by "paying a toll" as the most general way this can be true is by reducing the prior equation set's similarity degree α to $\alpha \wedge \beta$. It must do so whenever a prior equation set contains an equation between two terms whose respective head functors f and g are β-similar with f having at most as many arguments as g. It collects m corresponding subterm equations from the two terms's subterms using the specific one-to-one argument mapping p that associates with each position i among f's m a unique specific position $p(i)$ among g's n, $n \geq m$. Orienting all functorial term equations to have the lesser number of arguments on the left guarantees completeness over all such syntactic patterns. By structural induction, assuming that all $s_i \approx_{\alpha \wedge \beta} t_{p(i)}$ for all $i \in \{1, \ldots, m\}$, then whenever $f \approx^p_\beta g$, we must also have $f(s_1, \ldots, s_m) \approx_{\alpha \wedge \beta} g(t_1, \ldots, t_n)$ (by definition, since $\alpha \wedge \beta \leq \alpha$) for whatever arguments of g at indices missed by p and these two terms are in the same similarity class at approximation degree $\alpha \wedge \beta$ for arbitrary arguments in these positions (by definition of $f \approx^p_\beta g$ and since $\alpha \wedge \beta \leq \beta$). The rest of the equations in E that were true at approximation degree α must now be considered true only up to approximation degree $\alpha \wedge \beta$ in order to account for f and g being functors of possibly fuzzier similarity β. Hence, all solutions of the new set of equations are also solutions of the previous one, although only at the possibly lesser approximation degree $\alpha \wedge \beta$. This approximation degree is also the greatest such degree by virtue

of the $\wedge$ operation yielding the infimum of its operands.

Finally, when $m = n$ this rule is correct in either direction since a consistent similarity on a signature requires by definition that equal-arity functors f and g have arguments in bijection (inverse permutations of the set $\{1, \ldots, n\}$): $f \approx_\alpha^p g$ and $g \approx_\alpha^{p^{-1}} f$. In this case, the set of solutions of the new equation set is also a solution of the previous one, with equal similarity degree.

As for termination, it follows (like that of the Herbrand-Martelli-Montanari rules) from (1) the finite width and depth of $\mathcal{FOT}$s, and (2) there being no rule that is indefinitely applicable. Regarding (1), term decomposition always replaces a term equation with finitely many shallower term equations, which is a well-known well-founded process guaranteed to terminate (multiset ordering [4]). Regarding (2), Rule **FUZZY EQUATION ORIENTATION** may not be reapplied to the same functors thanks to the side condition $m < n$.

In other words, applying this modified ruleset to $E_1 \stackrel{\text{def}}{=} \{s \doteq t\}_1$, an equation set of similarity degree 1 (in any order as long as a rule applies and its similarity degree is not zero) always terminates. And when the final equation set is a substitution σ at approximation degree α, σ is the most general substitution (up to a variable renaming) that is a solution at approximation degree α (*i.e.*, $s\sigma \approx_\alpha t\sigma$), and α is the greatest approximation degree for which this is true. □

Example 2.9 $\mathcal{FOT}$ **fuzzy unification with similar functors of different arities** — Take a functor signature such that: $\{a, b, c, d\} \subseteq \Sigma_0$, $\{f, g, \ell\} \subseteq \Sigma_2$, $\{h\} \subseteq \Sigma_3$; and let us further assume the functor similarity that is the reflexive symmetric transitive closure of:[19]

$$a \approx_{.7} b, \quad c \approx_{.6} d, \quad f \approx_{.9}^{\{1\to 2, 2\to 1\}} g, \quad g \approx_{.9}^{\{1\to 2, 2\to 1\}} f, \quad \text{and} \quad \ell \approx_{.8}^{\{1\to 2, 2\to 3\}} h.$$

Let us consider the fuzzy equation set $\{t_1 \doteq t_2\}_1$:

$$\{h(X, g(Y, b), f(Y, c)) \doteq \ell(f(a, Z), g(d, c))\}_1 \tag{2.20}$$

and let us apply the rules of Figure 2.7 where rule **WEAK TERM DECOMPOSITION** has been replaced by the rules of Figure 2.11:
Rule **FUZZY EQUATION ORIENTATION** with $\alpha = 1$ because **arity**$(\ell) <$ **arity**(h); new set: $\{\ell(f(a, Z), g(d, c)) \doteq h(X, g(Y, b), f(Y, c))\}_1$;
Rule **FUZZY NON-ALIGNED-ARGUMENT TERM DECOMPOSITION** with $\alpha = 1$ and $\beta = .8$ since $\ell \approx_{.8}^{\{1\to 2, 2\to 3\}} h$; new set: $\{f(a, Z) \doteq g(Y, b), g(d, c) \doteq f(Y, c)\}_{.8}$;
Rule **FUZZY NON-ALIGNED-ARGUMENT TERM DECOMPOSITION** to $f(a, Z) \doteq g(Y, b)$ with $\alpha = .8$ and $\beta = .9$ since $f \approx_{.9}^{\{1\to 2, 2\to 1\}} g$; new set: $\{a \doteq b, Z \doteq Y, g(d, c) \doteq f(Y, c)\}_{.8}$;
Rule **FUZZY NON-ALIGNED-ARGUMENT TERM DECOMPOSITION** to $a \doteq b$ with $\alpha = .8$ and $\beta = .7$ since $a \approx_{.7} b$; new set: $\{Z \doteq Y, g(d, c) \doteq f(Y, c)\}_{.7}$;
Rule **FUZZY NON-ALIGNED-ARGUMENT TERM DECOMPOSITION** to $g(d, c) \doteq f\ (Y, c)$ with $\alpha = .7$ and $\beta = .9$ since $f \approx_{.9}^{\{1\to 2, 2\to 1\}} g$; new set: $\{Z \doteq Y, d \doteq c, c \doteq Y\}_{.7}$;
Rule **FUZZY NON-ALIGNED-ARGUMENT TERM DECOMPOSITION** to $d \doteq c$ with $\alpha = .7$ and $\beta = .6$ since $d \approx_{.6} c$; new set: $\{Z \doteq Y, c \doteq Y\}_{.6}$;
Rule **EQUATION ORIENTATION** to $c \doteq Y$ with $\alpha = .6$; new set:: $\{Z \doteq Y, Y \doteq c\}_{.6}$.
Rule **VARIABLE ELIMINATION** to $Y \doteq c$ with $\alpha = .6$; new set: $\{Z \doteq c, Y \doteq c\}_{.6}$.

[19] Recall that the argument mapping is the identity by default.

This last equation set at approximation degree .6 is in normal form and defines the substitution $\sigma = \{\ c/Z, c/Y\ \}$ so that: $t_1\sigma = h(X, g(Y, b), f(Y, c))\sigma \ \approx_{.6}\ \ell(f(a, Z), g(d, c))\sigma = t_2\sigma$; that is: $t_1\sigma = h(X, g(c, b), f(c, c)) \ \approx_{.6}\ \ell(f(a, c), g(d, c)) = t_2\sigma$.

Example 2.10 Example 2.9 with more expressive symbols — Example 2.9 uses abstract generic symbols such as f, g, a, b, ..., which have the advantage, being one-letter symbols, to make each normalization step more compact to write. But one may be more content with a more illustrative choice of identifiers as would be the case of a real-life data base.

So let us give such names to functors of Example 2.9 in the case of a gift-shop Prolog database which describes various configurations for multi-item gift boxes containing such items as flowers, sweets, *etc.*, which can be already joined as pairs or not joined as loose couples. In such a database are consigned facts over objects identified by the following functors (each corresponding to the one indicated from Example 2.9):

- $a/0 \stackrel{\text{def}}{=} \mathtt{violet}$,
- $b/0 \stackrel{\text{def}}{=} \mathtt{lilac}$,
- $c/0 \stackrel{\text{def}}{=} \mathtt{chocolate}$,
- $d/0 \stackrel{\text{def}}{=} \mathtt{candy}$,
- $f/2 \stackrel{\text{def}}{=} \mathtt{pair}$,
- $g/2 \stackrel{\text{def}}{=} \mathtt{couple}$,
- $h/3 \stackrel{\text{def}}{=} \mathtt{small\text{-}gift\text{-}box}$,

with the closure under reflexivity, symmetry, and transitivity of the following similar pairs:

- $\mathtt{violet} \sim_{.7} \mathtt{lilac}$,
- $\mathtt{chocolate} \sim_{.6} \mathtt{candy}$,
- $\mathtt{pair} \sim_{.9} \mathtt{couple}$.

With these functors and their similarity degrees, Equation (2.12) now reads:

$$
\begin{array}{ll}
 & \mathtt{small\text{-}gift\text{-}box}\,(\ \mathtt{pair}(\mathtt{violet}, X_1) \\
(t_1) & \qquad\qquad\quad ,\ \mathtt{couple}(X_1, \mathtt{lilac}) \\
 & \qquad\qquad\quad ,\ \mathtt{pair}(Y_1, Y_1) \\
 & \qquad\qquad\quad)
\end{array}
$$

$$\doteq$$

$$
\begin{array}{ll}
 & \mathtt{small\text{-}gift\text{-}box}\,(\ X_2 \\
(t_2) & \qquad\qquad\quad ,\ X_2 \\
 & \qquad\qquad\quad ,\ \mathtt{couple}(\mathtt{chocolate}, \mathtt{candy}) \\
 & \qquad\qquad\quad)
\end{array}
$$

Substitution (2.13) obtained after normalization is now defined as follows, using the new functor symbols:

$$\sigma \stackrel{\text{def}}{=} \{X_1 = \mathit{violet}, Y_1 = \mathit{chocolate}, X_2 = \mathit{pair}(\mathit{violet}, \mathit{violet})\}$$

and yields the fuzzy solution:

$$(t_1\sigma) \quad \begin{array}{l} \mathit{small\text{-}gift\text{-}box}(\ \mathit{pair}(\mathit{violet}, \mathit{violet}) \\ \qquad\qquad\quad , \ \mathit{couple}(\mathit{violet}, \mathit{lilac}) \\ \qquad\qquad\quad , \ \mathit{pair}(\mathit{chocolate}, \mathit{chocolate}) \\ \qquad\qquad\quad) \end{array}$$

$$\sim_{.6}$$

$$(t_2\sigma) \quad \begin{array}{l} \mathit{small\text{-}gift\text{-}box}(\ \mathit{pair}(\mathit{violet}, \mathit{violet}) \\ \qquad\qquad\quad , \ \mathit{pair}(\mathit{violet}, \mathit{violet}) \\ \qquad\qquad\quad , \ \mathit{couple}(\mathit{chocolate}, \mathit{candy}) \\ \qquad\qquad\quad) \end{array}$$

with similarity degree .6 capturing the fuzzy degree to which σ solves the original equation.

Rule **FUZZY NON-ALIGNED-ARGUMENT TERM DECOMPOSITION** is a very general rule for normalizing fuzzy equations over $\mathcal{FOT}$ structures. It has the following convenient properties:

1. it accounts for fuzzy mismatches of similar functors of possibly different arity or order of arguments;

2. when restricted to tolerating only similar equal-arity functors with matching argument positions, it reduces to Sessa's weak unification's **WEAK TERM DECOMPOSITION** rule;

3. when similarity degrees are further restricted to be in $\{0, 1\}$, it is the Herbrand-Martelli-Montanari **TERM DECOMPOSITION** rule;

4. it requires no alteration of the standard notions of $\mathcal{FOT}$s and $\mathcal{FOT}$ substitutions: similarity among $\mathcal{FOT}$s is derived from that of signature symbols;

5. finally, and most importantly, it keeps fuzzy unification in the same complexity class as crisp unification: that of Union-Find [127].[20]

As a result, it is more general than all other extant approaches we know which propose a fuzzy $\mathcal{FOT}$ unification operation. The same will be established for the fuzzification of the dual operation: first a limited "*functor-weak*" $\mathcal{FOT}$ generalization corresponding to the dual operation of Sessa's "weak" unification, then to a more expressive "*functor/arity-weak*" $\mathcal{FOT}$ generalization corresponding to our extension of Sessa's unification to functor/arity weak unification.

[20] Quasi-linear; *i.e.*, linear with a $\log \ldots \log$ coefficient [2].

2.6.2 Fuzzy $\mathcal{FOT}$ generalization

While there has been relatively intense interest in devising a fuzzy $\mathcal{FOT}$ unification operation, we know of ***no*** work regarding its dual operation, fuzzy $\mathcal{FOT}$ generalization. This comes as no surprise since even in the crisp case only marginal attention has been paid to generalization (*a.k.a.* anti-unification) as compared to unification.

The Reynolds-Plotkin characterization of $\mathcal{FOT}$ subsumption as a lattice ordering relies on formalizing this ordering as $\mathcal{FOT}$ instantiation. Namely, $t_1 \preceq t_2$ iff there exists a variable substitution σ such that $t_1 = t_2\sigma$. Then, unification and generalization are respectively the **glb** and **lub** operations for this ordering and are specified in terms of variable substitutions.

It is clear however, as overviewed in the previous section, that there are several ways one can propose to fuzzify $\mathcal{FOT}$ unification. As a consequence of this, for each specific fuzzification of $\mathcal{FOT}$ unification, and therefore of associated specific fuzzy subsumption ordering on $\mathcal{FOT}$s, there should also correspond a dual operation of fuzzy generalization of $\mathcal{FOT}$s.

In what follows, we first elaborate some lattice-theoretic consequences for Maria Sessa's "weak unification" fuzzy operation on $\mathcal{FOT}$s presented in [117]. In particular, we derive its corresponding fuzzy dual lattice operation that we shall dub "weak $\mathcal{FOT}$ generalization." We then extend this lattice to signatures admitting similar functors with differing arity or argument order.

Fuzzy functor-weak generalization

Let t_1 and t_2 be two $\mathcal{FOT}$s in $\mathcal{T}$ to generalize. We shall use the following notation for a fuzzy generalization judgment:

$$\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha} \vdash \begin{pmatrix}t_1\\ t_2\end{pmatrix} t \begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix}_{\beta} \tag{2.21}$$

given:

- $\sigma_i \in \textbf{SUBST}_{\mathcal{T}}$ $(i = 1, 2)$: two prior substitutions with prior similarity degree α,
- t_i $(i = 1, 2)$: two prior $\mathcal{FOT}$s,
- t: a posterior $\mathcal{FOT}$,
- $\theta_i \in \textbf{SUBST}_{\mathcal{T}}$ $(i = 1, 2)$: two posterior substitutions with similarity degree β.

DEFINITION 2.11 (FUZZY $\mathcal{FOT}$ GENERALIZATION JUDGMENT VALIDITY) *A fuzzy $\mathcal{FOT}$ generalization judgment such as (2.21) is valid whenever, for $i = 1, 2$:*

1. $\beta \in (0, \alpha]$;
2. $t_i\sigma_i \approx_\beta t\theta_i$;
3. $\exists\, \delta_i \in \textbf{SUBST}_{\mathcal{T}}$ *s.t.* $t_i \approx_\alpha t\delta_i$ *and* $\theta_i \approx_\beta \delta_i\sigma_i$ *(i.e.,* $t_i \preceq_\alpha t\sigma_i$ *and* $\theta_i \preceq_\beta \sigma_i$*).*

Figure 2.12 shows an illustration of a valid fuzzy generalization judgment constraint as a commutative diagram.

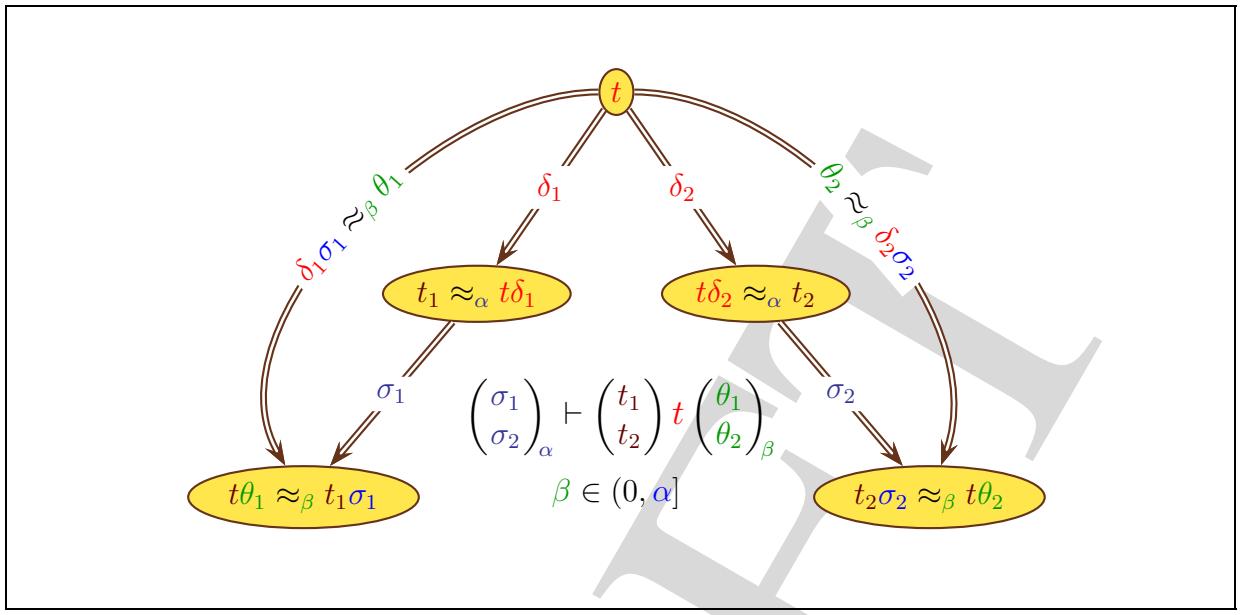


Figure 2.12: Fuzzy generalization judgment validity as a constraint

DEFINITION 2.12 (FUZZY GENERALIZATION RULE CORRECTNESS) *A fuzzy generalization rule is correct iff, whenever the side condition holds, if all the fuzzy generalization judgments making up its antecedent are valid, then necessarily the fuzzy generalization judgment in its consequent is valid.*

In Figure 2.13, we give a fuzzy version of the generalization rules of Figure 2.5. As was the case in Sessa's weak unification, we assume as well for now that we are given a similarity relation $\sim\colon \Sigma \times \Sigma \to [0,1]$ on the signature $\Sigma = \cup_{n\geq 0}\Sigma_n$ such that for all $m \geq 0$ and $n \geq 0$, $m \neq n$ implies $f \not\sim g$. In other words, functors of different arities may not be similar.

Rule **SIMILAR FUNCTORS** uses a "*fuzzy unapply*" operation ('$\uparrow_\alpha$') on a pair of terms (t_1, t_2) given a pair of substitutions (σ_1, σ_2) and a similarity degree α. It is the result of "*unapplying*" σ_i from t_i, for $i = 1, 2$, into a common variable X, if any such exists such that the terms $X\sigma_i$ are respectively similar to t_i with similarity degrees α_i. It returns a fuzzy pair of terms and a similarity degree in $(0, \alpha]$ defined as:

$$\begin{pmatrix}t_1\\ t_2\end{pmatrix} \uparrow_\alpha \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} \stackrel{\text{def}}{=} \begin{cases} \begin{pmatrix}X\\ X\end{pmatrix}_{\alpha\wedge\alpha_1\wedge\alpha_2} & \text{if } \exists\, X \in \mathcal{V},\ t_i \sim_{\alpha_i} X\sigma_i \\ & \text{for some } \alpha_i \in (0,1],\ i = 1,2; \\ \begin{pmatrix}t_1\\ t_2\end{pmatrix}_\alpha & \text{otherwise.} \end{cases} \tag{2.22}$$

The condition in Equation (2.22) is: "$\exists\, X \in \mathcal{V},\ t_i \sim_{\alpha_i} X\sigma_i$, for some $\alpha_i \in (0,1]$ $(i = 1, 2)$." But could there be two such variables? Namely, is it ever possible that:

$$\exists\, X \in \mathcal{V}, \exists\, Y \in \mathcal{V}, X \neq Y, \text{ s.t. } t_i \sim_{\alpha_i} X\sigma_i \text{ and } t_i \sim_{\beta_i} Y\sigma_i \tag{2.23}$$

FUZZY EQUAL VARIABLES

$$\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_\alpha \vdash \begin{pmatrix}X\\ X\end{pmatrix} X \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_\alpha$$

FUZZY VARIABLE-TERM

$$[t_1 \in \mathcal{V} \text{ or } t_2 \in \mathcal{V};\ \ t_1 \neq t_2;\ \ X \text{ is new}]$$
$$\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_\alpha \vdash \begin{pmatrix}t_1\\ t_2\end{pmatrix} X \begin{pmatrix}\sigma_1\{\, t_1/X\,\}\\ \sigma_2\{\, t_2/X\,\}\end{pmatrix}_\alpha$$

DISSIMILAR FUNCTORS

$$[f \not\sim g;\ \ m \geq 0, n \geq 0;\ \ X \text{ is new}]$$
$$\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_\alpha \vdash \begin{pmatrix}f(s_1,\ldots,s_m)\\ g(t_1,\ldots,t_n)\end{pmatrix} X \begin{pmatrix}\sigma_1\{\, f(s_1,\ldots,s_m)/X\,\}\\ \sigma_2\{\, g(t_1,\ldots,t_n)/X\,\}\end{pmatrix}_\alpha$$

SIMILAR FUNCTORS

$$\left[f \sim_\beta g;\ \ \beta > 0;\ \ n \geq 0;\ \ \alpha_0 \stackrel{\text{def}}{=} \alpha \wedge \beta\right]$$
$$\frac{\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha_0} \vdash \begin{pmatrix}s_1\\ t_1\end{pmatrix} \uparrow_{\alpha_0} \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} u_1 \begin{pmatrix}\sigma_1^1\\ \sigma_2^1\end{pmatrix}_{\alpha_1} \cdots \begin{pmatrix}\sigma_1^{n-1}\\ \sigma_2^{n-1}\end{pmatrix}_{\alpha_{n-1}} \vdash \begin{pmatrix}s_n\\ t_n\end{pmatrix} \uparrow_{\alpha_{n-1}} \begin{pmatrix}\sigma_1^{n-1}\\ \sigma_2^{n-1}\end{pmatrix} u_n \begin{pmatrix}\sigma_1^n\\ \sigma_2^n\end{pmatrix}_{\alpha_n}}{\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_\alpha \vdash \begin{pmatrix}f(s_1,\ldots,s_n)\\ g(t_1,\ldots,t_n)\end{pmatrix} f(u_1,\ldots,u_n) \begin{pmatrix}\sigma_1^n\\ \sigma_2^n\end{pmatrix}_{\alpha_n}}$$

Figure 2.13: Functor-weak generalization axioms and rule

for some $\alpha_i \in (0,1]$ and $\beta_i \in (0,1]$, $i = 1, 2$? Note that a new variable is introduced in the generalizing pair of substitutions only in Axiom **FUZZY VARIABLE-TERM** and Axiom **DISSIMILAR FUNCTORS**. Then, each axiom binds the new variable in the two substitutions to two terms that are dissimilar at any similarity degree (as required by their side conditions). However, by Lemma 2.4 on Page 23, Condition (2.23) would imply that:

$$t_i \sim_{\alpha_i \wedge \beta_i} X\sigma_i \sim_{\alpha_i \wedge \beta_i} Y\sigma_i$$

with $\alpha_i \wedge \beta_i \in (0,1]$, for $i = 1, 2$. This would mean that X or Y would have been introduced while the side condition of neither Axiom **FUZZY VARIABLE-TERM** nor Axiom **DISSIMILAR FUNCTORS** was verified; which is impossible. Thus, there can be at most only one such variable.

As importantly, note also that fuzzy unapplication defined by Equation (2.22) returns a pair of terms and a possibly lesser or equal approximation degree, unlike crisp unapplication defined by Equation (2.7) which returns only a pair of terms. Because of this, when we write a fuzzy judgment such as:

$$\begin{pmatrix}\sigma\\ \sigma'\end{pmatrix}_\alpha \vdash \begin{pmatrix}t\\ t'\end{pmatrix} \uparrow_\alpha \begin{pmatrix}\sigma\\ \sigma'\end{pmatrix} u \begin{pmatrix}\theta\\ \theta'\end{pmatrix}_\beta \tag{2.24}$$

as we do in the premiss of Rule **SIMILAR FUNCTORS**, this is shorthand to indicate that the posterior similarity degree β is *at most* the one returned by the fuzzy unapplication $\begin{pmatrix}t\\ t'\end{pmatrix} \uparrow_\alpha \begin{pmatrix}\sigma\\ \sigma'\end{pmatrix}$.

Formally, the notation of the fuzzy judgment (2.24) is equivalent to:

$$\begin{pmatrix} s \\ s' \end{pmatrix}_{\beta'} \stackrel{\text{def}}{=} \begin{pmatrix} t \\ t' \end{pmatrix} \uparrow_{\alpha} \begin{pmatrix} \sigma \\ \sigma' \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \sigma \\ \sigma' \end{pmatrix}_{\beta'} \vdash \begin{pmatrix} s \\ s' \end{pmatrix} u \begin{pmatrix} \theta \\ \theta' \end{pmatrix}_{\beta} \tag{2.25}$$

for some β' such that $\beta \leq \beta' \leq \alpha$. This is because a fuzzy unapplication invoked while proving the validity of a fuzzy judgment may require, by Expression (2.22), lowering the *prior* approximation degree of the judgment.

Finally, note that Rule "**SIMILAR FUNCTORS**" is defined for $n \geq 0$. For $n = 0$, it becomes the following fuzzy judgment:

$$\begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}_{\alpha} \vdash \begin{pmatrix} c \\ c \end{pmatrix} c \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}_{\alpha} \tag{2.26}$$

which can be verified to be an axiom since it is valid at any approximation degree α in $[0, 1]$, for any constant c in Σ_0, and any substitutions σ_1 and σ_2 in **SUBST** $_{\mathcal{T}}$, thanks to the reflexivity of the similarity $\sim_{\alpha}$ on $\mathcal{T}$.

Referring to the axioms (seen as rules with no antecedent) and the rule of Figure 2.13, we establish the following fact corresponding to Lemma 2.2 on Page 14 (taking $\sigma_i^0 \stackrel{\text{def}}{=} \sigma_i$, for $i = 1, 2$), where the fuzzy ordering on substitutions is defined in Definition 2.7 on Page 23.

LEMMA 2.5 *In Rule* **SIMILAR FUNCTORS** *of Figure 2.13, taking* $\sigma_i^0 \stackrel{\text{def}}{=} \sigma_i$, *for* $i = 1, 2$, *the approximation degrees* $\alpha_i^0, \ldots, \alpha_i^n$ *are such that* $\alpha_i^k \leq \alpha_i^{k-1}$, *and the substitutions* $\sigma_i^0, \ldots, \sigma_i^n$ *are such that* $\sigma_i^k \preceq_{\alpha_i^k} \sigma_i^{k-1}$, *for all* k, $1 \leq k \leq n$ $(i = 1, 2)$.

PROOF We proceed by induction on the depth d of the terms; *i.e.*, we consider only terms of depth less than or equal to d.

1. $d = 0$: This limits terms to constants and variables. The inequality between prior and posterior substitutions is verified for the three first axioms of Figure 2.13: each preserves the prior approximation degree and the posterior substitutions are all either equal to the corresponding prior substitutions or of the form $\theta = \sigma\{t/X\}$ where X is a new variable and σ is the corresponding prior substitution. As well, when limited to terms of 0 depth, Rule **SIMILAR FUNCTORS** becomes the Axiom (2.26), which preserves both the approximation degree and the substitutions.
2. $d > 0$: Let us assume now that this is true for all terms of depth less than d. That is, we consider two terms to generalize, at least one of which is of depth d. The same argument given above for when $d = 0$ for the three first axioms of Figure 2.5 justifies concluding that $\theta \preceq \sigma$, since this is true in these cases for terms of any depth. As for Rule **EQUAL FUNCTORS**, there are two possible cases for the two terms in its consequent (the "denominator"):
   (a) $n = 0$: then, the conclusion follows true by Axiom (2.26);
   (b) $n \geq 0$: since the fuzzy unapply operation (2.22) yields either a pair of terms having the same depth as the corresponding terms it is applied to, or 0 (because it can only be a new variable), we can say that all the terms resulting from fuzzy-unapplied pairs of arguments in the judgments of the rule's antecedent (the "numerator") are of depth at

most $d-1$. Therefore, this fact, together with our induction hypothesis being verified for depths less than d and the expression of a fuzzy judgment (2.25) involving only terms of such depths, we can conclude that all the judgments in the rule's antecedent can only reduce their prior approximation degree. Therefore, $\alpha_i^k \leq \alpha_i^{k-1}$ and $\sigma_i^k \preceq_{\alpha_i^k} \sigma_i^{k-1}$, for all $k = 1, \ldots, n$. Then, by Corollary 2.3 and transitivity of the "more general" ordering on substitutions $\preceq_\alpha$ at fixed α, the conclusion follows.

Hence, this establishes that, for both $i = 1, 2$, the approximation degree α_i^k is monotonically decreasing and the substitution σ_i^k is monotonically refined from more general to less, as k increases from 1 to n; which concludes our proof. □

And the corresponding corollary also follows.

COROLLARY 2.4 *In Rule* **SIMILAR FUNCTORS** *of Figure 2.13, for all* k, $1 \leq k \leq n$*:*

- *the approximation degrees* α_i^k *are such that* $\alpha_i^n \leq \alpha_i^{n-1} \leq \ldots \alpha_i^1 \leq \alpha_i^0$*, and*
- *the substitutions* σ_i^k *are such that* $\sigma_i^n \preceq_{\alpha_i^n} \sigma_i^{n-1} \preceq_{\alpha_i^{n-1}} \ldots \sigma_i^1 \preceq_{\alpha_i^1} \sigma_i^0$*,*

for $i = 1, 2$.

THEOREM 2.5 (FUNCTOR-WEAK GENERALIZATION CORRECTNESS) *The fuzzy generalization rules of Figure 2.13 are correct.*

PROOF We must show that they verify the conditions of Definition 2.12 on page 38. For each of the three axioms of Figure 2.13, this means that they must be valid as fuzzy judgments, verifying the three conditions of Definition 2.11, which are:

- *Condition 1*: $\beta \in (0, \alpha]$,
- *Condition 2*: $t_i\sigma_i \sim_\beta t\theta_i$,
- *Condition 3*: $t_i \preceq_\alpha t$ and $\theta_i \preceq_\beta \sigma_i$,

for $i = 1, 2$, for a fuzzy $\mathcal{FOT}$ generalization judgment such as (2.21). These conditions for the axioms and the rule of Figure 2.13 translate as the following.

Condition 1. All three axioms verify this condition because they preserve the approximation degree.

Condition 2. This condition becomes the following for each of the three axioms (for $i = 1, 2$):

- **FUZZY EQUAL VARIABLES**: Condition 2 becomes the similarity $X\sigma_i \sim_\alpha X\sigma_i$, which is true by reflexivity of $\sim_\alpha$ for all X, σ_i, and α;
- **FUZZY VARIABLE-TERM**: it becomes the similarity $t_i\sigma_i \sim_\alpha t_i\sigma_i$, which is true also by reflexivity of $\sim_\alpha$, for all t_i, σ_i, and α;
- **DISSIMILAR FUNCTORS**: Condition 2 becomes:

$$f(s_1, \ldots, s_m)\sigma_1 \sim_\alpha X\sigma_1\{\, f(s_1, \ldots, s_m)/X \,\}$$
$$g(t_1, \ldots, t_n)\sigma_2 \sim_\alpha X\sigma_2\{\, g(t_1, \ldots, t_n)/X \,\}$$

which, because X is a new variable that does not occurs in either σ_1 or σ_2, simplify respectively to the similarities:

$$f(s_1,\ldots,s_m) \sim_\alpha f(s_1,\ldots,s_m)$$
$$g(t_1,\ldots,t_n) \sim_\alpha g(t_1,\ldots,t_n)$$

which hold by reflexivity of $\sim_\alpha$ at any approximation degree α.

Condition 3. The three axioms verify the following at all approximation degrees α and β (for $i = 1, 2$):

- **FUZZY EQUAL VARIABLES**: $X \preceq_\alpha X$ and $\sigma_i \preceq_\beta \sigma_i$;
- **FUZZY VARIABLE-TERM**: $t_i \preceq_\alpha X$ and $\sigma_i\{t_i/X\} \preceq_\beta \sigma_i$;
- **DISSIMILAR FUNCTORS**:

$$f(s_1,\ldots,s_m) \preceq_\alpha X \text{ and } \sigma_1\{f(s_1,\ldots,s_m)/X\} \preceq_\beta \sigma_1,$$
$$g(t_1,\ldots,t_n) \preceq_\alpha X \text{ and } \sigma_2\{g(t_1,\ldots,t_n)/X\} \preceq_\beta \sigma_2.$$

As for Rule **SIMILAR FUNCTORS**, as required by Definition 2.12, we must show that if all the fuzzy judgments in the numerator are valid, then the fuzzy judgment in the denominator is valid too. For all three conditions, let us proceed by induction on the arity n:

Condition 1. For $n = 0$, the conclusion follows also because Axiom (2.26) applies and it also preserves the approximation degree; for $n > 0$, if we assume that $0 \leq \alpha_k \leq \alpha_{k-1} \leq 1$ for all $k = 1,\ldots,n$, by transitivity of $\leq$ on $[0,1]$, it follows that $0 \leq \alpha_n \leq \alpha_0 \leq 1$, which verifies the definition.

Condition 2. For $n = 0$, this rule becomes Axiom (2.26). Since it preserves the approximation degree, Condition 1 is verified. Also, this fuzzy judgment is trivially valid at all approximation degrees: the conditions of Definition 2.11 become the reflexive similarity $c \sim_\alpha c$, and the conjunction of reflexive fuzzy inequality $c \preceq_\alpha c$ and reflexive substitution fuzzy inequalities $\sigma_i \preceq_\alpha \sigma_i$, for $i = 1, 2$. Thus, this verifies both Condition 2 and Condition 3 for $n = 0$.

For $n > 0$, for each argument-position $k = 1,\ldots,n$, a fuzzy judgment in the rule's antecedent is of the form:

$$\begin{pmatrix}\sigma_1^{k-1}\\ \sigma_2^{k-1}\end{pmatrix}_{\alpha_{k-1}} \vdash \begin{pmatrix}s_k\\ t_k\end{pmatrix} \uparrow_{\alpha_{k-1}} \begin{pmatrix}\sigma_1^{k-1}\\ \sigma_2^{k-1}\end{pmatrix} u_k \begin{pmatrix}\sigma_1^{k}\\ \sigma_2^{k}\end{pmatrix}_{\alpha_k};$$

that is, the form of Expression (2.24), whose formal meaning is given as Expression (2.25), which in the above case is equivalent to:

$$\begin{pmatrix}v_1^{k}\\ v_2^{k}\end{pmatrix}_{\beta_k} \stackrel{\text{def}}{=} \begin{pmatrix}s_k\\ t_k\end{pmatrix} \uparrow_{\alpha_{k-1}} \begin{pmatrix}\sigma_1^{k-1}\\ \sigma_2^{k-1}\end{pmatrix} \text{ and } \begin{pmatrix}\sigma_1^{k-1}\\ \sigma_2^{k-1}\end{pmatrix}_{\beta_k} \vdash \begin{pmatrix}v_1^{k}\\ v_2^{k}\end{pmatrix} u_k \begin{pmatrix}\sigma_1^{k}\\ \sigma_2^{k}\end{pmatrix}_{\alpha_k}$$

for some β_k s.t. $\alpha_{k-1} \leq \beta_k \leq \alpha_k$. Let us now assume that all the fuzzy judgment in the rule's antecedent are valid. That is, for $k = 1,\ldots,n$ (defining $\alpha_0 \stackrel{\text{def}}{=} \alpha \wedge \beta$), for $i = 1, 2$:

$$u_k\sigma_i^k \sim_{\alpha_k} v_i^k\sigma_i^{k-1} \tag{2.27}$$

and (defining $\sigma_i^0 \stackrel{\text{def}}{=} \sigma_i$):

$$v_i^k \preceq_\alpha u_k \text{ and } \sigma_i^k \preceq_\beta \sigma_i^{k-1}. \tag{2.28}$$

By Equation (2.22), this means:

$$\binom{v_1^k}{v_2^k}_{\alpha_k} \stackrel{\text{def}}{=} \begin{cases} \binom{X}{X}_{\alpha_{k-1}\wedge\beta_1^k\wedge\beta_2^k} & \text{if } \exists\, X \in \mathcal{V} \text{ s.t. } s_k \sim_{\beta_1^k} X\sigma_1^{k-1} \text{ and } t_k \sim_{\beta_2^k} X\sigma_2^{k-1}; \\ \binom{s_k}{t_k}_{\alpha_{k-1}} & \text{otherwise.} \end{cases}$$

for some β_1^k and β_2^k in $(0, 1]$. In other words, for each $k = 1, \ldots, n$, there are two cases:

1. $s_k \sim_{\beta_1^k} X\sigma_1^{k-1}$ and $t_k \sim_{\beta_2^k} X\sigma_2^{k-1}$ for some variable X; then, by Axiom **FUZZY EQUAL VARIABLES**, we must have $\alpha_k = \alpha_{k-1} \wedge \beta_1^k \wedge \beta_2^k$, $u_k = X$, and $\sigma_i^k = \sigma_i^{k-1}$ for $i = 1, 2$; thus, $\alpha_k \leq \alpha_{k-1}$ and Similarity (2.27) becomes $u_k\sigma_i^k \sim_{\alpha_k} X\sigma_i^{k-1}$ So that:
$$s_k\sigma_1^{k-1} \sim_{\alpha_k} X\sigma_1^{k-1}\sigma_1^{k-1} = X\sigma_1^{k-1} = X\sigma_1^k \sim_{\alpha_k} u_k\sigma_1^k,$$
$$t_k\sigma_2^{k-1} \sim_{\alpha_k} X\sigma_2^{k-1}\sigma_2^{k-1} = X\sigma_2^{k-1} = X\sigma_2^k \sim_{\alpha_k} u_k\sigma_2^k.$$
2. There is no such variable X; in which case, $\alpha_k = \alpha_{k-1}$ and Similarity (2.27) becomes:
$$s_k\sigma_1^{k-1} \sim_{\alpha_k} u_k\sigma_1^k,$$
$$t_k\sigma_2^{k-1} \sim_{\alpha_k} u_k\sigma_2^k.$$

Thus, by the only non-identical transformation relating prior and posteriors substitutions in the axioms, for any argument position k, $1 \leq k \leq n$, we have:

$$\sigma_i^k \sim_{\alpha_k} \sigma_i^0\{\tau_1/X_1\}\ldots\{\tau_\ell/X_\ell\}$$

where each of the variables $X_1 \ldots X_\ell$, with $0 \leq \ell$, is a variable possibly introduced in proving the validity of the fuzzy judgment corresponding to some argument position k. Therefore, since for any argument position k, $1 \leq k \leq n$:

1. σ_i^k affects only a *new* variable introduced in one of the axioms verifying the validity of the subterm at argument position k; and,
2. such a newly introduced variable now occurring in u_k is always instantiated by the same term;

it comes that, at approximation degree α_k:

$$s_k\sigma_1^0 \sim_{\alpha_k} s_k\sigma_1^1 \sim_{\alpha_k} \ldots \sim_{\alpha_k} s_k\sigma_1^{k-1}$$
$$t_k\sigma_2^0 \sim_{\alpha_k} t_k\sigma_2^1 \sim_{\alpha_k} \ldots \sim_{\alpha_k} t_k\sigma_2^{k-1}$$

as well as, at approximation degree α_n:

$$u_k\sigma_1^k \sim_{\alpha_n} u_k\sigma_1^{k+1} \sim_{\alpha_n} \ldots \sim_{\alpha_n} u_k\sigma_1^n$$
$$u_k\sigma_2^k \sim_{\alpha_n} u_k\sigma_2^{k+1} \sim_{\alpha_n} \ldots \sim_{\alpha_n} u_k\sigma_2^n$$

which shows that in both cases we have, for all $k = 1, \ldots, n$:

$$s_k\sigma_1^0 \sim_{\alpha_k} u_k\sigma_1^n$$
$$t_k\sigma_2^0 \sim_{\alpha_k} u_k\sigma_2^n.$$

Therefore, for $k = n$:

$$f(s_1, \ldots, s_n)\sigma_1^0 \ \sim_{\alpha_n} \ f(u_1, \ldots, u_n)\sigma_1^n$$
$$f(t_1, \ldots, t_n)\sigma_2^0 \ \sim_{\alpha_n} \ f(u_1, \ldots, u_n)\sigma_2^n$$

which completes the proof of Condition 2.

Condition 3. This condition becomes, for all $k = 1, \ldots, n$:

$$\begin{array}{l} f(s_1, \ldots, s_n) \preceq_{\alpha_{k-1}} f(u_1, \ldots, u_n) \ \text{ and } \ \sigma_1^k \preceq_{\alpha_k} \sigma_1^{k-1} \\ g(t_1, \ldots, t_n) \preceq_{\alpha_{k-1}} g(u_1, \ldots, u_n) \ \text{ and } \ \sigma_2^k \preceq_{\alpha_k} \sigma_2^{k-1} \end{array}$$

from which, since $\alpha_k \leq \alpha_{k-1}$ for all $k = 1, \ldots, n$, it follows that:

$$\begin{array}{l} f(s_1, \ldots, s_n) \preceq_{\alpha_n} f(u_1, \ldots, u_n) \ \text{ and } \ \sigma_1^n \preceq_{\alpha_n} \sigma_1^0 \\ g(t_1, \ldots, t_n) \preceq_{\alpha_n} g(u_1, \ldots, u_n) \ \text{ and } \ \sigma_2^n \preceq_{\alpha_n} \sigma_2^0 \end{array}$$

or indifferently, using the same similarity class representative in both cases since $f \sim_{\alpha_n} g$ (because $f \sim_\beta g$ and $\alpha_n \leq \beta$):

$$g(t_1, \ldots, t_n) \preceq_{\alpha_n} f(u_1, \ldots, u_n) \ \text{ and } \ \sigma_2^n \preceq_{\alpha_n} \sigma_2^0$$

which completes the proof of Condition 3, and the proof of Theorem 2.5. □

Example 2.11 Fuzzy generalization with similar functors of same arities — Consider the signature Σ containing $\Sigma_0 = \{a, b, c, d\}$, and $\Sigma_2 = \{f, g\}$, and the closure $\sim$ of the similar pairs $a \sim_{.7} b$, $c \sim_{.6} d$, and $f \sim_{.8} g$. Let us apply the functor-weak generalization axioms and rule Figure 2.13 to $t_1 \stackrel{\text{def}}{=} g(c, d)$, and $t_2 \stackrel{\text{def}}{=} f(a, b)$; that is, let us find term t, substitutions $\sigma_i \in \textsc{Subst}_\mathcal{T}$ $(i = 1, 2)$, and similarity degree α in $[0, 1]$ such that $t\sigma_1 \sim_\alpha g(c, d)$ and $t\sigma_2 \sim_\alpha f(a, b)$. This is expressed as the following fuzzy judgment:

$$\begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix}_1 \vdash \begin{pmatrix} g(c,d) \\ f(a,b) \end{pmatrix} t \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}_\alpha .$$

By Rule **Similarity Functors**, we infer that $t = g(u_1, u_2)$:[21]

$$\begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix}_1 \vdash \begin{pmatrix} g(c,d) \\ f(a,b) \end{pmatrix} g(u_1, u_2) \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}_\alpha$$

which, replaced by the antecedents of Rule **Similarity Functors**, becomes (since $g \sim_{.8} f$):

$$\begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix}_{.8} \vdash \begin{pmatrix} c \\ a \end{pmatrix} \uparrow_{.8} \begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix} u_1 \begin{pmatrix} \sigma_1' \\ \sigma_2' \end{pmatrix}_{\alpha'}, \begin{pmatrix} \sigma_1' \\ \sigma_2' \end{pmatrix}_{\alpha'} \vdash \begin{pmatrix} d \\ b \end{pmatrix} \uparrow_{\alpha'} \begin{pmatrix} \sigma_1' \\ \sigma_2' \end{pmatrix} u_2 \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}_\alpha .$$

Since the prior substitutions of the first judgment are empty, evaluating its fuzzy unapplication (using Expression (2.25) in which $\beta' = \alpha$) yields the sequence:

$$\begin{pmatrix} \emptyset \\ \emptyset \end{pmatrix}_{.8} \vdash \begin{pmatrix} c \\ a \end{pmatrix} u_1 \begin{pmatrix} \sigma_1' \\ \sigma_2' \end{pmatrix}_{\alpha'}, \begin{pmatrix} \sigma_1' \\ \sigma_2' \end{pmatrix}_{\alpha'} \vdash \begin{pmatrix} d \\ b \end{pmatrix} \uparrow_{\alpha'} \begin{pmatrix} \sigma_1' \\ \sigma_2' \end{pmatrix} u_2 \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}_\alpha .$$

[21] This is a non-deterministic choice of a functor's similarity-class representative. We shall always take the left (or upper, in this notation) term's functor. This, of course, will also result in a non-deterministic choice of representative for any term elaborated in generalization modulo functor similarity. The lower the approximation degree, the larger the similarity class.

By Axiom **DISSIMILAR FUNCTORS**, it comes that $u_1 = X_1$, a new variable, and the sequence becomes:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_{.8} \vdash \begin{pmatrix}c\\ a\end{pmatrix} X_1 \begin{pmatrix}\{\ c/X_1\ \}\\ \{\ a/X_1\ \}\end{pmatrix}_{.8}, \begin{pmatrix}\{\ c/X_1\ \}\\ \{\ a/X_1\ \}\end{pmatrix}_{.8} \vdash \begin{pmatrix}d\\ b\end{pmatrix} \uparrow_{.8} \begin{pmatrix}\{\ c/X_1\ \}\\ \{\ a/X_1\ \}\end{pmatrix} u_2 \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha}.$$

The validity of the first fuzzy judgment is thereby established. We proceed with the remaining fuzzy judgment evaluating its fuzzy unapplication using Equation (2.22). Since X_1 is such that $d \sim_{.6} X_1\{c/X_1\} = c$ and $a \sim_{.7} X_1\{b/X_1\} = b$, it verifies the first of the conditions of Equation (2.22). Therefore, the new approximation degree of the judgment is $.8 \wedge .6 \wedge .7 = .6$, and $u_2 = X_1$ so that the judgment becomes:

$$\begin{pmatrix}\{\ c/X_1\ \}\\ \{\ a/X_1\ \}\end{pmatrix}_{.6} \vdash \begin{pmatrix}d\\ b\end{pmatrix} \uparrow_{.6} \begin{pmatrix}\{\ c/X_1\ \}\\ \{\ a/X_1\ \}\end{pmatrix} X_1 \begin{pmatrix}\{\ c/X_1\ \}\\ \{\ a/X_1\ \}\end{pmatrix}_{.6}.$$

This validates the last judgment completing the fuzzy generalization whereby $t = g(X_1, X_1)$ is the least fuzzy generalizer of $t_1 = g(c, d)$, and $t_2 = f(a, b)$ at approximation degree .6 with $\sigma_1 = \{\ c/X_1\ \}$ so that $t\sigma_1 = g(c, c) \sim_{.6} t_1$; and, $\sigma_2 = \{\ a/X_1\ \}$ so that $t\sigma_2 = g(a, a) \sim_{.6} t_2$.

Fuzzy functor/arity-weak generalization

In Figure 2.14, we give a fuzzy version of the generalization rules taking into account mismatches not only in functors, but also in arities; *i.e.*, number and/or order of arguments. We now assume that we are not only given a similarity relation $\sim$: $\Sigma \times \Sigma \to [0, 1]$ on the signature $\Sigma = \cup_{n\geq 0}\Sigma_n$, but also that functors of different arities may be similar with some non-zero similarity degree as specified by a one-to-one argument-position mapping for each pair of so-similar functors associating each argument position of the functor of least arity with a distinct argument position of the functor of larger arity. The only rule among those of Figure 2.13 that differs is the last one (**SIMILAR FUNCTORS**) which is now a pair of rules called **FUNCTOR/ARITY SIMILARITY LEFT** and **FUNCTOR/ARITY SIMILARITY RIGHT** as they account for non-identical correspondence among similar functors's argument positions whether in the left or in the right of the pair of terms to generalize, depending on which side has less arguments. If the arities are the same, the two rules are equivalent (each and all the arguments of the two terms are paired in bijection by a position permutation).

THEOREM 2.6 (FUNCTOR/ARITY-WEAK GENERALIZATION CORRECTNESS) *The fuzzy generalization rules of Figure 2.13 where Rule "***SIMILAR FUNCTORS***" is replaced with the rules in Figure 2.14 are correct.*

> **PROOF** The argument in this proof has exactly the same structure as the argument for the proof of Rule **SIMILAR FUNCTORS** of Figure 2.13. The only difference is that structural induction on a pair of terms with similar functors to generalize is always limited to the largest possible set of pairs of corresponding argument positions as specified by a one-to-one argument map from all the argument positions of the functor of lesser arity to those of the functor of larger arity, rather than the identity on equal cardinality sets of argument positions. Thus, in the following, parts of the proof that are omitted are identical to their corresponding parts in the proof of Rule **SIMILAR FUNCTORS**. Also, for reason of obvious symmetry, we need only provide the detailed proof of correctness of

FUNCTOR/ARITY SIMILARITY LEFT

$$\left[f \approx^{p}_{\beta} g;\ \ \beta > 0;\ \ 0 \leq m \leq n;\ \ \alpha_0 \stackrel{\text{def}}{=} \alpha \wedge \beta\right]$$

$$\frac{\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha_0} \vdash \begin{pmatrix}s_1\\ t_{p(1)}\end{pmatrix} \uparrow_{\alpha_0} \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} u_1 \begin{pmatrix}\sigma_1^1\\ \sigma_2^1\end{pmatrix}_{\alpha_1} \cdots \begin{pmatrix}\sigma_1^{m-1}\\ \sigma_2^{m-1}\end{pmatrix}_{\alpha_{m-1}} \vdash \begin{pmatrix}s_m\\ t_{p(m)}\end{pmatrix} \uparrow_{\alpha_{m-1}} \begin{pmatrix}\sigma_1^{m-1}\\ \sigma_2^{m-1}\end{pmatrix} u_m \begin{pmatrix}\sigma_1^{m}\\ \sigma_2^{m}\end{pmatrix}_{\alpha_m}}{\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha} \vdash \begin{pmatrix}f(s_1,\ldots,s_m)\\ g(t_1,\ldots,t_n)\end{pmatrix} f(u_1,\ldots,u_m) \begin{pmatrix}\sigma_1^{m}\\ \sigma_2^{m}\end{pmatrix}_{\alpha_m}}$$

FUNCTOR/ARITY SIMILARITY RIGHT

$$\left[g \approx^{p}_{\beta} f;\ \ \beta > 0;\ \ 0 \leq n \leq m;\ \ \alpha_0 \stackrel{\text{def}}{=} \alpha \wedge \beta\right]$$

$$\frac{\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha_0} \vdash \begin{pmatrix}s_{p(1)}\\ t_1\end{pmatrix} \uparrow_{\alpha_0} \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} u_1 \begin{pmatrix}\sigma_1^1\\ \sigma_2^1\end{pmatrix}_{\alpha_1} \cdots \begin{pmatrix}\sigma_1^{n-1}\\ \sigma_2^{n-1}\end{pmatrix}_{\alpha_{n-1}} \vdash \begin{pmatrix}s_{p(n)}\\ t_n\end{pmatrix} \uparrow_{\alpha_{n-1}} \begin{pmatrix}\sigma_1^{n-1}\\ \sigma_2^{n-1}\end{pmatrix} u_n \begin{pmatrix}\sigma_1^{n}\\ \sigma_2^{n}\end{pmatrix}_{\alpha_n}}{\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha} \vdash \begin{pmatrix}f(s_1,\ldots,s_m)\\ g(t_1,\ldots,t_n)\end{pmatrix} g(u_1,\ldots,u_n) \begin{pmatrix}\sigma_1^{n}\\ \sigma_2^{n}\end{pmatrix}_{\alpha_n}}$$

Figure 2.14: Functor/arity-weak generalization rules

Rule **FUNCTOR/ARITY SIMILARITY LEFT**. The proof of correctness of Rule **FUNCTOR/ARITY SIMILARITY RIGHT** is the pointwise similar dual argument in the other direction.

Considering Rule **FUNCTOR/ARITY SIMILARITY LEFT**, as required by Definition 2.12, we must show that if all the fuzzy judgments in the numerator are valid, then the fuzzy judgment in the denominator is valid too. Since the proofs of Condition 1 and Condition 3 are the same for equal-arity functor similarity, we need only provide a proof of Condition 2 of Definition 2.12. Let us proceed by induction on the argument-position number k, for $k = 1, \ldots, m$, where m is the arity of f (the first of the two terms' functor, with the same or a smaller arity as required by the side condition).

For $m = 0$, this rule becomes Axiom (2.26). This fuzzy judgment is trivially valid at all approximation degrees: Condition 2 of Definition 2.11 becomes the reflexive similarity $c \approx_\alpha c$ and Condition 3 becomes the conjunction $c \preceq_\alpha c$ and $\sigma_i \preceq_\alpha \sigma_i$, for $i = 1, 2$. Thus, this verifies both Condition 2 and Condition 3 for $m = 0$.

For $m > 0$, for each argument-position $k = 1, \ldots, m$, a fuzzy judgment in the rule's antecedent is of the form:

$$\begin{pmatrix}\sigma_1^{k-1}\\ \sigma_2^{k-1}\end{pmatrix}_{\alpha_{k-1}} \vdash \begin{pmatrix}s_k\\ t_{p(k)}\end{pmatrix} \uparrow_{\alpha_{k-1}} \begin{pmatrix}\sigma_1^{k-1}\\ \sigma_2^{k-1}\end{pmatrix} u_k \begin{pmatrix}\sigma_1^{k}\\ \sigma_2^{k}\end{pmatrix}_{\alpha_k};$$

that is, the form of Expression (2.24), whose formal meaning is given as Expression (2.25), which

in the above case is equivalent to:

$$\begin{pmatrix} v_1^k \\ v_2^k \end{pmatrix}_{\beta_k} \stackrel{\text{def}}{=} \begin{pmatrix} s_k \\ t_{p(k)} \end{pmatrix} \uparrow_{\alpha_{k-1}} \begin{pmatrix} \sigma_1^{k-1} \\ \sigma_2^{k-1} \end{pmatrix} \text{ and } \begin{pmatrix} \sigma_1^{k-1} \\ \sigma_2^{k-1} \end{pmatrix}_{\beta_k} \vdash \begin{pmatrix} v_1^k \\ v_2^k \end{pmatrix} u_k \begin{pmatrix} \sigma_1^k \\ \sigma_2^k \end{pmatrix}_{\alpha_k}$$

for some β_k s.t. $\alpha_{k-1} \leq \beta_k \leq \alpha_k$. Let us now assume that all the fuzzy judgment in the rule's antecedent are valid. That is, for $k = 1, \ldots, m$ (defining $\alpha_0 \stackrel{\text{def}}{=} \alpha \wedge \beta$), for $i = 1, 2$:

$$u_k \sigma_i^k \approx_{\alpha_k} v_i^k \sigma_i^{k-1} \tag{2.29}$$

and (defining $\sigma_i^0 \stackrel{\text{def}}{=} \sigma_i$):

$$v_i^k \preceq_\alpha u_k \text{ and } \sigma_i^k \preceq_\beta \sigma_i^{k-1}. \tag{2.30}$$

By Equation (2.22), this means that for all $k = 1, \ldots, m$, v_1^k, v_2^k, and α_k are defined by:

$$\begin{pmatrix} v_1^k \\ v_2^k \end{pmatrix}_{\alpha_k} \stackrel{\text{def}}{=} \begin{cases} \begin{pmatrix} X \\ X \end{pmatrix}_{\alpha_{k-1} \wedge \beta_1^k \wedge \beta_2^k} & \text{if } \exists\, X \in \mathcal{V} \text{ s.t. } \begin{pmatrix} s_k & \approx_{\beta_1^k} & X\sigma_1^{k-1} \\ t_{p(k)} & \approx_{\beta_2^k} & X\sigma_2^{k-1} \end{pmatrix}; \\ \begin{pmatrix} s_k \\ t_{p(k)} \end{pmatrix}_{\alpha_{k-1}} & \text{otherwise.} \end{cases}$$

for some β_1^k and β_2^k in $(0, 1]$. In other words, for each $k = 1, \ldots, m$, there are two cases:

1. $s_k \approx_{\beta_1^k} X\sigma_1^{k-1}$ and $t_{p(k)} \approx_{\beta_2^k} X\sigma_2^{k-1}$ for some variable X; then, by Axiom **FUZZY EQUAL VARIABLES**, we must have $\alpha_k = \alpha_{k-1} \wedge \beta_1^k \wedge \beta_2^k$, $u_k = X$, and $\sigma_i^k = \sigma_i^{k-1}$ for $i = 1, 2$; thus, $\alpha_k \leq \alpha_{k-1}$ and Similarity (2.29) becomes $u_k \sigma_i^k \approx_{\alpha_k} X\sigma_i^{k-1}$ So that:

$$\begin{aligned} s_k\sigma_1^{k-1} &\approx_{\alpha_k} X\sigma_1^{k-1}\sigma_1^{k-1} = X\sigma_1^{k-1} = X\sigma_1^k \approx_{\alpha_k} u_k\sigma_1^k, \\ t_{p(k)}\sigma_2^{k-1} &\approx_{\alpha_k} X\sigma_2^{k-1}\sigma_2^{k-1} = X\sigma_2^{k-1} = X\sigma_2^k \approx_{\alpha_k} u_k\sigma_2^k. \end{aligned}$$

2. There is no such variable X; in which case, $\alpha_k = \alpha_{k-1}$ and Similarity (2.29) becomes:

$$\begin{aligned} s_k\sigma_1^{k-1} &\approx_{\alpha_k} u_k\sigma_1^k, \\ t_{p(k)}\sigma_2^{k-1} &\approx_{\alpha_k} u_k\sigma_2^k. \end{aligned}$$

Thus, by the only non-identical transformation relating prior and posteriors substitutions in the axioms, for any argument position k, $1 \leq k \leq m$, we have:

$$\sigma_i^k \approx_{\alpha_k} \sigma_i^0\{\tau_1/X_1\}\ldots\{\tau_\ell/X_\ell\}$$

where each of the variables $X_1 \ldots X_\ell$, with $0 \leq \ell$, is a variable possibly introduced in proving the validity of the fuzzy judgment corresponding to some argument position preceding k. Therefore, since for any argument position k, $1 \leq k \leq m$:

1. σ_i^k affects only a *new* variable introduced in one of the axioms verifying the validity of the subterm at argument position k; and,
2. such a newly introduced variable now occurring in u_k is always instantiated by the same term;

it comes that, at approximation degree α_k:

$$s_k\sigma_1^0 \quad \approx_{\alpha_k} s_k\sigma_1^1 \quad \approx_{\alpha_k} \ldots \approx_{\alpha_k} s_k\sigma_1^{k-1}$$
$$t_{p(k)}\sigma_2^0 \approx_{\alpha_k} t_{p(k)}\sigma_2^1 \approx_{\alpha_k} \ldots \approx_{\alpha_k} t_{p(k)}\sigma_2^{k-1}$$

as well as, at approximation degree α_m:

$$u_k\sigma_1^k \approx_{\alpha_m} u_k\sigma_1^{k+1} \approx_{\alpha_m} \ldots \approx_{\alpha_m} u_k\sigma_1^m$$
$$u_k\sigma_2^k \approx_{\alpha_m} u_k\sigma_2^{k+1} \approx_{\alpha_m} \ldots \approx_{\alpha_m} u_k\sigma_2^m$$

This means that in both cases we have, for all $k = 1, \ldots, m$:

$$s_k\sigma_1^0 \quad \approx_{\alpha_m} \quad u_k\sigma_1^m$$
$$t_{p(k)}\sigma_2^0 \quad \approx_{\alpha_m} \quad u_k\sigma_2^m.$$

Therefore, for $k = m$:

$$f(s_1, \ldots, s_m)\sigma_1^0 \quad \approx_{\alpha_m} \quad f(u_1, \ldots, u_m)\sigma_1^m$$
$$f(t_{p(1)}, \ldots, t_{p(m)})\sigma_2^0 \quad \approx_{\alpha_m} \quad f(u_1, \ldots, u_m)\sigma_2^m$$

which completes the proof of Condition 2 of Theorem 2.6, and that of the theorem because of the facts stated at the outset regarding all other cases each of whose proof is identical to when arities are equal. □

Example 2.12 Fuzzy generalization with similar functors of different arities — Let us take again the functor signature of Example 2.9 where $\{a, b, c, d\} \subseteq \Sigma_0$, $\{f, g, \ell\} \subseteq \Sigma_2$, and $\{h\} \subseteq \Sigma_3$, with similarity defined as the reflexive symmetric transitive closure of the following pairs, along with their argument alignment maps: $a \approx_{.7} b$, $c \approx_{.6} d$, $f \approx_{.9}^{\{1\mapsto 2, 2\mapsto 1\}} g$ and $g \approx_{.9}^{\{1\mapsto 2, 2\mapsto 1\}} f$, and $\ell \approx_{.8}^{\{1\mapsto 2, 2\mapsto 3\}} h$ (and equal similarity degree for symmetric entries). With this signature and similarity, let us try to find the fuzzy generalization of $t_1 \stackrel{\text{def}}{=} h(X, g(Y, b), f(Y, c))$, and $t_2 \stackrel{\text{def}}{=} \ell(f(a, Z), g(d, c))$.

Thus we need to us find the most general term $t \in \mathcal{T}$ along with two most general substitutions $\sigma_i :\in \textsc{Subst}_{\mathcal{T}}$ $(i = 1, 2)$, and the maximal similarity degree $\alpha \in [0, 1]$, such that $t\sigma_1 \approx_\alpha^{\mathcal{T}} h(X, g(Y, b), f(Y, c))$ and $t\sigma_2 \approx_\alpha^{\mathcal{T}} \ell(f(a, Z), g(d, c))$; that is, solve the following fuzzy generalization constraint problem:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_1 \vdash \begin{pmatrix}h(X, g(Y, b), f(Y, c))\\ \ell(f(a, Z), g(d, c))\end{pmatrix} t \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_\alpha .$$

Rule **Functor/Arity Similarity Right** entails $t = \ell(u_1, u_2)$, and because $\ell \approx_{.8}^{\{1\mapsto 2, 2\mapsto 3\}} h$:

- u_1 is the fuzzy generalization of $\begin{pmatrix}g(Y, b)\\ g(d, c)\end{pmatrix} \uparrow_{.8} \begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}$; that is, of $g(Y, b)$ and $g(d, c)$, such that:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_{.8} \vdash \begin{pmatrix}g(Y, b)\\ g(d, c)\end{pmatrix} u_1 \begin{pmatrix}\sigma_1^1\\ \sigma_2^1\end{pmatrix}_{\alpha_1} ;$$

i.e., $u_1 = g(v_1, v_2)$ by Rule **FUNCTOR/ARITY SIMILARITY LEFT** with $g \approx_1^{\{1\mapsto 1, 2\mapsto 2\}} g$, s.t.:[22]

- $v_1 = U$ since by Rule **FUZZY VARIABLE-TERM**:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_{.8} \vdash \begin{pmatrix}Y\\ d\end{pmatrix} U \begin{pmatrix}\{\, Y/U \,\}\\ \{\, d/U \,\}\end{pmatrix}_{.8};$$

- $v_2 = V$ since with $b \not\approx c$ and by **DISSIMILAR FUNCTORS**:

$$\begin{pmatrix}\{\, Y/U \,\}\\ \{\, d/U \,\}\end{pmatrix}_{.8} \vdash \begin{pmatrix}b\\ c\end{pmatrix} V \begin{pmatrix}\{\, Y/U, b/V \,\}\\ \{\, d/U, c/V \,\}\end{pmatrix}_{.8};$$

and so $\sigma_1^1 = \{\, Y/U, b/V \,\}$, $\sigma_2^1 = \{\, d/U, c/V \,\}$, and $\alpha_1 = .8$; that is:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_{.8} \vdash \begin{pmatrix}g(Y,b)\\ g(d,c)\end{pmatrix} g(U,V) \begin{pmatrix}\{\, Y/U, b/V \,\}\\ \{\, d/U, c/V \,\}\end{pmatrix}_{.8};$$

- u_2 is the fuzzy generalization of $\begin{pmatrix}f(Y,c)\\ f(a,Z)\end{pmatrix} \uparrow_{.8} \begin{pmatrix}\{\, Y/U, b/V \,\}\\ \{\, d/U, c/V \,\}\end{pmatrix}$; that is, of $f(Y,c)$ and $f(a,Z)$, s.t. $u_2 = f(w_1, w_2)$ by Rule **FUNCTOR/ARITY SIMILARITY LEFT** with $f \approx_1^{\{1\mapsto 1, 2\mapsto 2\}} f$ (or Rule **FUNCTOR/ARITY SIMILARITY RIGHT**):

$$\begin{pmatrix}\{\, Y/U, b/V \,\}\\ \{\, d/U, c/V \,\}\end{pmatrix}_{.8} \vdash \begin{pmatrix}f(Y,c)\\ f(a,Z)\end{pmatrix} f(w_1, w_2) \begin{pmatrix}\sigma_1^2\\ \sigma_2^2\end{pmatrix}_{\alpha_2};$$

where:

- by Rule **FUZZY VARIABLE-TERM**:

$$\begin{pmatrix}\{\, Y/U, b/V \,\}\\ \{\, d/U, c/V \,\}\end{pmatrix}_{.8} \vdash \begin{pmatrix}Y\\ a\end{pmatrix} W \begin{pmatrix}\{\, Y/U, b/V, Y/W \,\}\\ \{\, d/U, c/V, a/W \,\}\end{pmatrix}_{.8};$$

  so $w_1 = W$; and,

- by Rule **FUZZY VARIABLE-TERM**:

$$\begin{pmatrix}\{\, Y/U, b/V, Y/W \,\}\\ \{\, d/U, c/V, a/W \,\}\end{pmatrix}_{.8} \vdash \begin{pmatrix}c\\ Z\end{pmatrix} C \begin{pmatrix}\{\, Y/U, b/V, Y/W, c/C \,\}\\ \{\, d/U, c/V, a/W, Z/C \,\}\end{pmatrix}_{.8};$$

  so $w_2 = C$;

thus $\sigma_1^2 = \{\, Y/U, b/V, Y/W, c/C \,\}$, $\sigma_2^2 = \{\, d/U, c/V, a/W, Z/C \,\}$, and $\alpha_2 = .8$.

[22] Note that, for two terms of equal arity, using either **FUNCTOR/ARITY SIMILARITY LEFT** or **FUNCTOR/ARITY SIMILARITY RIGHT** is always equivalent. As explained in Figure 2.9, similar functors of equal arity have a pair of mutually inverse bijections map corresponding argument positions in each term with the other's. Therefore, choosing any of the two functors for the generalized term yields a term in the same similarity class. This is true *a fortiori* when the two functors are equal, as in this case, when the similarity degree is 1 and the argument map is the identity.

Therefore,

$$\binom{\emptyset}{\emptyset}_1 \vdash \binom{h(X, g(Y,b), f(Y,c))}{\ell(f(a,Z), g(d,c))} \ell(f(W,C), g(U,V)) \binom{\{\, Y/U, b/V, Y/W, c/C \,\}}{\{\, d/U, c/V, a/W, Z/C \,\}}_{.8};$$

that is, $t = \ell(f(W,C), g(U,V))$, with:

$$\sigma_1 = \{\, Y/U, b/V, Y/W, c/C \,\} \text{ and } \sigma_2 = \{\, d/U, c/V, a/W, Z/C \,\}$$

and $\alpha = .8$, since:

$$t_1 = t\sigma_1 = \ell(f(Y,c), g(Y,b)) \approx^{\mathcal{T}}_{.8} h(X, g(Y,b), f(Y,c)) \text{ and } t_2 = t\sigma_2 = \ell(f(a,Z), g(d,c)).$$

Example 2.13 Example 2.12 with more expressive symbols — Let us again, as we did in Example 2.10 (gift-shop Prolog database), give more expressive names to functors of Example 2.12:

- $a/0 \stackrel{\text{def}}{=} \texttt{violet}, b/0 \stackrel{\text{def}}{=} \texttt{lilac}, c/0 \stackrel{\text{def}}{=} \texttt{chocolate}, d/0 \stackrel{\text{def}}{=} \texttt{candy}$,
- $f/2 \stackrel{\text{def}}{=} \texttt{pair}, g/2 \stackrel{\text{def}}{=} \texttt{couple}$,
- $\ell/2 \stackrel{\text{def}}{=} \texttt{small-gift-bag}, h/3 \stackrel{\text{def}}{=} \texttt{small-gift-box}$,

with the following similarity degrees and argument maps,:

- $\texttt{violet} \approx_{.7} \texttt{lilac}$,
- $\texttt{chocolate} \approx_{.6} \texttt{candy}$,
- $\texttt{pair} \approx^{\{1\mapsto 2, 2\mapsto 1\}}_{.9} \texttt{couple}$ and $\texttt{couple} \approx^{\{1\mapsto 2, 2\mapsto 1\}}_{.9} \texttt{pair}$,
- $\texttt{small-gift-bag} \approx^{\{1\mapsto 2, 2\mapsto 3\}}_{.8} \texttt{small-gift-box}$.

With these functors symbols, the generalization problem of Example 2.9 appears now with the terms:

$$\begin{aligned} t_1 &\stackrel{\text{def}}{=} \texttt{small-gift-box}(X \\ &\qquad\qquad , \texttt{couple}(Y, \texttt{lilac}) \\ &\qquad\qquad , \texttt{pair}(Y, \texttt{chocolate}) \\ &\qquad\qquad) \\ t_2 &\stackrel{\text{def}}{=} \texttt{small-gift-bag}(\texttt{pair}(\texttt{violet}, Z) \\ &\qquad\qquad , \texttt{couple}(\texttt{candy}, \texttt{chocolate}) \\ &\qquad\qquad) \end{aligned}$$

which are the results of applying the substitutions:

$$\sigma_1 = \{\, Y/U, \texttt{lilac}/V, Y/W, \texttt{chocolate}/C \,\}$$

and

$$\sigma_2 = \{\, \texttt{candy}/U, \texttt{chocolate}/V, \texttt{violet}/W, Z/C \,\}$$

to the least fuzzy generalization:

$$t \stackrel{\text{def}}{=} \texttt{small-gift-bag}\big(\ \texttt{pair}(W,C) \\ ,\ \texttt{couple}(U,V) \\ \big)$$

obtained after normalization with a similarity degree .8.

Example 2.14 Fuzzy generalization with similar functors of different arities—2nd example — Consider the signature Σ containing $\Sigma_0 = \{a,b,c,d\}$, $\Sigma_2 = \{f,g,l\}$, and $\Sigma_3 = \{h\}$, and the closure $\sim$ of the similar pairs $a \sim_{.7} c$, $c \sim_{.6} d$, $f \sim_{.8} g$, and $l \sim_{.9} h$. Let us take all argument-position mappings as the default (identity on least-arity set). Let us apply the fuzzy generalization axioms of Figure 2.13 and the rule of Figure 2.14 to $t_1 \stackrel{\text{def}}{=} h(g(b,Y),f(Y,c),V)$, and $t_2 \stackrel{\text{def}}{=} l(f(a,Z),g(c,d))$; that is, let us find term t, substitutions $\sigma_i \in \textsc{Subst}_{\mathcal{T}}$ $(i = 1,2)$, and similarity degree α in $[0,1]$, such that $t\sigma_1 \sim_\alpha h(g(b,Y),f(Y,c),V)$ and $t\sigma_2 \sim_\alpha l(f(a,Z),g(c,d))$. This is expressed as the following fuzzy judgment:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_1 \vdash \begin{pmatrix}h(g(b,Y),f(Y,c),V)\\ l(f(a,Z),g(c,d))\end{pmatrix} t \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_\alpha .$$

By Rule **Functor/Arity Similarity Right**, we can infer that $t = l(u_1,u_2)$:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_1 \vdash \begin{pmatrix}h(g(b,Y),f(Y,c),V)\\ l(f(a,Z),g(c,d))\end{pmatrix} l(u_1,u_2) \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_\alpha$$

which, when replaced by the rule's antecedents, since $h \sim_{.9} l$ and $1 \wedge .9 = .9$, becomes the sequence:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_{.9} \vdash \begin{pmatrix}g(b,Y)\\ f(a,Z)\end{pmatrix} \uparrow_{.9} \begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix} u_1 \begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix}_{\alpha'}, \begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix}_{\alpha'} \vdash \begin{pmatrix}f(Y,c)\\ g(c,d)\end{pmatrix} \uparrow_{\alpha'} \begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix} u_2 \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_\alpha .$$

By evaluating the fuzzy unapplication in its first judgment, this sequence becomes:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_{.9} \vdash \begin{pmatrix}g(b,Y)\\ f(a,Z)\end{pmatrix} u_1 \begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix}_{\alpha'}, \begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix}_{\alpha'} \vdash \begin{pmatrix}f(Y,c)\\ g(c,d)\end{pmatrix} \uparrow_{\alpha'} \begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix} u_2 \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_\alpha .$$

By Rule **Functor/Arity Similarity Left**,[23] it comes that $u_1 = g(u_3,u_4)$ and, since $g \sim_{.8} f$ and $.9 \wedge .8 = .8$, the sequence becomes:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_{.8} \vdash \begin{pmatrix}b\\ a\end{pmatrix} \uparrow_{.8} \begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix} u_3 \begin{pmatrix}\sigma_1''\\ \sigma_2''\end{pmatrix}_{\alpha''}, \begin{pmatrix}\sigma_1''\\ \sigma_2''\end{pmatrix}_{\alpha''} \vdash \begin{pmatrix}Y\\ Z\end{pmatrix} \uparrow_{\alpha''} \begin{pmatrix}\sigma_1''\\ \sigma_2''\end{pmatrix} u_4 \begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix}_{\alpha'},$$
$$\begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix}_{\alpha'} \vdash \begin{pmatrix}f(Y,c)\\ g(c,d)\end{pmatrix} \uparrow_{\alpha'} \begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix} u_2 \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_\alpha .$$

By evaluating the fuzzy unapplication in the first judgment, and using Rule **Functor/Arity Similarity Left** in the 0-arity case as Axiom (2.26), since $b \sim_{.7} a$ and $.8 \wedge .7 = .7$, we have $u_3 = b$, and the sequence becomes:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_{.7} \vdash \begin{pmatrix}b\\ a\end{pmatrix} b \begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_{.7}, \begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_{.7} \vdash \begin{pmatrix}Y\\ Z\end{pmatrix} \uparrow_{.7} \begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix} u_4 \begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix}_{\alpha'},$$

[23] Since f and g have equal arities, we could also use Rule **Functor/Arity Similarity Right**. This would end in an equivalent final result, modulo functor similarities at the final approximation degree. In the remainder of this example, we shall omit making this remark, and choose the left rule over the right for equal-arity functors.

$$\begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix}_{\alpha'} \vdash \begin{pmatrix} f(Y,c)\\ g(c,d)\end{pmatrix} \uparrow_{\alpha'} \begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix} u_2 \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha}.$$

The validity of the first fuzzy judgment is thereby established. We proceed with the remaining sequence of fuzzy judgments evaluating the fuzzy unapplication in the first of its judgments, which sets $\alpha' = .7$:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_{.7} \vdash \begin{pmatrix} Y\\ Z\end{pmatrix} u_4 \begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix}_{.7}, \begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix}_{.7} \vdash \begin{pmatrix} f(Y,c)\\ g(c,d)\end{pmatrix} \uparrow_{.7} \begin{pmatrix}\sigma_1'\\ \sigma_2'\end{pmatrix} u_2 \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha}.$$

By Axiom **FUZZY VARIABLE-TERM**, we infer from this that $u_4 = X_1$, a new variable, and the judgments become:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_{.7} \vdash \begin{pmatrix} Y\\ Z\end{pmatrix} X_1 \begin{pmatrix}\{\, Y/X_1 \,\}\\ \{\, Z/X_1 \,\}\end{pmatrix}_{.7},$$

$$\begin{pmatrix}\{\, Y/X_1 \,\}\\ \{\, Z/X_1 \,\}\end{pmatrix}_{.7} \vdash \begin{pmatrix} f(Y,c)\\ g(c,d)\end{pmatrix} \uparrow_{.7} \begin{pmatrix}\{\, Y/X_1 \,\}\\ \{\, Z/X_1 \,\}\end{pmatrix} u_2 \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha}.$$

The validity of the first fuzzy judgment of the above sequence is thereby established. We proceed with the remainder evaluating the fuzzy unapplication in the first of its judgments, which returns the same pair of terms with the similarity degree kept at .7:

$$\begin{pmatrix}\{\, Y/X_1 \,\}\\ \{\, Z/X_1 \,\}\end{pmatrix}_{.7} \vdash \begin{pmatrix} f(Y,c)\\ g(c,d)\end{pmatrix} u_2 \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha}.$$

and by Rule **FUNCTOR/ARITY SIMILARITY LEFT** with $u_2 = f(u_5, u_6)$, this becomes:

$$\begin{pmatrix}\{\, Y/X_1 \,\}\\ \{\, Z/X_1 \,\}\end{pmatrix}_{.7} \vdash \begin{pmatrix} Y\\ c\end{pmatrix} \uparrow_{.7} \begin{pmatrix}\{\, Y/X_1 \,\}\\ \{\, Z/X_1 \,\}\end{pmatrix} u_5 \begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix}_{\beta}, \begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix}_{\beta} \vdash \begin{pmatrix} c\\ d\end{pmatrix} \uparrow_{\beta} \begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix} u_6 \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha}.$$

Evaluating the fuzzy unapplication gives $\beta = .7$:

$$\begin{pmatrix}\{\, Y/X_1 \,\}\\ \{\, Z/X_1 \,\}\end{pmatrix}_{.7} \vdash \begin{pmatrix} Y\\ c\end{pmatrix} u_5 \begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix}_{.7}, \begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix}_{.7} \vdash \begin{pmatrix} c\\ d\end{pmatrix} \uparrow_{.7} \begin{pmatrix}\theta_1\\ \theta_2\end{pmatrix} u_6 \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha}.$$

and by Axiom **FUZZY VARIABLE-TERM**, we infer from this that $u_5 = X_2$, a new variable, which yields:

$$\begin{pmatrix}\{\, Y/X_1 \,\}\\ \{\, Z/X_1 \,\}\end{pmatrix}_{.7} \vdash \begin{pmatrix} Y\\ c\end{pmatrix} X_2 \begin{pmatrix}\{\, Y/X_1, Y/X_2 \,\}\\ \{\, Z/X_1, c/X_2 \,\}\end{pmatrix}_{.7},$$

$$\begin{pmatrix}\{\, Y/X_1, Y/X_2 \,\}\\ \{\, Z/X_1, c/X_2 \,\}\end{pmatrix}_{.7} \vdash \begin{pmatrix} c\\ d\end{pmatrix} \uparrow_{.7} \begin{pmatrix}\{\, Y/X_1, Y/X_2 \,\}\\ \{\, Z/X_1, c/X_2 \,\}\end{pmatrix} u_6 \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha},$$

and establishes the penultimate judgment. The last remaining judgment, after evaluating its fuzzy unapplication, since $c \sim_{.6} d$ and $.7 \wedge .6 = .6$, is:

$$\begin{pmatrix}\{\, Y/X_1, Y/X_2 \,\}\\ \{\, Z/X_1, c/X_2 \,\}\end{pmatrix}_{.6} \vdash \begin{pmatrix} c\\ d\end{pmatrix} u_6 \begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha},$$

for which Axiom **FUZZY VARIABLE-TERM** allows us to infer that $u_6 = c$ and $\alpha = .6$:

$$\begin{pmatrix}\{\, Y/X_1, Y/X_2 \,\}\\ \{\, Z/X_1, c/X_2 \,\}\end{pmatrix}_{.6} \vdash \begin{pmatrix} c\\ d\end{pmatrix} c \begin{pmatrix}\{\, Y/X_1, Y/X_2 \,\}\\ \{\, Z/X_1, c/X_2 \,\}\end{pmatrix}_{.6}.$$

This validates the last judgment and completes the fuzzy generalization whereby $t = l(g(b, X_1), f(X_2, c))$ is the least fuzzy generalizer of $t_1 = h(g(b, Y), f(Y, c), V)$ and $t_2 = l(f(a, Z), g(c, d))$ at approximation degree .6, with:

- $\sigma_1 = \{\ Y/X_1, Y/X_2\ \}$ so that $t\sigma_1 = l(g(b, Y), f(Y, c)) \sim_{.6} t_1$; and,
- $\sigma_2 = \{\ Z/X_1, c/X_2\ \}$ so that $t\sigma_2 = l(g(b, Z), f(c, c)) \sim_{.6} t_2$.

2.6.3 Partial maps

In the foregoing sections, we gave declarative presentations for three lattice structures over $\mathcal{FOT}$s (one crisp and two fuzzy) in the form of axioms and rules. These axioms and rules specify the six corresponding dual lattice operations as constraints in these algebraic structures. An executable semantics for each operation is thus obtained for free as constraint solving. The latter may be summarized as follows, for each of the three $\mathcal{FOT}$ lattice structures:[24]

1. **for conventional signatures** (no operator similarity besides identity):
   - — we presented the declarative $\mathcal{FOT}$ unification rules due to Herbrand and to Martelli & Montanari;
   - ✓ we provided a declarative constraint-based version of generalization equivalent to the original procedural methods due to Reynolds and Plotkin;
2. **for signatures with "weak" similarity** (all pairs of similar operators have the same number and order of arguments):
   - — we presented "weak" fuzzy unification as constraint normalization using declarative rules due to Maria Sessa;
   - ✓ we provided a "weak" fuzzy generalization as a constraint solving using a declarative specification for the dual operation of Sessa's "weak" unification;
3. **for signatures with possibly misaligned similarity** (similar operators possibly with different number or order of arguments):
   - ✓ we extended the above constraint-driven declarative "weak" fuzzy unification to $\mathcal{FOT}$s with possible different/mixed arities;
   - ✓ we extended the above constraint-driven declarative "weak" fuzzy generalization of $\mathcal{FOT}$s with possible different/mixed arities.

This last pair of lattice operations on $\mathcal{FOT}$ modulo a similarity involving operators with misaligned or unordered arguments extends the previous pair of "weak" operations given argument maps specified for similar operators. That is, a similar pair of functors has a similarity degree as

[24] We use the "✓" check symbol to indicate what items are contribution of this work.

well as an injective argument-realigning map for each pair of operators in the signature. If unspecified, this map is the identity from the term with less arguments to the one with more arguments. In effect, this third lattice of $\mathcal{FOT}$s is closer to permit Fuzzy-Logic Programming querying with misaligned databases, or more generally Information Retrieval (using fuzzy unification) and Approximate Knowledge Acquisition (using fuzzy generalization) over heterogeneous but similar data models. In the remainder of this section, we will develop yet another lattice of $\mathcal{FOT}$s which is even closer to such models in that it allows an even more expressive similarity between functors than the ones we presented above, all of which are again special cases of this more generic $\mathcal{FOT}$ similarity.

The third $\mathcal{FOT}$ lattice above, being the most expressive of the three, tolerates similarity pairs of functors with different arities and assumes given a similarity matrix $\approx$ indexed by the functor signature Σ and for each pair of functors f/m and g/n such that $0 \leq m \leq n$, an injective map $\mu_{fg} : \{1, \ldots, m\} \to \{1, \ldots, n\}$ verifying consistency conditions (2.16), (2.17), and (2.18) on page 27. However, this lattice, although less constraining than the former, still requires that μ_{fg} be a *total function* map associating to *each* argument position in $\{1, \ldots, m\}$ a unique argument position in $\{1, \ldots, n\}$. This constraint, as we see next, may be relaxed to accommodate similar functors with *partial* argument alignments.

Example 2.15 Partial-map non-aligned similar functors — Consider two functors $\mathtt{foo} \in \Sigma_5$ and $\mathtt{bar} \in \Sigma_4$, and a non-zero approximation degree $\alpha \in (0, 1]$, where this similarity may be homomorphically extended from these functors to terms they construct *only* when, at this approximation degree α, $\mathtt{foo}$'s 3^{rd} argument is similar to $\mathtt{bar}$'s 4^{th} argument, and when $\mathtt{foo}$'s 4^{th} argument is similar to $\mathtt{bar}$'s 2^{nd} argument. This means that there are two mutually inverse partial bijective maps between the argument positions of functors $\mathtt{foo}$ and $\mathtt{bar}$ specifying which argument position of one corresponds to which unique argument position of the other; *viz.*, $\mu^{\alpha}_{\mathtt{foo,bar}} : \{3, 4\} \to \{1, 2, 3, 4\} = \{3 \mapsto 4, 4 \mapsto 2\}$ and $\mu^{\alpha}_{\mathtt{bar,foo}} : \{2, 4\} \to \{1, 2, 3, 4, 5\} = \{2 \mapsto 4, 4 \mapsto 3\}$. We shall denote such a partial non-aligned functor similarity with the symmetric pairs $\mathtt{foo} \approx^{\mu^{\alpha}_{\mathtt{foo,bar}}}_{\alpha} \mathtt{bar}$ and $\mathtt{bar} \approx^{\mu^{\alpha}_{\mathtt{bar,foo}}}_{\alpha} \mathtt{foo}$ as in the $\mathcal{FOT}$ similarity Expression (2.31).

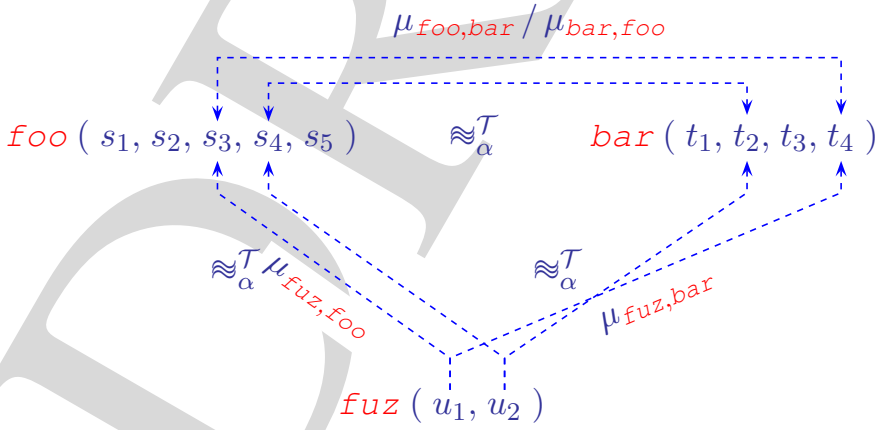
$$\mathtt{foo}\,(\,s_1,\,s_2,\,s_3,\,s_4,\,s_5) \;\; \overset{\mu_{\mathtt{foo,bar}}\,/\,\mu_{\mathtt{bar,foo}}}{\approx^{\mathcal{T}}_{\alpha}} \;\; \mathtt{bar}\,(\,t_1,\,t_2,\,t_3,\,t_4\,) \tag{2.31}$$

The formalism we have developed in the previous sections cannot apply for such non-aligned similar functors with only partial argument-position maps because the assumptions we made for the unification rules of Figure 2.11 to be correct do not hold. Indeed, these rules work because whenever an equation between two constructed terms has a term of lesser arity on the right, Rule **FUZZY EQUATION ORIENTATION** swaps its sides into an equation with the lesser-arity term on the left. And, for a such an equation as the latter, Rule **FUZZY NON-ALIGNED-ARGUMENT TERM DECOMPOSITION** replaces this equation only by equations between subterms at corresponding positions, taking *all* arguments of the lesser-arity *from position* 1, and *all the way up to its full arity*. This is no longer possible with partial maps between two similar functors' non-aligned argument positions. Indeed, a lesser arity functor's partial argument maps may not

be defined for argument position 1, nor for consecutive positions, nor up to the functor's arity. For the same reason, the fuzzy generalization rules of Figure 2.14 will not apply either. Indeed, both Rule **FUNCTOR/ARITY SIMILARITY LEFT** and **FUNCTOR/ARITY SIMILARITY RIGHT** specify the least generalizer to be constructed using the least-arity functor and the generalizers of all its arguments; this, clearly, is no longer possible with partial argument-position maps between the substerms of non-aligned similar functors.

However, in some specific situations, it may be possible to come back to the previously studied $\mathcal{FOT}$ lattice—which requires that in any equation between two constructed terms one of the two terms always be a lesser-arity functor's with a total injective map from the set of all its argument positions to a subset of the larger-arity functor's term's set of argument positions. Indeed, the justification for the need of reorienting some equations using Rule **FUZZY EQUATION ORIENTATION** of Figure 2.11 is that Rule **FUZZY NON-ALIGNED-ARGUMENT TERM DECOMPOSITION** may then easily choose a term's functor's similarity class representative as the one of least arity; *viz.*, the one on the left-hand side. However, these assumptions do not hold in general in our new situation. Indeed, this is possible only if each functor similarity class at approximation degree α happens to have a least-arity functor term representative with *total* argument-position maps to all other members of the similarity class at this approximation degree.

Example 2.16 Composing partial non-aligned argument-position map for similar functors — Let us elaborate on Example 2.15: in addition to the similar functors "$\mathtt{foo}/5$" and "$\mathtt{bar}/4$" at a given approximation degree $\alpha \in (0,1]$ with *partial* argument maps $\mu^\alpha_{\mathtt{foo},\mathtt{bar}} : \{3,4\} \to \{2,4\} = \{3 \mapsto 4, 4 \mapsto 2\}$ and $\mu^\alpha_{\mathtt{bar},\mathtt{foo}} : \{2,4\} \mapsto \{3,4\} = \{2 \mapsto 4, 4 \mapsto 3\}$ so that $\mu^\alpha_{\mathtt{foo},\mathtt{bar}} = (\mu^\alpha_{,\mathtt{bar},\mathtt{foo}})^{-1}$, and $\mu^\alpha_{\mathtt{bar},\mathtt{foo}} = (\mu^\alpha_{\mathtt{foo},\mathtt{bar}})^{-1}$), there also exists a functor "$\mathtt{fuz}/2$" that is similar at this approximation degree α to both $\mathtt{foo}/5$ and $\mathtt{bar}/4$ with *total* argument maps $\mu^\alpha_{\mathtt{fuz},\mathtt{foo}} : \{1,2\} \to \{1,2,3,4,5\} = \{1 \mapsto 3, 2 \mapsto 4\}$ and $\mu^\alpha_{\mathtt{fuz},\mathtt{bar}} : \{1,2\} \to \{1,2,3,4\} = \{1 \mapsto 4, 2 \mapsto 2\}$, in such a way that $\mu^\alpha_{\mathtt{fuz},\mathtt{foo}} = \mu^\alpha_{\mathtt{bar},\mathtt{foo}} \circ \mu^\alpha_{\mathtt{fuz},\mathtt{bar}}$ and $\mu^\alpha_{\mathtt{fuz},\mathtt{bar}} = \mu^\alpha_{\mathtt{foo},\mathtt{bar}} \circ \mu^\alpha_{\mathtt{fuz},\mathtt{foo}}$. This, of course, requires that $\mathbf{ran}(\mu^\alpha_{\mathtt{fuz},\mathtt{foo}}) = \mathbf{dom}(\mu^\alpha_{\mathtt{foo},\mathtt{bar}})$ and also that $\mathbf{ran}(\mu^\alpha_{\mathtt{fuz},\mathtt{bar}}) = \mathbf{dom}(\mu^\alpha_{\mathtt{bar},\mathtt{foo}})$, as well as $\mathbf{ran}(\mu^\alpha_{\mathtt{fuz},\mathtt{foo}}) = \mathbf{ran}(\mu^\alpha_{\mathtt{bar},\mathtt{foo}})$ and $\mathbf{ran}(\mu^\alpha_{\mathtt{fuz},\mathtt{bar}}) = \mathbf{ran}(\mu^\alpha_{\mathtt{foo},\mathtt{bar}})$, as in the term similarity Expressions (2.32).



(2.32)

We show next how this can be accommodated in our formalization. In the following, the set denoted as $\{1,\ldots,n\}$ with $n = 0$ is always equal to the empty set $\emptyset$. In other words, for any $n \in \mathbb{N}$, $\{1,\ldots,n\} = \emptyset$ if and only if $n = 0$.

DEFINITION 2.13 (PARTIAL-MAP NON-ALIGNED SIMILAR FUNCTORS) *Let $m \geq 0$, $n \geq 0$; two functors $f \in \Sigma_m$ and $g \in \Sigma_n$ are said to be partial-map non-aligned similar functors at approximation degree $\alpha \in [0,1]$ whenever:*

1. *there is a set $\mathcal{D}^{\alpha}_{fg} \subseteq \{1,\ldots,m\}$ of argument positions of f and a set $\mathcal{D}^{\alpha}_{gf} \subseteq \{1,\ldots,n\}$ of argument positions of g such that $|\mathcal{D}^{\alpha}_{fg}| = |\mathcal{D}^{\alpha}_{gf}|$; and,*

2. *there exist a pair of mutually inverse bijections $\mu^{\alpha}_{fg} : \mathcal{D}^{\alpha}_{fg} \to \{1,\ldots,n\}$ and $\mu^{\alpha}_{gf} : \mathcal{D}^{\alpha}_{gf} \to \{1,\ldots,m\}$ such that $\mathbf{ran}(\mu^{\alpha}_{fg}) = \mathcal{D}^{\alpha}_{gf} = \mathbf{dom}(\mu^{\alpha}_{gf})$ and $\mathbf{ran}(\mu^{\alpha}_{gf}) = \mathcal{D}^{\alpha}_{fg} = \mathbf{dom}(\mu^{\alpha}_{fg})$.*

Note that it is possible in the above definition that $\mathbf{dom}(\mu^{\alpha}_{fg}) = \emptyset$ or $\mathbf{ran}(\mu^{\alpha}_{fg}) = \emptyset$. The former means that no argument of f need be similar to any argument of g, and the latter means that no argument of g need be similar to any argument of f. That is, in both cases, the similarity of terms they construct reduces to that of the functors, regardless of any subterms.

We shall always require, for any approximation degree $\alpha \in [0,1]$ and any functor f, that $\mathcal{D}^{\alpha}_{ff} = \{1, \ldots, \mathbf{arity}(f)\}$, $|\mathcal{D}^{\alpha}_{ff}| = \mathbf{arity}(f)$, and $\mu^{\alpha}_{ff} = \mathbb{1}_{\{1,\ldots,\mathbf{arity}(f)\}}$. This means that pairs of the form $\langle f, f\rangle$ (*i.e.*, the diagonal) always have as argument-position map the *total identity* on *all* the argument positions of f at any approximation degree.

The case where at least one of any two similar functors has a total injective map of its argument positions into the other functor's is a special case of this. When this is so, argument-position maps are composable because *all* the positions in the range of a map are always in the domain of any map from this functor to another (of greater arity). With partial maps however, this may no longer be possible.

Example 2.17 Non-composable inconsistent partial-map non-aligned functors — In addition to the functors $\mathtt{foo}/5$ and $\mathtt{bar}/4$ of Example 2.16 where $\mu^{\alpha}_{\mathtt{foo},\mathtt{bar}} = \{\,3 \mapsto 4,\ 4 \mapsto 2\,\}$ (and $\mu^{\alpha}_{\mathtt{bar},\mathtt{foo}} = \{\,2 \mapsto 4,\ 4 \mapsto 3\,\}$), consider the functor $\mathtt{biz}/4$ and the map $\mu^{\alpha}_{\mathtt{bar},\mathtt{biz}} : \{1 \mapsto, 2,\ 3 \mapsto 4\}$. These maps will not be composable simply because $\mathbf{ran}(\mu^{\alpha}_{\mathtt{foo},\mathtt{bar}}) = \{2,4\}$ and $\mathbf{dom}(\mu^{\alpha}_{\mathtt{bar},\mathtt{biz}}) = \{1,3\}$ have no elements in common. That is, $\mathbf{ran}(\mu^{\alpha}_{\mathtt{foo},\mathtt{bar}}) \cap \mathbf{dom}(\mu^{\alpha}_{\mathtt{bar},\mathtt{biz}}) = \emptyset$.

And even if they had compatible domain and range, say if $\mu^{\alpha}_{\mathtt{bar},\mathtt{biz}} : \{\,1 \mapsto 2,\ 2 \mapsto 4\,\}$, but $\mu^{\alpha}_{\mathtt{foo},\mathtt{biz}} : \{\,3 \mapsto 3,\ 4 \mapsto 4\,\}$, this would mean that the composition $\mu^{\alpha}_{\mathtt{bar},\mathtt{biz}} \circ \mu^{\alpha}_{\mathtt{foo},\mathtt{bar}}$ and the map $\mu^{\alpha}_{\mathtt{foo},\mathtt{biz}}$ also disagree as we have, on one hand:

$$\mu^{\alpha}_{\mathtt{foo},\mathtt{biz}} = \{3 \mapsto 3, 4 \mapsto 4\}$$

and on the other hand:

$$\begin{aligned}\mu^{\alpha}_{\mathtt{bar},\mathtt{biz}} \circ \mu^{\alpha}_{\mathtt{foo},\mathtt{bar}} &= \{\,3 \mapsto \mu^{\alpha}_{\mathtt{bar},\mathtt{biz}}(\mu^{\alpha}_{\mathtt{foo},\mathtt{bar}}(3)), 4 \mapsto \mu^{\alpha}_{\mathtt{bar},\mathtt{biz}}(\mu^{\alpha}_{\mathtt{foo},\mathtt{bar}}(4))\,\}\\ &= \{\,3 \mapsto \mu^{\alpha}_{\mathtt{bar},\mathtt{biz}}(4), 4 \mapsto \mu^{\alpha}_{\mathtt{bar},\mathtt{biz}}(2)\,\}\\ &= \{\,3 \mapsto ?, 4 \mapsto 4\,\}\end{aligned}$$

which is compositionally inconsistent, and thus $\mu^{\alpha}_{\mathtt{bar},\mathtt{biz}} \circ \mu^{\alpha}_{\mathtt{foo},\mathtt{bar}} \neq \mu^{\alpha}_{\mathtt{foo},\mathtt{biz}}$.

And this is inconsistent also in the other direction as well, since $\mu^{\alpha}_{\mathtt{bar},\mathtt{foo}} = \{\,2 \mapsto 4,\ 4 \mapsto 3\,\}$, $\mu^{\alpha}_{\mathtt{bar},\mathtt{biz}} : \{\,1 \mapsto 2,\ 2 \mapsto 4\,\}$, and $\mu^{\alpha}_{\mathtt{foo},\mathtt{biz}} : \{\,3 \mapsto 3,\ 4 \mapsto 4\,\}$, entail that the composition $\mu^{\alpha}_{\mathtt{foo},\mathtt{biz}} \circ \mu^{\alpha}_{\mathtt{bar},\mathtt{foo}}$ and the map $\mu^{\alpha}_{\mathtt{bar},\mathtt{biz}}$ would also disagree. We have, on one hand:

$$\mu^{\alpha}_{\mathtt{bar},\mathtt{biz}} = \{1 \mapsto 2, 2 \mapsto 4\}$$

and on the other hand:

$$\begin{aligned}\mu^{\alpha}_{\mathtt{foo,biz}} \circ \mu^{\alpha}_{\mathtt{bar,foo}} &= \{\, 2 \mapsto \mu^{\alpha}_{\mathtt{foo,biz}}(\mu^{\alpha}_{\mathtt{bar,foo}}(2)), 4 \mapsto \mu^{\alpha}_{\mathtt{foo,biz}}(\mu^{\alpha}_{\mathtt{bar,foo}}(4)) \,\} \\ &= \{\, 2 \mapsto \mu^{\alpha}_{\mathtt{foo,biz}}(4), 4 \mapsto \mu^{\alpha}_{\mathtt{foo,biz}}(3) \,\} \\ &= \{\, 1 \mapsto ?, 2 \mapsto 4, 4 \mapsto 3 \,\}\end{aligned}$$

which, again, entails $\mu^{\alpha}_{\mathtt{foo,biz}} \circ \mu^{\alpha}_{\mathtt{bar,foo}} \neq \mu^{\alpha}_{\mathtt{bar,biz}}$, and thus is compositionally inconsistent as well.

We next define formally the conditions for similar functor partial-alignment consistency of a signature to make argument-position maps always be composable at any given approximation degree.

DEFINITION 2.14 (CONSISTENT PARTIAL SIMILARITY OF NON-ALIGNED SIGNATURE) *Let $\Sigma \stackrel{def}{=} \cup_{k\geq 0}\Sigma_k$ be a functor signature, and let $\approxeq : \Sigma \times \Sigma \to [0,1]$ be a similarity on Σ. It is said that signature Σ is non-aligned admitting $\approxeq$ as a consistent partial similarity whenever all the following statements hold:*

1. *all argument-position mappings conditions (2.15)–(2.18) are verified;*

2. *all pairs $\langle f, g\rangle \in \Sigma_m \times \Sigma_n$ of similar functors—i.e., when $f \approxeq_\alpha g$ with $\alpha \in [0,1]$—are partial-map non-aligned similar functors at approximation degree α as specified by Definition 2.13;*

3. *for any functors $f \in \Sigma$ and $g \in \Sigma$, and approximation degrees $\alpha \in [0,1]$ and $\beta \in [0,1]$:*

$$\alpha \leq \beta \implies \mathcal{D}^{\alpha}_{fg} \subseteq \mathcal{D}^{\beta}_{fg}\,; \tag{2.33}$$

4. *for all $f \in \Sigma_m$, $g \in \Sigma_n$, $h \in \Sigma_\ell$, $m \geq 0$, $n \geq 0$, and $\ell \geq 0$:*

$$\begin{cases} \mathbf{ran}(\mu^{\alpha}_{fg}) = \mathbf{dom}(\mu^{\alpha}_{gh}) \;\; (= \mathcal{D}^{\alpha}_{gh}) \\ \mathbf{ran}(\mu^{\alpha}_{fh}) = \mathbf{ran}(\mu^{\alpha}_{gh})\,; \end{cases} \tag{2.34}$$

and:

$$\begin{cases} \mu^{\alpha}_{hf} = \mu^{\alpha}_{gf} \circ \mu^{\alpha}_{hg} \\ \mu^{\alpha}_{hg} = \mu^{\alpha}_{fg} \circ \mu^{\alpha}_{hf}\,. \end{cases} \tag{2.35}$$

These conditions are concisely summarized as the commutative functional diagram of Figure 2.15.

COROLLARY 2.5 (COMPOSABILITY OF ARGUMENT-POSITION MAPS) *A non-aligned signature Σ with a consistent partial similarity $\approxeq$ verifying all the conditions of Definition 2.14 are always consistently composable at any given approximation degree $\alpha \in [0,1]$.*

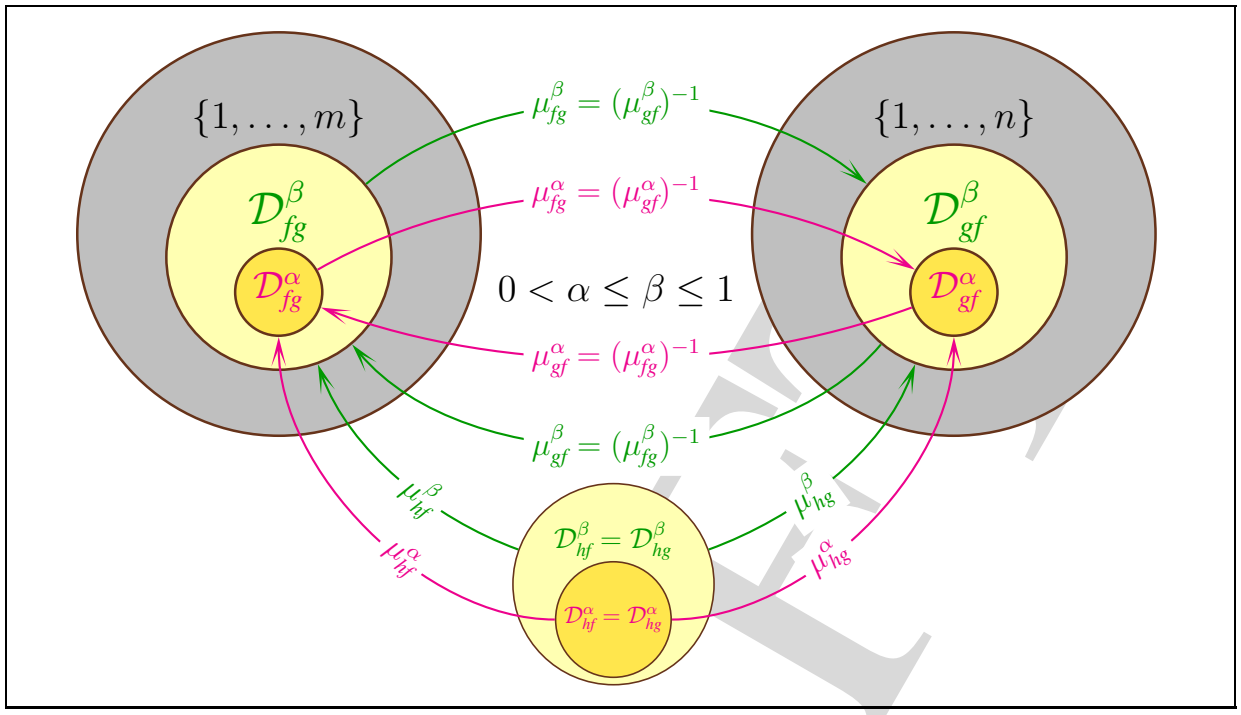


Figure 2.15: Partial-map non-aligned similar functors argument-map consistency diagram

PROOF We need to verify that, for any functors f, g, h and any $\alpha \in [0,1]$:

$$\mathbf{ran}(\mu^{\alpha}_{fg}) \subseteq \mathbf{dom}(\mu^{\alpha}_{gh})$$

and,

$$\mathbf{dom}(\mu^{\alpha}_{fh}) = \mathbf{dom}(\mu^{\alpha}_{fg}) \text{ and } \mathbf{ran}(\mu^{\alpha}_{fh}) = \mathbf{ran}(\mu^{\alpha}_{gh})$$

and that, for all positions $i \in \mathbf{dom}(\mu^{\alpha}_{fg})$:

$$\mu^{\alpha}_{fh}(i) = \mu^{\alpha}_{gh}(\mu^{\alpha}_{fg}(i)).$$

All three properties can be verified to be immediate consequences of the conditions of Definition 2.14. □

The following is also a corollary of Definition 2.14 and Definition 2.13.

COROLLARY 2.6 (STABILITY OF PARTIAL SIMILARITY DOMAINS AND CLASSES) *Given any two functors f and g such that $f \approxeq_{\alpha} g$ at approximation degree $\alpha \in [0,1]$, the size $|\mathcal{D}^{\alpha}_{fg}|$ of the set $\mathcal{D}^{\alpha}_{fg}$ is constant for fixed α; that is, $|[f]^{\alpha}_{\approxeq}| = |\mathcal{D}^{\alpha}_{fg}| = |\mathcal{D}^{\alpha}_{gf}| = |[g]^{\alpha}_{\approxeq}|$.*

PROOF This follows since, at any fixed approximation degree, all maps between functors in a similarity class are bijective and the fact that they all verify the properties specified in Definition 2.14 and Definition 2.13. □

PARTIAL NON-ALIGNED SIMILAR TERM DECOMPOSITION

$$\frac{(\, E \cup \{\, f(s_1,\ldots,s_m) \doteq g(t_1,\ldots,t_n) \,\}\,)_\alpha}{(\, E \cup \{\, s_{d_1} \doteq t_{\mu_{fg}^{\alpha\wedge\beta}(d_1)}, \ldots, s_{d_p} \doteq t_{\mu_{fg}^{\alpha\wedge\beta}(d_m)} \,\}\,)_{\alpha\wedge\beta}}$$

$$\left[f \overset{\mu_{fg}^{\beta}}{\approx\!\!\!\approx}_{\beta} g;\ \ 0 \leq |\mathcal{D}_{fg}^{\alpha\wedge\beta}| = p,\ p \leq \min(m,n);\ \ \mathcal{D}_{fg}^{\alpha\wedge\beta} = \{d_1,\ldots d_p\} \right]$$

Figure 2.16: Partial non-aligned similar $\mathcal{FOT}$ similar term decomposition rule

DEFINITION 2.15 *The fuzzy relation $\approx\!\!\!\approx^{\mathcal{T}}$ on $\mathcal{T}_{\Sigma,\mathcal{V}}$ is defined inductively as:*

1. $\forall\, X \in \mathcal{V},\ \ X \approx\!\!\!\approx^{\mathcal{T}}_1 X$;
2. $\forall\, X \in \mathcal{V}, \forall\, t \in \mathcal{T}$ *such that* $X \neq t$, $X \sim^{\mathcal{T}}_0 t$ *and* $t \sim^{\mathcal{T}}_0 X$;
3. *for* $m \in \mathbb{N}$, $n \in \mathbb{N}$, $f \approx\!\!\!\approx_\alpha g$ *with* $\mu^\alpha_{fg} : \mathcal{D}^\alpha_{fg} \to \mathcal{D}^\alpha_{gf}$ *and* $s_i \approx\!\!\!\approx^{\mathcal{T}}_{\alpha_i} t_{\mu^\alpha_{fg}(i)}$ *for all* $i \in \mathcal{D}^\alpha_{fg}$, *then:*

$$f(s_1,\ldots,s_m) \approx\!\!\!\approx^{\mathcal{T}}_{(\alpha \wedge \bigwedge_{i\in\mathcal{D}^\alpha_{fg}} \alpha_i)} g(t_1,\ldots,t_n). \tag{2.36}$$

With this definition and the following theorem, we shall now have all the necessary formal tools to proceed as we did for the two previous $\mathcal{FOT}$ lattice structure constructions in the case where non-aligned Σ admits $\approx\!\!\!\approx$ as a consistent partial similarity.

THEOREM 2.7 *The relation $\approx\!\!\!\approx^{\mathcal{T}}$ defined by Definition 2.15 is a similarity relation on $\mathcal{T}$ the set of $\mathcal{FOT}$s.*

From this, in the same manner as we did before, we shall derive a weaker subsumption preorder on $\mathcal{FOT}$s as well as adapt our previous sets of rules to specify the corresponding unification and generalization lattice operations for this preorder. This is what we present next.

The rules for unification of similar partial-map non-aligned $\mathcal{FOT}$s are those of Maria Sessa's weak unification (see Figure 2.7) where Rule **WEAK TERM DECOMPOSITION** is replaced with Rule **PARTIAL NON-ALIGNED TERM DECOMPOSITION** given in Figure 2.16. ***N.B.***: there is no need to re-orient a term equation as for total maps (see Figure 2.11). **Why?**

The rules for generalization of partial-map non-aligned similar $\mathcal{FOT}$s are those given in Figure 2.13 where Rule **SIMILAR FUNCTORS** is replaced with Rule **PARTIAL NON-ALIGNED FUNCTOR SIMILARITY** given in Figure 2.17, where, for $i = 1,\ldots,p$:

$$\begin{pmatrix} s'_i \\ t'_i \end{pmatrix}_{\beta_i} \overset{\text{def}}{=} \begin{pmatrix} s_{\mu_{hf}^{\alpha_{i-1}}(i)} \\ t_{\mu_{hg}^{\alpha_{i-1}}(i)} \end{pmatrix} \uparrow_{\alpha_{i-1}} \begin{pmatrix} \sigma_1^{i-1} \\ \sigma_2^{i-1} \end{pmatrix} \qquad \text{and} \qquad \begin{pmatrix} \sigma_1^{i-1} \\ \sigma_2^{i-1} \end{pmatrix}_{\beta_i} \vdash \begin{pmatrix} s'_i \\ t'_i \end{pmatrix} u_i \begin{pmatrix} \sigma_1^{i} \\ \sigma_2^{i} \end{pmatrix}_{\alpha_i}.$$

N.B.: there is no differentiating left/right rules as for total maps (see Figure 2.14); only a single rule is needed. **Why?**

PARTIAL NON-ALIGNED FUNCTOR SIMILARITY

$$\left[f/m \approx_{\beta} g/n;\ \ \alpha_0 \stackrel{\text{def}}{=} \alpha \wedge \beta;\ \ h/p \in [f/m, g/n]_{\alpha_0};\ \ |\mathcal{D}^{\alpha_0}_{hf}| = |\mathcal{D}^{\alpha_0}_{hg}| = p \right]$$

$$\frac{\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha_0} \vdash \begin{pmatrix}s'_1\\ t'_1\end{pmatrix} u_1 \begin{pmatrix}\sigma^1_1\\ \sigma^1_2\end{pmatrix}_{\alpha_1} \ldots \begin{pmatrix}\sigma^{p-1}_1\\ \sigma^{p-1}_2\end{pmatrix}_{\alpha_{p-1}} \vdash \begin{pmatrix}s'_p\\ t'_p\end{pmatrix} u_p \begin{pmatrix}\sigma^p_1\\ \sigma^p_2\end{pmatrix}_{\alpha_p}}{\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix}_{\alpha} \vdash \begin{pmatrix}f(s_1,\ldots,s_m)\\ g(t_1,\ldots,t_n)\end{pmatrix} h(u_1,\ldots,u_p) \begin{pmatrix}\sigma^p_1\\ \sigma^p_2\end{pmatrix}_{\alpha_p}}$$

Figure 2.17: Partial non-aligned similar $\mathcal{FOT}$ generalization rule

Automated signature completion Note that the unification and generalization rules above will work with non-aligned similar functors with partial argument-position maps as long as the conditions on the signature, the functor similarity, as well as all the corresponding partial argument alignment, verify the signature consistency conditions given in Definition 2.14 and illustrated in Figure 2.15. These conditions are necessary to ensure *composability* of all partial argument maps at any approximation level $\alpha \in [0, 1]$. Indeed, it is composability of similar functor argument maps that ensures consistent transitivity of the functor similarity.

However, one may object to requiring signatures and their similarities to possess complete consistent partial maps as rather demanding. We now see how the process of verifying this to be true for a given signature and similarity can be automated. In fact, there are two questions that must be addressed:

1. Can a given signature with specified similarity and partial alignment maps *automatically* be either verified to meet these requirements, or proven inconsistent?
2. Can a consistent but incomplete signature be completed to a minimal consistent one containing it while respecting all its similaririties and argument alignment maps?

The good news is that the answer to both questions is "*yes*."

1. There is a finite procedure to verify that for all pairs of transitive pairs of functors $\langle f, g\rangle$ and $\langle g, h\rangle$, all specified argument maps abide by necessary consistency conditions or to point out where this is violated, and why.
2. Should a signature be detected to be incomplete at a given approximation level $\alpha \in [0, 1]$ in the sense that some functor similarity class does not possess a least-arity representative with complete and consistent argument maps to all other functors in its class at level α, then either the signature can be completed with a new functor with this property with appropriate least (in terms of inclusion of sets of pairs) argument maps to all functors in its class, or an explicit counter-example can be given that shows why there is no consistent signature can complete this signature while respecting all argument maps.

This procedure may be performed at all levels $\alpha \in$ **DEGREES**$^{\approx}$. Pseudocode specifying this completion procedure is given in Figure 2.18. It automatically completes an incomplete signature and a specified base set of similar functor pairs, together with some partial argument-position

maps using a completion procedure on the signature and the similarity so that the signature may either be proven inconsistent, or completed with a similarity such that each similarity class at any approximation degree $\alpha \in \textbf{DEGREES}^{\approx\!\!\approx}$ contains at least one representative functor with consistently aligned *total* argument-position maps to all members of the class.

forall $\alpha \in \textbf{DEGREES}^{\approx\!\!\approx}$ and similarity class $c \in \Pi_\alpha^{\approx\!\!\approx}$ **do**

if $\nexists$ a least-arity similarity class representative in c with *total* argument-position maps to *all* members of c

then

- add a *new* functor h/m to signature Σ_m such that:

$$m = \min\{|\mathcal{D}_{fg}^\alpha| \mid f \in c,\ g \in c\},$$

with least *total* consistent injective maps:

$$\mu_{hf}^\alpha : \{1, \ldots, m\} \to \{1, \ldots, n\}$$

for *all* $f/n \in c$; if not possible, Σ is inconsistent;

- add functor h/m to similarity class c;

Figure 2.18: Automated completion of partial-maps for non-aligned signature similarity

Example 2.18 Non-aligned signature partial similarity completion — Consider a signature Σ in which the only pair of non-identical similar functors at a given similarity degree α are $f/4$ and $g/3$ such that $f \approx_\alpha^{\mu_{fg}^\alpha} g$ and $g \approx_\alpha^{\mu_{gf}^\alpha} f$ with mutually inverse injective partial argument-position maps $\mu_{fg}^\alpha = \{\langle 2,1\rangle, \langle 4,3\rangle\}$ and $\mu_{gf}^\alpha = \{\langle 1,2\rangle, \langle 3,4\rangle\}$, so that $\mathcal{D}_{fg}^\alpha \stackrel{\text{def}}{=} \{2,4\}$ and $\mathcal{D}_{fg}^\alpha \stackrel{\text{def}}{=} \{1,3\}$.

The similarity class $c = \{f, g\}$ does not have a least-arity functor class representative with total argument-position maps to all members of c. So, since $\{|\mathcal{D}_{fg}^\alpha| \mid f \in c,\ g \in c\} = 2$, the minimum value in this set is 2. So we add a new functor $h/2$ to Σ_2 with the *total* argument maps: $\mu_{hf}^\alpha = \{\langle 1, \mu_{gf}^\alpha(1)\rangle, \langle 2, \mu_{gf}^\alpha(3)\rangle\}$ and $\mu_{hg}^\alpha = \{\langle 1, \mu_{fg}^\alpha(2)\rangle, \langle 2, \mu_{fg}^\alpha(4)\rangle\}$; that is, $\mu_{hf}^\alpha = \{\langle 1,2\rangle, \langle 2,4\rangle\}$ and $\mu_{hg}^\alpha = \{\langle 1,1\rangle, \langle 2,3\rangle\}$.

And this is consistent by construction since $\mu_{hg}^\alpha = \mu_{fg}^\alpha \circ \mu_{hf}^\alpha$ and $\mu_{hf}^\alpha = \mu_{gf}^\alpha \circ \mu_{hg}^\alpha$ (as can be easily verified). So $h/2$ can be added to class c.

2.7 Recapitulation

In this chapter, we have developed a formal derivation of fuzzy lattice operations for the data structure known as first-order term. This is achieved by means of syntax-driven constraint normalization rules for both unification and generalization. These operations are then extended to enable arbitrary mismatch between similar terms whether functor-based, arity-based (number

and order), or combinations, and then yet to consistent *partial* mappings of argument positions between similar functors.

This last lattice of $\mathcal{FOT}$s permits Fuzzy Logic Programming over arbitrary misaligned-data bases, or more generally Approximate Information Retrieval (using fuzzy unification) and Approximate Knowledge Acquisition (using fuzzy generalization) over heterogeneous but similar data models, whereby approximating modulo constructor similarity is consistently conjugated with approximating with less structure details.

As for implementation, the prospects are many and discussed in Chapter 4.[25] The most immediate concerns implementation of such operations in the form of public libraries to complement extant tools for first-order terms and substitutions [73].

As for what perspectives this work may open, there are several avenues to explore. There are several other disciplines where this technology has potential for fuzzifying applications wherever $\mathcal{FOT}$s are used for their lattice-theoretic properties such as linguistics and learning. Finally, most promising is using this work's approach to more generic and more expressive knowledge structures for applications such as Information Retrieval (*e.g.*, in the line of [38]), or Data and Knowledge Base Management, Ontology Alignment, *etc.*, ...

2.8 Relation to Other Works

There have been other works dealing with the larger issue of integrating general equational theories into logical reasoning, not just the specific theory of fuzzy equivalence among terms. Among the most formally and operationally complete approach, pioneered by Goguen *et al.* in the seventies, is the set of works based on *initial algebras* [64].[26] Recall that an operator algebra is initial iff there exists a homomorphism from it to all other algebras that are semantic models of first-order terms freely built with these operators and variables [62]. Namely, initiality is the property that guarantees that the formal meaning of syntactic terms defined for $\mathcal{FOT}$s modulo congruence classes defined by equations is preserved for all interpretations. This result was later extended from equational systems to implicational systems by Mahr and Makowsky [89]. In this latter paper, it was also shown that Horn Logic (*i.e.*, an implicational system consitituting the formal basis of the Prolog language), is the largest class of logic that admits an initial algebra semantics.

While the initial algebra approach has shown its general applicability for term unification and generalization, including more recently with the work of Alpuente *et al.* over order-sorted signatures [24] and with equational theories [25], its specific application to fuzzy congruences has yet to be done. While it could conceivably be specified by instantiating the general scheme of initial semantics, this must be at the expense of both formal and operational simplicity when compared with our approach which can be expressed as a direct extension of conventional operations of unification and generalization of first-order terms. This is because it is specifically adapted to a fuzzy equivalence of terms rather than general-purpose reasoning modulo equational theories. In the latter more general approach, equations can be used as terminating rewrite rules and unifica-

[25] Section 4.4.

[26] An initial algebra is also called *free* algebra, or *syntactic* algebra, or *tree* algebra, or *term* algebra—because its elements are the syntactic term structures one can define recursively by nesting terms as arguments of other terms (*i.e.*, the model taking as interpretation homomorphism the identity function on terms).

tion modulo this theory is made operational using the general equation-solving technique known as "*narrowing*" [58], [57]. However, for this to work, one must have a terminating and confluent set of rewrite rules. Such may be derived in some cases based on undecidable procedures such as Knuth-Bendix completion [82], or unterminating term rewriting [55]. Rather, we limit ourselves to the obviously decidable theory generated homomorphically on $\mathcal{FOT}$s from a fuzzy equivalence relation (a similarity) on a finite signature. This follows the intuition behind Maria Sessa's formal work on fuzzy $\mathcal{FOT}$ unification as well [117]. Also, we support arity and argument position mismatch for similar operators.[27] Be that as it may, one could envisage studying how E-unification for common equational theories such as associativity or commutativity could be extended to unification of terms with similar functors. However, this is another issue and we do not do so in this document's setting.

There have been also works in logic-based databases such as using a similarity degree while comparing syntactically unequal terms; for example, Francesca Arcelli *et al.*'s LIKELOG database logic programming language [30], [28]. These works, however, concern only *ground* terms (*i.e.*, with no variables), not first-order terms (*i.e.*, possibly having variables). Arcelli *et al.*'s notion of similarity distance between terms was later extended from ground terms to first-order terms by Shroeder and Gilbert in the fuzzy logic programming language FURY [61], [115], [116]. They use the same concept as Arcelli *et al.*'s fuzzy equivalence but derived from dynamically evaluating so-called "*edit distance*" between strings on ground terms as well as first-order terms.[28] Thus, their objective is to derive dynamically an estimate of an "edit distance" between terms. The same comments also apply to work by Kutsia *et al.* [84], [85], where the objective is to check all the possibilities of dynamically matching $\mathcal{FOT}$s with *equal* function symbols having unspecified number of arguments (*i.e.*, the same sort of search objective pursued in FURY, where this is done for *unequal* symbols as well). This objective is not ours in that we are not trying to infer dynamically distances between terms. Rather, we *assume given* a matrix of similarity degrees for such a static similarity relation on term constructors and use this information exactly in the same manner as done by Maria Sessa in [117]. In Sessa's context (and ours), this information is *given statically*, not inferred dynamically. Finally, the main advantage of working from a given static matrix of similarity degrees rather than estimating syntactic edit distances (whether dynamically or statically) is that similarity may be semantic among syntactically unrelated but semantically close strings. The context of dynamic syntactic distance estimation is typically for applications of purely lexical variants such as estimating gene similarity in biology [61]. Ours concerns deriving approximate solutions to fuzzy equations given similarity among term constructors.

Our last reference to other work related to unification and generalization of graph data and type structures, is the set of work due to Aït-Kaci *et al.* (*i.e.*, [18] and [21]). In this context, nodes denote sorts (that are organized in a lattice) and arrows denote features (functional attributes between sort nodes); variables denote equations among (possibly cyclic) feature paths. Just as such subsumption and its lattice operations on $\mathcal{FOT}$s can be fuzzified, so can indeed the lattice of Order-Sorted Feature terms. We address this topic in Chapter 3.

[27] See Section 2.6.1.

[28] See [114] this email discussion on the issue.

Chapter 3

Version of January 8, 2019

Order-Sorted Feature Terms

In the previous chapter, it may have come many times to the reader's mind that a $\mathcal{FOT}$ is just syntax for a *tree* where node labels are functors except possibly for some leaf nodes that are replaced with variables, and functor-labeled nodes have position-numbered arrows pointing to the root of the subterm tree at each argument position. It is so indeed. In fact, when the leaf nodes that have the same variable are joined, it is a rooted directed acyclic graph (or "*dag*").[1] As such, it is a special case of a more general kind of labeled rooted (possibly cyclic) graph—called a rooted *order-sorted feature* ($\mathcal{OSF}$) graph. Sort symbols are node labels. Sorts are partially ordered to reflect subsumption and form a lattice. Feature symbols labeling arrows represent functional attributes. These graphs are so ordered by endomorphic structure-preserving (sort, feature, and feature path equations) subsumption. Lattice-theoretic operations for these more general rooted graphs, labeled with partially-orderd sorts and with features, can be shown to extend those on more restricted kind of rooted graphs such as $\mathcal{FOT}$s, labeled with functors, positions, and variables.[2]

In this chapter, we elaborate on these notions. In Section 3.1, we start with an informal introduction to the $\mathcal{OSF}$ formalism (Section 3.1.1), then continue with a formal presentation (Section 3.1.2), leading to expressing $\mathcal{OSF}$ graph lattice operations declaratively as constraint normalization. In Section 3.1.3, we characterize these operations in terms of solving constraints where solutions are functional mappings between reference tags. These mappings generalize to $\mathcal{OSF}$ graphs the notion of $\mathcal{FOT}$ substitution.[3] In Section 3.2, we turn to fuzzifying $\mathcal{OSF}$-graph subsumption and its lattice operations: in Section 3.2.1 we explicate fuzzy $\mathcal{OSF}$ graph unification, and in Section 3.2.2 fuzzy $\mathcal{OSF}$ graph generalization.

3.1 $\mathcal{OSF}$ Formalism

The $\mathcal{OSF}$ formalism extends to typed attributed data structures the representation, syntax, and operations enjoyed by $\mathcal{FOT}$s as used in Logic Programming. It was developed to express rigor-

[1] A rooted (directed) graph is one with a distinguished node, called its *root*, from which all other nodes can be reached.

[2] See Appendix Section B.3.

[3] For implementation issues, see [9], and [13].

ously the meaning and properties of graph structures used in the experimental language $\mathcal{LIFE}$ to represent data and knowledge as, respectively, elements and types [7], [14].

A formal semantics for $\mathcal{OSF}$ structures can be precisely expressed using each of these three mathematical frameworks—(1) algebraic (as terms), (2) logical (as clauses), and (3) operational (as graphs). In [18], the mathematical equivalence of these three formal semantics was established with explicit cross-interpretations. In this part, we shall rely on this result by (1) defining lattice-theoretic operations on the algebra of $\mathcal{OSF}$ terms, (2) defining corresponding constraint normalization rules for these operations (on the logic of $\mathcal{OSF}$ clauses), and (3) deriving an implicit operational semantics from these declarative rules (on $\mathcal{OSF}$ graph structures).

Next, we start with a brief informal description of the kind of labeled graphs $\mathcal{OSF}$ structures are, and how they may be expressed syntactically in a manner that naturally extends the syntax of algebraic $\mathcal{FOT}$s. This term syntax is further transformed into a clausal language upon which unification can be rendered as constraint normalization. We then delve into more details defining this syntax formally as well as $\mathcal{OSF}$ term subsumption, unification, and generalization.

3.1.1 Informal background

Let us take an example (taken from [9]). Let us assume that we wish to describe a 30-year-old person with an id consisting of a name, itself made of two parts: a first name and a last name, both represented as strings. Let us also say that we wish to indicate that such a person has a spouse that is a person sharing his or her id's last name, and such that this latter person's spouse is the first person in question. We propose to represent this information as the graph in Figure 3.1.

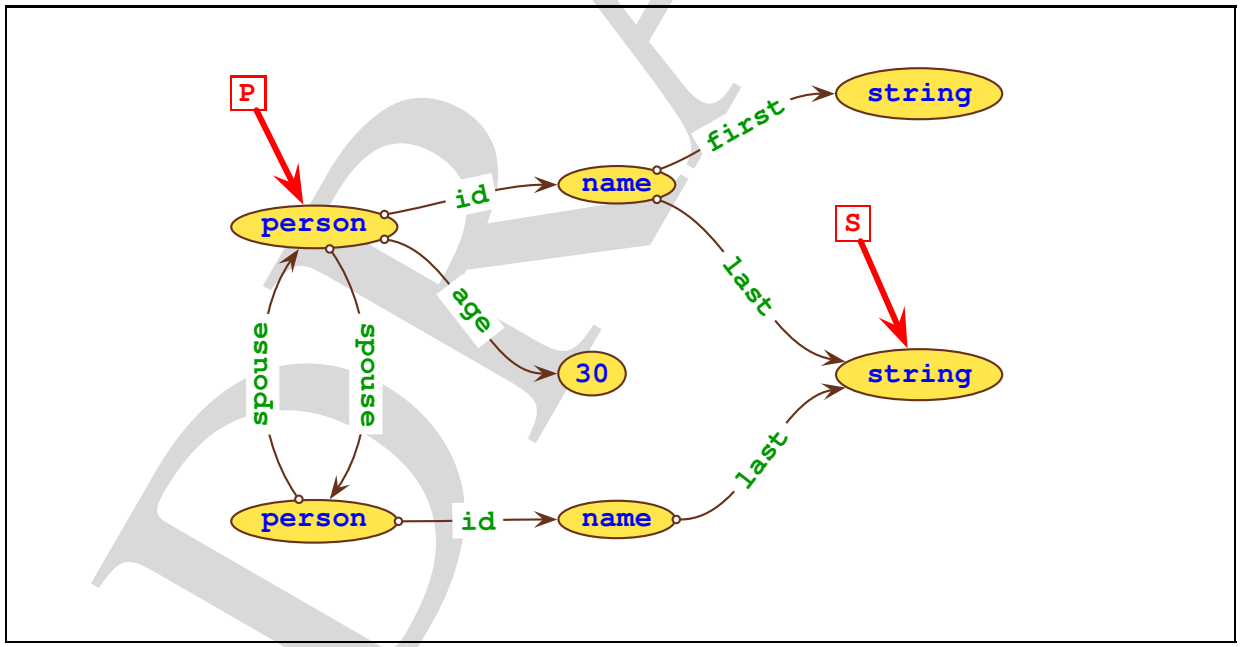


Figure 3.1: Example of $\mathcal{OSF}$ graph

This graph consists of a set of nodes and arcs between some nodes. This graph, however, is a *labeled* graph. There are two kinds of label symbols: one kind is for the nodes (*e.g.*, **person**,

name, *etc.*), and the other is for the arcs. We call the symbols labeling nodes "*sorts*," and the symbols labeling arcs "*features*" (*e.g.*, **spouse**, **age**, *etc.*). Intuitively, sort symbols denote sets, and feature symbols denote functions between these sets. We will identify value-denoting symbols such as **30** as sorts as well since they can be thought of as singleton sets—*i.e.*, in this case the set $\{30\}$ containing the single integer 30. We will also assume that sort symbols are partially ordered with a "*subsort*" relation "***is-a***" denoting set inclusion on the sets denoted by the sorts. This justifies calling such a graph an "*order-sorted feature*" graph; or, $\mathcal{OSF}$ graph for short.

In fact, an $\mathcal{OSF}$ graph can be defined as a rooted sorted *commutative diagram* of function compositions of the same nature as those used in mathematics. In other words, all feature-composition paths between two sort nodes commute.

The other labels such as ***P*** and ***S*** in the graph of Figure 3.1 are used as reference pointers to designate specific nodes—in this case the root node (*e.g.*, **P**)—or indicate an equational constraint among feature paths (*e.g.*, **P** and **S**). we shall call them *reference tags*. Formally, these will be assimilated to logical variables, *i.e.*, lexical references that could be consistently renamed in their context, without altering the meaning of the $\mathcal{OSF}$ graphs, terms, and constraints, in which they appear.

The above informal description of an $\mathcal{OSF}$ graph should make intuitive sense to anyone familiar with the kind of data structure used in object-oriented programming to represent typed object records. This is indeed a good way to think about it. It is this pragmatic understanding that we wish the reader to keep in mind. Our intention, however, is to go beyond merely *representing* and *using* structured types and objects: we not only want these to be convenient for computing, we also want them to be convenient for *reasoning*—while never losing this simple intuition. Thus, we follow a simple syntax to represent these graphs that will enable us to do so. For example, we represent the graph shown in Figure 3.1 using the syntax shown in Figure 3.2.

```
P :person(id → name(first → string,
                    last → S :string),
          age → 30,
          spouse → person(id → name(last → S),
                          spouse → P)).
```

Figure 3.2: $\mathcal{OSF}$ term syntax for the $\mathcal{OSF}$ graph of Figure 3.1

The reader with some knowledge of Logic Programming will have noted that this syntax generalizes that of Prolog terms—*i.e.*, of $\mathcal{FOT}$s. Indeed, a Prolog term can be seen as a restricted kind of $\mathcal{OSF}$ term, where sort symbols are data constructors; feature symbols are (implicit) subterm positions; and, reference tags are so-called "logical variables," and can only occur as leaves.[4] It is to stress this fact that we call an expression such as that in Figure 3.2 an "$\mathcal{OSF}$ *term*." Note that, unlike a Prolog term where subterms are written following an implicit position order, an $\mathcal{OSF}$ term may be written up to permutation of its subterms since explicit feature labels allow specifying subterms in any order while still representing the same $\mathcal{OSF}$ graph. For

[4] See Appendix Section B.3.

example, the $\mathcal{OSF}$ term in Figure 3.3 could as well be used to represent the same $\mathcal{OSF}$ graph in Figure 3.1.

```
P :person(age → 30,
          spouse → person(spouse → P,
                          id → name(last → S :string)),
          id → name(last → S,
                    first → string)).
```

Figure 3.3: Equivalent $\mathcal{OSF}$ term syntax for the $\mathcal{OSF}$ graph of Figure 3.1

3.1.2 Formal background

In this section, we give a brief formal account of the notions illustrated in the foregoing informal description. The reader is referred to [18], [19], and [20] for all technical details such as mathematical proofs, and to [9] and [13] for more operational details such as implementation, as well as all further relevant references in them.

§ $\mathcal{OSF}$ Signature

An $\mathcal{OSF}$ *signature* is a quadruplet $\langle \mathcal{S}, \preceq, \curlywedge, \mathcal{F} \rangle$ where:

- $\mathcal{S}$ is a set of *sorts* containing at least the two distinguished sorts $\top$ and $\bot$;
- $\preceq$ is a decidable partial order on $\mathcal{S}$ for which $\bot$ is unique least element and $\top$ is unique greatest element;
- $\langle \mathcal{S}, \preceq, \curlywedge \rangle$ is a lower semi-lattice ($s \curlywedge s'$ is called the greatest common subsort of s and s');
- $\mathcal{F}$ is a set of *feature symbols*.

Referring to the ψ-term example in Figure 3.2, the set of sorts $\mathcal{S}$ contains set-denoting symbols such as **person**, **name**, and **string**. The set of features $\mathcal{F}$ contains function-denoting symbols (on the left of $\rightarrow$), such as **id**, **age**, **spouse**, **first**, **last**, *etc.*, ... The ordering on the sorts $\mathcal{S}$ denotes set inclusion and the infimum operation $\curlywedge$ denotes set intersection. Therefore, $\top$ denotes the all-inclusive sort (the set of all things), and $\bot$ denotes the all-exclusive sort (the set of no things). This is formalized next.

§ $\mathcal{OSF}$ Algebras

Given an $\mathcal{OSF}$ signature $\langle \mathcal{S}, \preceq, \curlywedge, \mathcal{F} \rangle$, an $\mathcal{OSF}$ *algebra* $\mathcal{A}$ is a mathematical structure consisting of a triplet:

$$\mathcal{A} \stackrel{\text{def}}{=} \langle D^{\mathcal{A}}, (s^{\mathcal{A}})_{s \in \mathcal{S}}, (f^{\mathcal{A}})_{f \in \mathcal{F}} \rangle \tag{3.1}$$

where:

- $D^{\mathcal{A}}$ is a non-empty set, called the *domain* of $\mathcal{A}$;
- for each sort symbol s in $\mathcal{S}$, $s^{\mathcal{A}}$ is a subset of the domain; in particular, $\top^{\mathcal{A}} = D^{\mathcal{A}}$ and $\bot^{\mathcal{A}} = \emptyset$;
- $(s_1 \curlywedge s_2)^{\mathcal{A}} = s_1^{\mathcal{A}} \cap s_2^{\mathcal{A}}$ for two sorts s_1 and s_2 in $\mathcal{S}$;
- for each feature f in $\mathcal{F}$, $f^{\mathcal{A}}$ is a total unary function from the domain into the domain; *i.e.*, $f^{\mathcal{A}} : D^{\mathcal{A}} \mapsto D^{\mathcal{A}}$.[5]

§ $\mathcal{OSF}$ Homomorphisms

In algebra, a homomorphism between two algebraic structures is a structure-preserving mapping. The essence of a structure-preserving mapping between $\mathcal{OSF}$ algebras is that it should preserve sort inclusion and feature application. Thus, an $\mathcal{OSF}$ *homomorphism* $\gamma : \mathcal{A} \mapsto \mathcal{B}$ between two $\mathcal{OSF}$ algebras $\mathcal{A}$ and $\mathcal{B}$ is a function $\gamma : D^{\mathcal{A}} \mapsto D^{\mathcal{B}}$ such that:

- $\gamma(s^{\mathcal{A}}) \subseteq s^{\mathcal{B}}$; and,
- $\gamma(f^{\mathcal{A}}(d)) = f^{\mathcal{B}}(\gamma(d))$ for all $d \in D^{\mathcal{A}}$.

§ $\mathcal{OSF}$ Endomorphisms

The notion of interest for inheritance is that of $\mathcal{OSF}$ *endomorphism*. That is, when an $\mathcal{OSF}$ homomorphism γ is internal to an $\mathcal{OSF}$ algebra (*i.e.*, $\mathcal{A} = \mathcal{B}$), it is called an $\mathcal{OSF}$ endomorphism of $\mathcal{A}$. This means:

- $\forall s \in \mathcal{S},\ \gamma(s^{\mathcal{A}}) \subseteq s^{\mathcal{A}}$; and,
- $\forall f \in \mathcal{F}, \forall d \in D^{\mathcal{A}},\ \gamma(f^{\mathcal{A}}(d)) = f^{\mathcal{A}}(\gamma(d))$.

Such an endomorphism defines a natural pre-order $\preceq$ on the domain $D^{\mathcal{A}}$ of any $\mathcal{OSF}$ algebra $\mathcal{A}$ whereby $t_1 \preceq t_2$ iff $\exists \gamma : D^{\mathcal{A}} \to D^{\mathcal{A}}$ such that $t_1 = \gamma(t_2)$. This is as pictured in Figure 3.4. As can be seen in this figure, this definition captures formally and precisely *inheritance of attributes* as used, *e.g.*, in object-oriented classes, semantic networks, and formal ontological logics defining *concept hierarchies*. Namely, a concept C_1 (the subconcept) inherits from a concept C_2 (its superconcept) *if and only if* there exists an $\mathcal{OSF}$ endormorphism (γ) taking the set denoted by the superconcept C_2 ($= s$) to the set denoted by the subconcept C_1 ($= \gamma(s)$). Being an $\mathcal{OSF}$ endomorphism means therefore that $C_1.f = \gamma(C_2.f) = \gamma(f(s)) = f(\gamma(s))$. In other words, feature application and sort refinement commute—doing the former then the latter or *vice versa* yields the same result.

§ Syntax of $\mathcal{OSF}$ Terms

Definition 3.1 ($\mathcal{OSF}$ Term Syntax) *An $\mathcal{OSF}$ term t is an expression taking one of the three possible syntactic forms:*

1. *an unsorted variable X; or,*

[5] See Appendix Section B.2 for partial features, and other extensions.

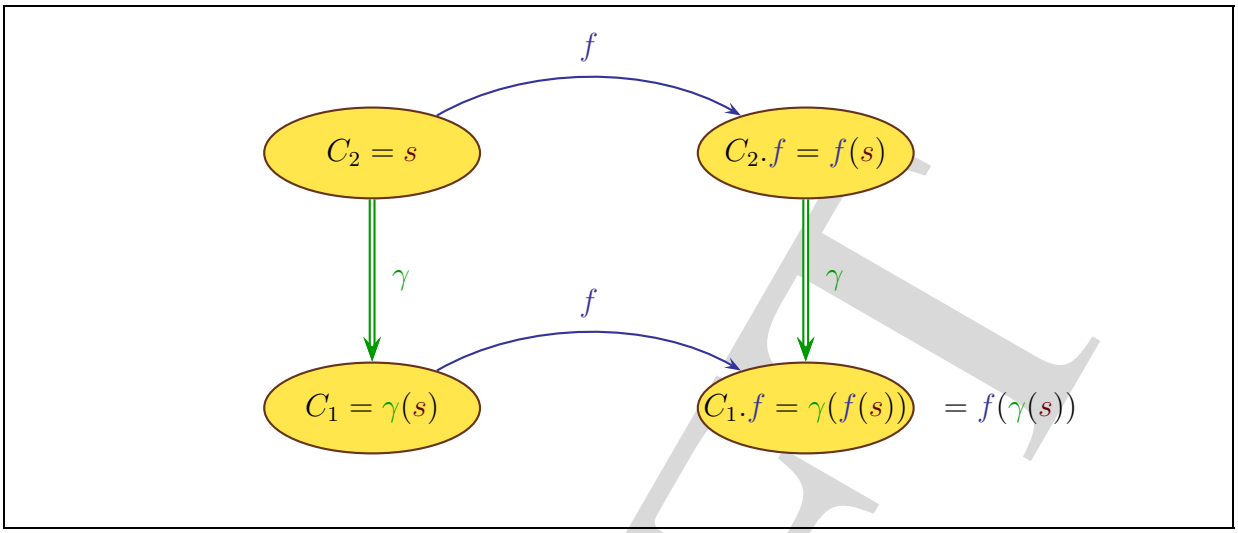


Figure 3.4: Property inheritance as $\mathcal{OSF}$ endomorphism

2. *a sorted variable* $X : s$*; or,*
3. *an attributed sorted variable* $X : s(f_1 \to t_1, \ldots, f_n \to t_n)$*;*

where X *is an element in a countably infinite set of variables* $\mathcal{V}$*,* s *is a sort in* $\mathcal{S}$ *and,* $f_1, \ldots, f_n$ *are features in* $\mathcal{F}$ *and* $t_1, \ldots, t_n$ *are* $\mathcal{OSF}$ *terms, for any* $n \geq 1$*.*

In such a term t, the variable X is called its *root* variable, and is referred to as $\mathbf{ROOT}(t)$. The set of all variables occurring in t is defined as $\mathbf{TagSet}(t) \stackrel{\text{def}}{=} \{\, \mathbf{ROOT}(t) \,\} \cup \bigcup_{i=1}^{n} \mathbf{TagSet}(t_i)$. In what follows, we consider "*variable*" and "*tag*" to be synonymous: a logical variable points to the root of an $\mathcal{OSF}$ term as does a reference to a data structure. As justified by the following semantics given to this syntax, we shall consider the first form for an unsorted variable X that occurs nowhere else in its context as a sorted variable, as a shorthand for $X : \top$.

§ Semantics of $\mathcal{OSF}$ Terms

For a term t such as above, an $\mathcal{OSF}$ algebra $\mathcal{A}$, and a specific $\mathcal{A}$-valuation $\upsilon : \mathcal{V} \mapsto D^{\mathcal{A}}$, the *denotation* $[\![t]\!]^{\mathcal{A},\upsilon}$ of t is given by:

$$[\![t]\!]^{\mathcal{A},\upsilon} \stackrel{\text{def}}{=} \{\, \upsilon(X) \,\} \cap s^{\mathcal{A}} \cap \bigcap_{1 \leq i \leq n} (f_i^{\mathcal{A}})^{-1}([\![t_i]\!]^{\mathcal{A},\upsilon}). \tag{3.2}$$

Hence, for a fixed $\mathcal{A}$-valuation υ, $[\![t]\!]^{\mathcal{A},\upsilon}$ is either the empty set or the singleton set $\{\, \upsilon(\mathbf{ROOT}(t)) \,\}$. In fact, it is *not* the empty set if and only if the value $\upsilon(\mathbf{ROOT}(t))$ lies in the denotation of the sort s, as well as every inverse image by $(f_i^{\mathcal{A}})^{-1}$ (the reciprocal of the denotation $f_i^{\mathcal{A}}$ of each feature f_i) of the denotation of the corresponding subterm $[\![t_i]\!]^{\mathcal{A},\upsilon}$ *under the same* $\mathcal{A}$*-valuation* υ*.* Thus, the denotation of an $\mathcal{OSF}$ term t for all possible valuations of the variables is given by the set:

$$[\![t]\!]^{\mathcal{A}} \stackrel{\text{def}}{=} \bigcup_{\upsilon : \mathcal{V} \mapsto D^{\mathcal{A}}} [\![t]\!]^{\mathcal{A},\upsilon}. \tag{3.3}$$

§ $\mathcal{OSF}$ Term Subsumption

Let t and t' be two $\mathcal{OSF}$ terms. Since we have both a formal syntax and its formal semantics, there are two ways we can define $\mathcal{OSF}$ term subsumption.

Definition 3.2 (Syntactic $\mathcal{OSF}$ Term Subsumption) *We say that "t is* syntactically subsumed *by t'" (written: $t \preceq t'$) if and only if, there exists an $\mathcal{OSF}$ endomorphism γ such that $t = \gamma(t')$.*

Definition 3.3 (Semantic $\mathcal{OSF}$ Term Subsumption) *We say that "t is* semantically subsumed *by t'" (written: $t[\![\preceq]\!]t'$) if and only if, for all $\mathcal{OSF}$ algebras $\mathcal{A}$, $[\![t]\!]^{\mathcal{A}} \subseteq [\![t']\!]^{\mathcal{A}}$.*

The following theorem was established in [18] and states that the two definitions coincide.

Theorem 3.1 ($\mathcal{OSF}$ Term Subsumption) *For any $\mathcal{OSF}$ terms t and t',*

$$t \preceq t' \iff t[\![\preceq]\!]t'.$$

§ $\mathcal{OSF}$ Term Normal Form

An $\mathcal{OSF}$ term $t = X : s(f_1 \to t_1, \ldots, f_n \to t_n)$ is said to be *"in normal form"* whenever all the following properties hold:

- s is a non-bottom sort in $\mathcal{S}$;
- $f_1, \ldots, f_n$ are pairwise distinct features in $\mathcal{F}$, for all $n \geq 1$;
- $t_1, \ldots, t_n$ are all $\mathcal{OSF}$ terms in *normal* form, for all $n \geq 1$; and,
- every variable in $\mathbf{TagSet}(t)$ is sorted at most once.[6]

With these criteria, we note hence that all the examples of $\mathcal{OSF}$ terms shown in the previous section are, formally speaking, in normal form. Such a normal form ensures that it is devoid of actual or potential inconsistencies. We shall call an $\mathcal{OSF}$ term in normal form a "ψ-*term*," and designate as Ψ the set of all ψ-terms.[7] How to *normalize* an $\mathcal{OSF}$ term into a semantically equivalent ψ-term is explained next.

§ From $\mathcal{OSF}$ Terms to $\mathcal{OSF}$ Constraints

A logical reading of an $\mathcal{OSF}$ term is immediate as its information content can be characterized by a simple formula. For this purpose, we need a simple clausal language as follows.

An atomic $\mathcal{OSF}$ *constraint* is one of (1) $X : s$, (2) $X \doteq X'$, or (3) $X.f \doteq X'$, where X and X' are variables in $\mathcal{V}$, s is a sort in $\mathcal{S}$, and f is a feature in $\mathcal{F}$. A (conjunctive) $\mathcal{OSF}$ *constraint* is a conjunction (*i.e.*, a set) of atomic $\mathcal{OSF}$ constraints $\phi_1 \,\&\, \ldots \,\&\, \phi_n$. Given an $\mathcal{OSF}$ algebra $\mathcal{A}$,

[6] That is, if $X \in \mathbf{TagSet}(t)$ occurs in t both as $X : s$ and $X : s'$, then $s = \top$ or $s' = \top$.

[7] The expression "ψ-term" was introduced originally by the first author in his PhD thesis [3] as a shorthand for "*Property Structure Inheritance Term*," as a formalization of some parts of Ron Brachman's "*Structured Inheritance Networks*" (or "*SI-Nets*") defined informally in his PhD thesis [43].

we say that an $\mathcal{OSF}$ constraint ϕ is *satisfiable* in $\mathcal{A}$ with a valuation $\upsilon : \mathcal{V} \mapsto D^{\mathcal{A}}$ (and we write this as "$\mathcal{A}, \upsilon \models \phi$") whenever:

$$\begin{array}{lll} \mathcal{A}, \upsilon \models X : s & \textbf{iff} & \upsilon(X) \in s^{\mathcal{A}}; \\ \mathcal{A}, \upsilon \models X \doteq Y & \textbf{iff} & \upsilon(X) = \upsilon(Y); \\ \mathcal{A}, \upsilon \models X.f \doteq Y & \textbf{iff} & f^{\mathcal{A}}(\upsilon(X)) = \upsilon(Y) \\ \mathcal{A}, \upsilon \models \phi \ \& \ \phi' & \textbf{iff} & \mathcal{A}, \upsilon \models \phi \ \textbf{and} \ \mathcal{A}, \upsilon \models \phi'. \end{array} \tag{3.4}$$

We can always associate with an $\mathcal{OSF}$ term $t = X : s(f_1 \rightarrow t_1, \ldots, f_n \rightarrow t_n)$ a corresponding $\mathcal{OSF}$ constraint $\varphi(t)$ as follows:

$$\begin{array}{lcccccccc} \varphi(t) & \stackrel{\text{def}}{=} & X : s & \& & X.f_1 \doteq X_1 & \& & \ldots & \& & X.f_n \doteq X_n \\ & & & \& & \varphi(t_1) & & \& & \ldots & \& \ \varphi(t_n) \end{array} \tag{3.5}$$

where $X_1, \ldots, X_n$ are the roots of $t_1, \ldots, t_n$, respectively. We say that $\varphi(t)$ is obtained from *dissolving* the $\mathcal{OSF}$ term t. The following theorem, also established in [18], states that the algebraic denotation of an $\mathcal{OSF}$ term as a set and the logical semantics of its dissolved form coincide exactly.

THEOREM 3.2 (COINCIDING ALGEBRAIC AND LOGICAL SEMANTICS OF $\mathcal{OSF}$ TERMS)

$$[\![t]\!]^{\mathcal{A}} = \{\, \upsilon(\textbf{ROOT}(t)) \mid \upsilon \in \textbf{VALUATIONS}(\mathcal{A}) \textit{ and } \mathcal{A}, \upsilon \models \varphi(t) \,\}.$$

3.1.3 $\mathcal{OSF}$ Lattice Structure

As seen in the two previous sections, the $\mathcal{FOT}$ as a data structure can be extended and made more expressive as a knowledge structure into a rooted $\mathcal{OSF}$ graph written as an $\mathcal{OSF}$ term. It is more expressive because it can be used to represent specific data as well as generic conceptual structures. Formally, it is a more generic formal scheme than $\mathcal{FOT}$s and their lattice-theoretic properties exposed by Reynolds and Plotkin. These are just a special case of $\mathcal{OSF}$ terms and their lattice operations.[8] The operations of unification and generalization can be extended from $\mathcal{FOT}$s to $\mathcal{OSF}$ terms, obtaining improved expressivity while keeping the same complexity, to provide a simple and intuitive semantics for knowledge and data structures of partial descriptions. In [3] and [5], it was formally established that the Reynolds-Plotkin $\mathcal{FOT}$ lattice-theoretic algebra extends naturally to $\mathcal{OSF}$ structures, which express order-sorted functional commutative diagrams among sets as used in mathematics to express equational constraints among feature compositions.

The subsumption ordering defined on ψ-terms is an extension of the subsumption ordering on $\mathcal{FOT}$s. Specifically, for an $\mathcal{OSF}$ term t to subsume an $\mathcal{OSF}$ term t' (*i.e.*, $t' \preceq t$), there must be a mapping γ from the tags of t to the tags of t'—*i.e.*, $\gamma : \textbf{TagSet}(t) \mapsto \textbf{TagSet}(t')$—that respects sorting and preserves feature applications. Such a mapping is called a *sort-consistent feature-preserving endomorphic mapping* and is the formal notion extending the concept of $\mathcal{FOT}$ substitution to ψ-terms. To emphasize the existence of this mapping γ whenever $t' \preceq t$, we shall also write $t' = \gamma(t)$.

[8] See Appendix Section B.3.

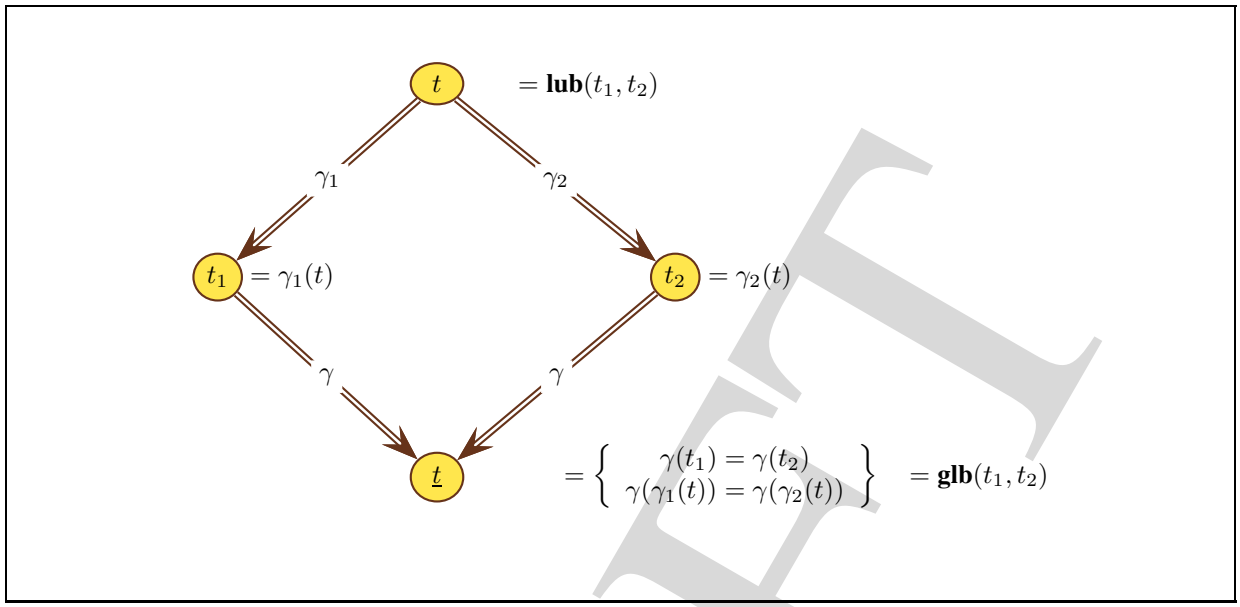


Figure 3.5: $\mathcal{OSF}$ subsumption lattice operations

Formally, such a mapping γ associates each tag symbol of a ψ-term t to a tag symbol of ψ-term $t' = \gamma(t)$ in such as way that, whenever:

$$\{\, X : d, X.f \doteq Y, Y : r \,\} \;\subseteq\; \varphi(t),$$

then necessarily, for some sorts d' and r' s.t. $d' \preceq d$ and $r' \preceq r$:

$$\{\, \gamma(X) : d', \gamma(X).f \doteq \gamma(Y), \gamma(Y) : r' \,\} \;\subseteq\; \varphi(t') = \varphi(\gamma(t)).$$

This, as pictured in the diagram in Figure 3.4, captures formally attribute inheritance as used in object-oriented programming and knowledge representation, with the added bonus of providing a means to perform deduction (unification) and induction (generalization) on such structures.

As shown in Figure 3.5, endomorphic mappings define lattice operations on ψ-terms whose sets of tags have been consistently renamed apart in exactly the same way as "renaming substitutions" do for $\mathcal{FOT}$s.[9] Just like $\mathcal{FOT}$s with substitutions, whenever two ψ-terms t and t' are such that $\exists\gamma$ s.t. $t' = \gamma(t)$ and $\exists\gamma'$ s.t. $t = \gamma'(t')$, they are said to be *identical "up to tag renaming."* This is because necessarily $\gamma' = \gamma^{-1}$ and $\gamma = \gamma'^{-1}$ make up a pair of mutually inverse "*one-to-one onto*" tag mappings (bijections). How to compute these endomorphic mappings by $\mathcal{OSF}$ constraint normalization is shown next.

§ $\mathcal{OSF}$ Unification: Deduction by Constraint Normalization

Definition 3.4 (Solved $\mathcal{OSF}$ Constraint) *An $\mathcal{OSF}$ constraint ϕ is said to be in* solved form *if for every variable X, ϕ contains:*

[9] See Figure 2.1 in Section 2.3.

SORT INTERSECTION:

$$\frac{\phi \;\&\; X : s \;\&\; X : s'}{\phi \;\&\; X : s \curlywedge s'}$$

FEATURE FUNCTIONALITY:

$$\frac{\phi \;\&\; X.f \doteq Y \;\&\; X.f \doteq Y'}{\phi \;\&\; X.f \doteq Y \;\&\; Y \doteq Y'}$$

INCONSISTENT SORT:

$$\frac{\phi \;\&\; X : \bot}{\mathfrak{false}}$$

TAG ELIMINATION:

$$\frac{\phi \;\&\; X \doteq Y}{\phi[X/Y] \;\&\; X \doteq Y} \; [Y \in \mathbf{TagSet}(\phi)]$$

Figure 3.6: Constraint normalization rules for $\mathcal{OSF}$ unification

- *at most one sort constraint* $X : s$ *and* $s \neq \bot$*;*
- *at most one feature constraint* $X.f \doteq Y$ *for any* $X.f$ *that occurs in* ϕ*;*

and whenever $X \doteq Y \in \phi$*, then* X *does not appear anywhere else in* ϕ*.*

Given an $\mathcal{OSF}$ constraint ϕ, non-deterministically applying any applicable rule among the rules shown in Figure 3.6 until none applies always terminates either in the inconsistent constraint $\mathfrak{false}$ or in a solved $\mathcal{OSF}$ constraint. The notation $\phi[X/Y]$ in Rule **TAG ELIMINATION** stands for the constraint ϕ in which all occurrences of the tag Y have been replaced with the tag X. As a result of applying this rule, the tag being eliminated (Y) will occur nowhere else in the normal form except as the right-hand side of constraint $X \doteq Y$.

In [18], it is shown that the rules in Figure 3.6 are:

1. *solution-preserving*—for each rule, the set of solutions of the posterior constraint is equal to the set of solutions of the prior constraint;
2. *finite terminating*—they always terminate after a finite number of formula transformations;
3. *confluent*—they always end up with the same constraint up to consistent tag renaming.

Furthermore, they always result in a normal form that is either the inconsistent constraint $\mathfrak{false}$ or a consistent $\mathcal{OSF}$ constraint in solved form. These rules are all we need to perform the unification of two ψ-terms. Namely, two ψ-terms t_1 and t_2 are unifiable if and only if the normal form of the $\mathcal{OSF}$ constraint $\mathbf{ROOT}(t_1) \doteq \mathbf{ROOT}(t_2) \;\&\; \varphi(t_1) \;\&\; \varphi(t_2)$ is not $\mathfrak{false}$. For a detailed example of how applying these rules corresponds to computing the **glb** of two ψ-terms, see Example B.2 in Appendix B.1.

An $\mathcal{OSF}$ constraint ϕ in solved form is always satisfiable in a canonical interpretation structure; *viz.*, the $\mathcal{OSF}$ graph algebra Ψ [18]. As a consequence, the $\mathcal{OSF}$-constraint normalization rules yield a decision procedure for the satisfiability of $\mathcal{OSF}$ constraints. This decision procedure is also operationally efficient [13]. One important reason for its efficiency is that computing sort intersection as specified by Rule **SORT INTERSECTION** can be done in constant time by encoding sorts as binary vectors as shown in [12]. This results in tremendous speed performance

when compared to encoding a class hierarchy's partial order using First-Order Logic implication, even when resorting to proof memoization (and thus paying a hefty space and time overhead, as done in for example in [81]).

§ $\mathcal{OSF}$ Generalization: Induction by Constraint Normalization

Just like $\mathcal{FOT}$s, ψ-terms also possess an operation of generalization that is dual to unification ([3], [21]). The operation is very similar, with the additional taking into account of common symbolic features rather then all contiguous positions, as well as partially ordered sorts denoting sets and the sort ordering denoting set inclusion.

What follows is a formal specification of the $\mathcal{OSF}$ generalization operation. This formulation is different than the one used in [21].[10] It is equivalent to it, however, expressed with this now familiar notation we used for $\mathcal{FOT}$ generalization in Section 2.5, extended to $\mathcal{OSF}$ terms; that is,

$$\begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix} \vdash \begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \psi \begin{pmatrix}\gamma'_1\\ \gamma'_2\end{pmatrix}$$

where, ψ is the ψ-term generalizing the ψ-terms ψ_i, for $i = 1, 2$, and γ_i and γ'_i are, respectively, the judgement's pairs of *prior* and *posterior* tag maps.[11] In order to do this, all the generalization constraint normalization rules must be reformulated into syntax-directed judgment axioms and rules directly on the syntactic form of ψ-terms (*i.e.*, $\mathcal{OSF}$ terms in normal form) defined in Equation (3.1), rather than on dissolved $\mathcal{OSF}$ constraints as done in [21]. The dissolved form is more convenient for unification while traditional term syntax is more convenient for our formulation of generalization, as for that of $\mathcal{FOT}$s, (and the fuzzification of the corresponding lattice operation).

Since our definition of the generalizing operation's definition will assume that its arguments are ψ-terms (*i.e.*, $\mathcal{OSF}$ terms in normal form), each tag symbol occurring in either term will always be the root tag of a unique ψ-term. This property is easily verified to be preserved by the axiom and the rule of Figure 3.7. Thus, a sortless occurrence of a tag always refers to (*i.e.*, is the root tag of) this unique ψ-term. If none of a tag symbol's occurrences is sorted, then this tag is the root of a common occurrence of the most general sort: $\top$ (*i.e.*, "*anything*").

Axiom **Equal Tags** in Figure 3.7 simply states that generalizing a pair made of the same ψ-term results in this ψ-term and the posterior tag maps are the same as the prior ones.

Rule **Unequal Tags** in Figure 3.7 uses an "unapply" operation '$\uparrow$' which is defined as follows:

$$\begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \uparrow \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix} \stackrel{\text{def}}{=} \begin{cases} \begin{pmatrix}X : \ldots\\ X : \ldots\end{pmatrix} & \text{if } \exists\, X \text{ s.t. } \gamma_i(X) = \mathbf{ROOT}(\psi_i),\ \text{for } i = 1, 2; \\ \begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} & \text{otherwise.} \end{cases} \tag{3.6}$$

[10] See Appendix A.6

[11] A tag map is a tag-to-tag substitution.

EQUAL TAGS

$$\begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix} \vdash \begin{pmatrix}\psi\\ \psi\end{pmatrix} \psi \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}$$

UNEQUAL TAGS

$$\left[\begin{array}{l} X \neq Y; \\ m, n, p \geq 0 \text{ and } \{h_1, \ldots, h_p\} \stackrel{\text{def}}{=} \{f_1, \ldots, f_m\} \cap \{g_1, \ldots, g_n\} \text{ s.t. } h_k \stackrel{\text{def}}{=} f_k = g_k \text{ for all } k = 1, \ldots, p; \\ \gamma_i^0 \stackrel{\text{def}}{=} \gamma_1 \circ \{X/Z\} \text{ and } \gamma_2^0 \stackrel{\text{def}}{=} \gamma_2 \circ \{Y/Z\}, \text{ where } Z \text{ is a new variable} \end{array}\right]$$

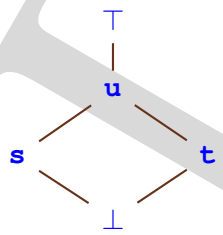
$$\frac{\begin{pmatrix}\gamma_1^0\\ \gamma_2^0\end{pmatrix} \vdash \begin{pmatrix}\psi_1\\ \xi_1\end{pmatrix} \uparrow \begin{pmatrix}\gamma_1^0\\ \gamma_2^0\end{pmatrix} \chi_1 \begin{pmatrix}\gamma_1^1\\ \gamma_2^1\end{pmatrix} \quad \cdots \quad \begin{pmatrix}\gamma_1^{p-1}\\ \gamma_2^{p-1}\end{pmatrix} \vdash \begin{pmatrix}\psi_p\\ \xi_p\end{pmatrix} \uparrow \begin{pmatrix}\gamma_1^{p-1}\\ \gamma_2^{p-1}\end{pmatrix} \chi_p \begin{pmatrix}\gamma_1^{p}\\ \gamma_2^{p}\end{pmatrix}}{\begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix} \vdash \begin{pmatrix}X : s\,(f_i \rightarrow \psi_i)_{i=1}^m\\ Y : t\,(g_j \rightarrow \xi_j)_{j=1}^n\end{pmatrix} Z : s \vee t\,(h_k \rightarrow \chi_k)_{k=1}^p \begin{pmatrix}\gamma_1^{p}\\ \gamma_2^{p}\end{pmatrix}}$$

Figure 3.7: Judgment-based $\mathcal{OSF}$ generalization axiom and rule

This has the same purpose as the unapply operation used in $\mathcal{FOT}$ generalization judgments: identify in the prior pair of tag maps (γ_1, γ_2) whether or not they already map a common variable (X) to the roots of the pair of ψ-terms to be generalized (ψ_1, ψ_2). If so, the result of the unapplication is the pair made of the same ψ-term rooted in X $(X : \ldots)$; if not, it is the original pair of ψ-terms (ψ_1, ψ_2).

Rule $\mathcal{OSF}$ **UNEQUAL TAGS** of Figure 3.7 states that generalizing two ψ-terms $\psi_1 \stackrel{\text{def}}{=} X : s\,(f_i \rightarrow \psi_i)_{i=0}^m$ and $\psi_2 \stackrel{\text{def}}{=} Y : t\,(g_j \rightarrow \xi_j)_{j=0}^n$ results in the ψ-term $\psi_1 \vee \psi_2 \stackrel{\text{def}}{=} Z : s \vee t\,(h_k \rightarrow \chi_k)_{k=0}^p$, where the set of features of the resulting ψ-term is the intersection of the sets of features of ψ_1 and ψ_2 (*i.e.*, the features they have in common), and Z is a new variable, when each subterm corresponds to generalizing each of the corresponding pairs of subterms under all common features. Note that this can be done in any order, as long as each subterm judgment is validated with its pair of prior tag maps equal to the preceding judgment's pair of posterior tag maps.

Example 3.1 $\mathcal{OSF}$ **term generalization** — Let us consider the sort signature:



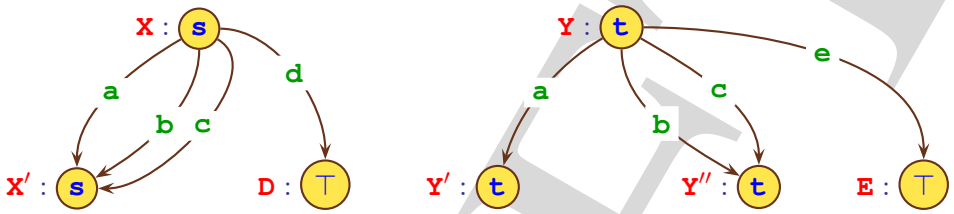
and the two ψ-terms:

$$\psi_1 \stackrel{\text{def}}{=} \mathtt{X} : \mathtt{s}(\mathtt{a} \to \mathtt{X}' : \mathtt{s},\ \mathtt{b} \to \mathtt{X}',\ \mathtt{c} \to \mathtt{X}',\ \mathtt{d} \to \mathtt{D})$$

and:

$$\psi_2 \stackrel{\text{def}}{=} \mathtt{Y} : \mathtt{t}(\mathtt{a} \to \mathtt{Y}' : \mathtt{t},\ \mathtt{b} \to \mathtt{Y}'' : \mathtt{t},\ \mathtt{c} \to \mathtt{Y}'',\ \mathtt{e} \to \mathtt{E})$$

corresponding to the following two $\mathcal{OSF}$ graphs:



Let us now consider the following $\mathcal{OSF}$ generalization judgment constraint to resolve:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix} \vdash \begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \psi_1 \vee \psi_2 \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}$$

in which the ψ-term $\psi_1 \vee \psi_2$ and the tag maps γ_1 and γ_2 are to be determined by normalizing this judgment according to the axiom and rule of Figure 3.7.

We start with the judgment:

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix} \vdash \begin{pmatrix}\mathtt{X} : \mathtt{s}(\mathtt{a} \to \mathtt{X}' : \mathtt{s},\ \mathtt{b} \to \mathtt{X}',\ \mathtt{c} \to \mathtt{X}',\ \mathtt{d} \to \mathtt{D})\\ \mathtt{Y} : \mathtt{t}(\mathtt{a} \to \mathtt{Y}' : \mathtt{t},\ \mathtt{b} \to \mathtt{Y}'' : \mathtt{t},\ \mathtt{c} \to \mathtt{Y}'',\ \mathtt{e} \to \mathtt{E})\end{pmatrix} \psi_1 \vee \psi_2 \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}.$$

Applying Rule UNEQUAL TAGS, since $\mathtt{s} \vee \mathtt{t} = \mathtt{u}$, keeping only common features and introducing a new tag $\mathtt{Z}$, this yields $\psi_1 \vee \psi_2 = \mathtt{Z} : \mathtt{u}(\mathtt{a} \to \psi', \mathtt{b} \to \psi'', \mathtt{c} \to \psi''')$ and this judgment becomes the following sequence of three judgments:

$$\begin{pmatrix}\{\mathtt{X}/\mathtt{Z}\}\\ \{\mathtt{Y}/\mathtt{Z}\}\end{pmatrix} \vdash \begin{pmatrix}\mathtt{X}' : \mathtt{s}\\ \mathtt{Y}' : \mathtt{t}\end{pmatrix} \uparrow \begin{pmatrix}\{\mathtt{X}/\mathtt{Z}\}\\ \{\mathtt{Y}/\mathtt{Z}\}\end{pmatrix} \psi' \begin{pmatrix}\gamma'_1\\ \gamma'_2\end{pmatrix},$$

$$\begin{pmatrix}\gamma'_1\\ \gamma'_2\end{pmatrix} \vdash \begin{pmatrix}\mathtt{X}' : \mathtt{s}\\ \mathtt{Y}'' : \mathtt{t}\end{pmatrix} \uparrow \begin{pmatrix}\gamma'_1\\ \gamma'_2\end{pmatrix} \psi'' \begin{pmatrix}\gamma''_1\\ \gamma''_2\end{pmatrix},$$

$$\begin{pmatrix}\gamma''_1\\ \gamma''_2\end{pmatrix} \vdash \begin{pmatrix}\mathtt{X}' : \mathtt{s}\\ \mathtt{Y}'' : \mathtt{t}\end{pmatrix} \uparrow \begin{pmatrix}\gamma''_1\\ \gamma''_2\end{pmatrix} \psi''' \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}.$$

Evaluating the unapplication in the first of these judgments, it becomes:

$$\begin{pmatrix}\{\mathtt{X}/\mathtt{Z}\}\\ \{\mathtt{Y}/\mathtt{Z}\}\end{pmatrix} \vdash \begin{pmatrix}\mathtt{X}' : \mathtt{s}\\ \mathtt{Y}' : \mathtt{t}\end{pmatrix} \psi' \begin{pmatrix}\gamma'_1\\ \gamma'_2\end{pmatrix},$$

to which we can apply again Rule UNEQUAL TAGS, since $\mathtt{s} \vee \mathtt{t} = \mathtt{u}$, introducing a new tag $\mathtt{Z}'$, this first judgment becomes:

$$\begin{pmatrix}\{\mathtt{X}/\mathtt{Z}\}\\ \{\mathtt{Y}/\mathtt{Z}\}\end{pmatrix} \vdash \begin{pmatrix}\mathtt{X}' : \mathtt{s}\\ \mathtt{Y}' : \mathtt{t}\end{pmatrix} \mathtt{Z}' : \mathtt{u} \begin{pmatrix}\{\mathtt{X}/\mathtt{Z}, \mathtt{X}'/\mathtt{Z}'\}\\ \{\mathtt{Y}/\mathtt{Z}, \mathtt{Y}'/\mathtt{Z}'\}\end{pmatrix}.$$

This, in turn, makes the second judgment in the sequence become:

$$\begin{pmatrix}\{\mathtt{X}/\mathtt{Z},\mathtt{X}'/\mathtt{Z}'\}\\ \{\mathtt{Y}/\mathtt{Z},\mathtt{Y}'/\mathtt{Z}'\}\end{pmatrix} \vdash \begin{pmatrix}\mathtt{X}':\mathtt{s}\\ \mathtt{Y}'':\mathtt{t}\end{pmatrix} \uparrow \begin{pmatrix}\{\mathtt{X}/\mathtt{Z},\mathtt{X}'/\mathtt{Z}'\}\\ \{\mathtt{Y}/\mathtt{Z},\mathtt{Y}'/\mathtt{Z}'\}\end{pmatrix} \psi'' \begin{pmatrix}\gamma_1''\\ \gamma_2''\end{pmatrix},$$

and after evaluating the unapplication, it becomes:

$$\begin{pmatrix}\{\mathtt{X}/\mathtt{Z},\mathtt{X}'/\mathtt{Z}'\}\\ \{\mathtt{Y}/\mathtt{Z},\mathtt{Y}'/\mathtt{Z}'\}\end{pmatrix} \vdash \begin{pmatrix}\mathtt{X}':\mathtt{s}\\ \mathtt{Y}'':\mathtt{t}\end{pmatrix} \psi'' \begin{pmatrix}\gamma_1''\\ \gamma_2''\end{pmatrix},$$

to which we can apply again Rule **UNEQUAL TAGS**, since $\mathtt{s} \vee \mathtt{t} = \mathtt{u}$, introducing a new tag $\mathtt{Z}''$, this second judgment becomes:

$$\begin{pmatrix}\{\mathtt{X}/\mathtt{Z},\mathtt{X}'/\mathtt{Z}'\}\\ \{\mathtt{Y}/\mathtt{Z},\mathtt{Y}'/\mathtt{Z}'\}\end{pmatrix} \vdash \begin{pmatrix}\mathtt{X}':\mathtt{s}\\ \mathtt{Y}'':\mathtt{t}\end{pmatrix} \mathtt{Z}'':\mathtt{u} \begin{pmatrix}\{\mathtt{X}/\mathtt{Z},\mathtt{X}'/\mathtt{Z}',\mathtt{X}'/\mathtt{Z}''\}\\ \{\mathtt{Y}/\mathtt{Z},\mathtt{Y}'/\mathtt{Z}',\mathtt{Y}''/\mathtt{Z}''\}\end{pmatrix}.$$

This then makes the third judgment in the initial sequence become:

$$\begin{pmatrix}\{\mathtt{X}/\mathtt{Z},\mathtt{X}'/\mathtt{Z}',\mathtt{X}'/\mathtt{Z}''\}\\ \{\mathtt{Y}/\mathtt{Z},\mathtt{Y}'/\mathtt{Z}',\mathtt{Y}''/\mathtt{Z}''\}\end{pmatrix} \vdash \begin{pmatrix}\mathtt{X}':\mathtt{s}\\ \mathtt{Y}'':\mathtt{t}\end{pmatrix} \uparrow \begin{pmatrix}\{\mathtt{X}/\mathtt{Z},\mathtt{X}'/\mathtt{Z}',\mathtt{X}'/\mathtt{Z}''\}\\ \{\mathtt{Y}/\mathtt{Z},\mathtt{Y}'/\mathtt{Z}',\mathtt{Y}''/\mathtt{Z}''\}\end{pmatrix} \psi''' \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}.$$

But now, because both $\mathtt{X}'/\mathtt{Z}''$ and $\mathtt{Y}''/\mathtt{Z}''$ are in the two respective prior tag maps, the unapplication in this judgment comes to:

$$\begin{pmatrix}\mathtt{X}':\mathtt{s}\\ \mathtt{Y}'':\mathtt{t}\end{pmatrix} \uparrow \begin{pmatrix}\{\mathtt{X}/\mathtt{Z},\mathtt{X}'/\mathtt{Z}',\mathtt{X}'/\mathtt{Z}''\}\\ \{\mathtt{Y}/\mathtt{Z},\mathtt{Y}'/\mathtt{Z}',\mathtt{Y}''/\mathtt{Z}''\}\end{pmatrix} = \begin{pmatrix}\mathtt{Z}'':\mathtt{u}\\ \mathtt{Z}'':\mathtt{u}\end{pmatrix}$$

which makes the third judgment become:

$$\begin{pmatrix}\{\mathtt{X}/\mathtt{Z},\mathtt{X}'/\mathtt{Z}',\mathtt{X}'/\mathtt{Z}''\}\\ \{\mathtt{Y}/\mathtt{Z},\mathtt{Y}'/\mathtt{Z}',\mathtt{Y}''/\mathtt{Z}''\}\end{pmatrix} \vdash \begin{pmatrix}\mathtt{Z}'':\mathtt{u}\\ \mathtt{Z}'':\mathtt{u}\end{pmatrix} \psi''' \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix};$$

and using Axiom **EQUAL TAGS**, this third judgment becomes:

$$\begin{pmatrix}\{\mathtt{X}/\mathtt{Z},\mathtt{X}'/\mathtt{Z}',\mathtt{X}'/\mathtt{Z}''\}\\ \{\mathtt{Y}/\mathtt{Z},\mathtt{Y}'/\mathtt{Z}',\mathtt{Y}''/\mathtt{Z}''\}\end{pmatrix} \vdash \begin{pmatrix}\mathtt{Z}'':\mathtt{u}\\ \mathtt{Z}'':\mathtt{u}\end{pmatrix} \mathtt{Z}'':\mathtt{u} \begin{pmatrix}\{\mathtt{X}/\mathtt{Z},\mathtt{X}'/\mathtt{Z}',\mathtt{X}'/\mathtt{Z}''\}\\ \{\mathtt{Y}/\mathtt{Z},\mathtt{Y}'/\mathtt{Z}',\mathtt{Y}''/\mathtt{Z}''\}\end{pmatrix}.$$

and terminates the proof.

This results in the ψ-term that is the generalization of ψ_1 and ψ_2:

$$\psi_1 \vee \psi_2 = \mathtt{Z}:\mathtt{u}(\mathtt{a} \to \mathtt{Z}':\mathtt{u},\ \mathtt{b} \to \mathtt{Z}'':\mathtt{u},\ \mathtt{c} \to \mathtt{Z}'')$$

with the corresponding tag maps:

$$\gamma_1 = \{\mathtt{X}/\mathtt{Z},\ \mathtt{X}'/\mathtt{Z}',\ \mathtt{X}'/\mathtt{Z}''\}$$

and:

$$\gamma_2 = \{\mathtt{Y}/\mathtt{Z},\ \mathtt{Y}'/\mathtt{Z}',\ \mathtt{Y}''/\mathtt{Z}''\}$$

as pictured in Figure 3.8.

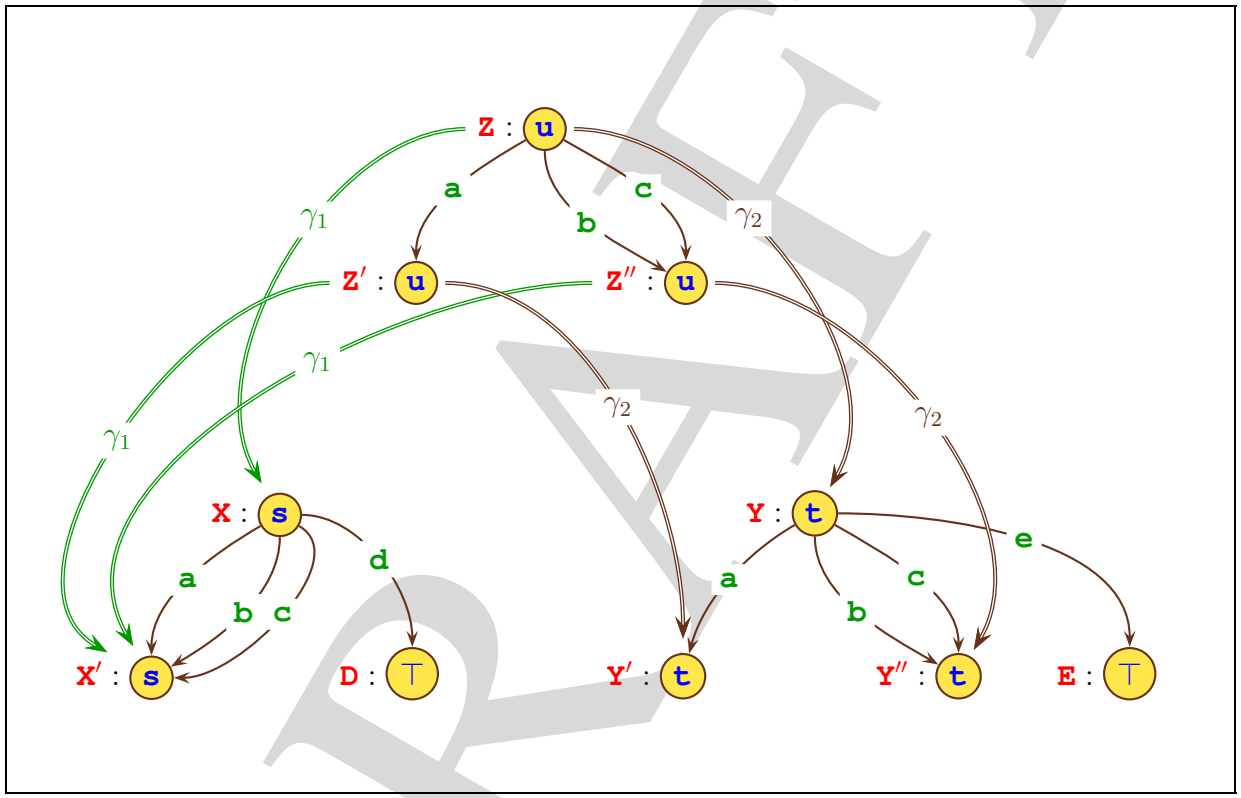


Figure 3.8: Example of $\mathcal{OSF}$ graph endomorphisms realized by $\mathcal{OSF}$ generalization

§ Misaligned Features

We will assume that each sort $s \in \mathcal{S}$ has a "feature arity" $\mathbf{arity}(s) : \mathcal{S} \to \mathbf{2}^{\mathcal{F}}$ which associates to the sort s a finite set of features consistent with the ordering on sorts. This is expressed formally as the following lattice homomorphism:

$$\mathbf{arity}(s \curlywedge t) \;=\; \mathbf{arity}(s) \cup \mathbf{arity}(t) \tag{3.7}$$

$$\mathbf{arity}(s \curlyvee t) \;=\; \mathbf{arity}(s) \cap \mathbf{arity}(t) \tag{3.8}$$

for all sorts s and t in $\mathcal{S}$. Note that the anti-monotocity of sort subsumption and feature set inheritance, since Equation (3.7) and Equation (3.8) imply necessarily that:

$$s \preceq t \;\;\Longrightarrow\;\; \mathbf{arity}(t) \subseteq \mathbf{arity}(s); \tag{3.9}$$

i.e., the more specific the sort, the larger its arity feature set.

Again, note that the arity of a sort symbol is not a natural number as is the case for a function symbol where is counts their argument positions. The arity of a sort is a set of features which are the admissible field names of structure components rooted in a node of this sort. It is a set because order does not matter and it may also be empty. In this setting, for a function symbol $f/n \in \Sigma_n$ seen as a sort, $\mathbf{arity}(f/n) \stackrel{\text{def}}{=} \{1, \ldots, n\}$, and the arity of a constant is the empty set.

Inspired by the possiblities demonstrated when comparing $\mathcal{FOT}$s using similar functors with misaligned argument positions, sorts may as well be made to tolerate feature sets that are one another's finite permutations. By default (and up to now), when comparing two sorts s and t, every feature name is mapped to itself—*i.e.*, feature mapping from one sort to another is always the identity on $\mathcal{F}$. But now, extending the same idea as for function symbols in $\mathcal{FOT}$s, for any pair of sorts s and t in $\mathcal{S}$, we assume defined a feature mapping $\pi_{st} : \mathcal{F} \to \mathcal{F}$ that verifies the following properties.

- π_{st} is *injective* (*i.e.*, one-to-one); *i.e.*, for any pair of sorts s and t in $\mathcal{S}$:

  $$f \neq f' \to \pi_{st}(f) \neq \pi_{st}(f') \tag{3.10}$$

  for all *features* f and f' in $\mathcal{F}$;[12]

- π_{st} is the *identity almost everywhere on* $\mathcal{F}$ except on a finite (possibly empty) subset of features in $\mathbf{arity}(s)$ and the non-identical image of this set is a (possibly empty) subset of features in $\mathbf{arity}(t)$;

- it is *self-consistent*; *i.e.*,

  $$\pi_{ss} = \mathbb{1}_{\mathcal{F}} \tag{3.11}$$

  for all sorts s in $\mathcal{S}$;

[12] Or, equivalently: $\pi_{st}(f) = \pi_{st}(f') \to f = f'$.

- it is *inverse-consistent*; *i.e.*,

$$\pi_{st} = \pi_{ts}^{-1} \tag{3.12}$$

for all sorts s and t in $\mathcal{S}$;

- it is *composition-consistent*; *i.e.*,

$$\pi_{tu} \circ \pi_{st} = \pi_{su} \tag{3.13}$$

for all sorts s, t, u in $\mathcal{S}$—that is, $\pi_{tu}(\pi_{st}(f)) = \pi_{su}(f)$, for all feature symbols $f \in \mathcal{F}$;

- it is *order-consistent*; *i.e.*, if any sorts s, s', t, and t' in $\mathcal{S}$ are such that $s \preceq s'$ and $t \preceq t'$, then this necessarily implies that,

$$\pi_{s't'}^{\neq} \subseteq \pi_{st}^{\neq} \tag{3.14}$$

where $\pi_{st}^{\neq} \stackrel{\text{def}}{=} \{\langle f, \pi_{st}(f)\rangle \,|\, \pi_{st}(f) \neq f\}$. This ensures that the same feature is always mapped to the same feature along a sort-order chain.

Authors' comment: *The algebraically-minded reader will probably flinch. In particular, isn't this π mapping the "*missing link*" (so to speak) needed in order to complete a formal connection between our $\mathcal{OSF}$ formalism and Category Theory? We, the authors, agree that this does look, feel, and taste like a classical so-called* functor of categories*. Namely, sorted (or function-symbol) nodes can be seen as objects and their features (and/or positions) can then be seen semantically as categorical arrows between objects. This is because they denote functions that are composable—the feature of a feature is a (semantically composed) function—and all compositions out of a node that converge to a common node must commute (sharing a tag). Then, the field- (and/or position-) alignment mappings π from one sort to another (subject to the coherence constraints above) can quite be seen as a* fuzzy categorical endofunctor *for the (strict monoidal) category $\langle \mathcal{S}, \mathbf{2}^{\mathcal{F}} \mapsto \mathbf{2}^{\mathcal{F}} \rangle$.*[13] *For our present purposes, we will specify next this feature mapping more operationally by substituting the features yielding a sort t out of a tag X of sort s that occur in an $\mathcal{OSF}$ constraint set using the* $\mathbf{fmap}_X^{\pi_{st}}$ *construct we define below as Expression* (3.15).

§ $\mathcal{OSF}$ Unification Modulo Feature Permutation

In Rule **Sort Intersection Modulo Feature Permutation** of Figure 3.9, the function **fmap** is parameterized by (1) a variable X, and (2) a one-to-one feature mapping π_{st} associating to each feature f in $\mathcal{F}$ a unique feature $\pi_{st}(f)$ in $\mathcal{F}$. When sort s is compared with sort t, it realigns feature f for tag X of sort s with feature $\pi_{st}(f)$ of sort t. So parameterized, $\mathbf{fmap}_X^{\pi_{st}}$ transforms a conjunctive $\mathcal{OSF}$ constraint ϕ into another conjunctive $\mathcal{OSF}$ constraint $\mathbf{fmap}_X^{\pi_{st}}(\phi)$ obtained

[13] It might be worthwhile trying to make this connection formally more explicit in the style of [99] and [62], for example. However, we shall abstain here for fear of losing some readership despite our best efforts. Still, the interested reader is invited to correspond with the authors on this very subject. We may even be tempted to work out some basic ideas and share them.

SORT INTERSECTION MODULO FEATURE PERMUTATION:

$$\frac{\phi \;\&\; X : s \;\&\; X : t}{\mathbf{fmap}_X^{\pi_{st}}(\phi) \;\&\; X : s \curlywedge t}$$

Figure 3.9: Sort intersection rule for $\mathcal{OSF}$ unification modulo feature permutation

from ϕ by replacing each constraint of the form $X.f \doteq Y$ by the constraint $X.\pi_{st}(f) \doteq Y$. Formally,

$$\begin{cases} \mathbf{fmap}_X^{\pi_{st}}(\phi) \stackrel{\text{def}}{=} \phi & \left[\begin{array}{l}\text{if } X.f \doteq Y \text{ is not part of } \phi \\ \text{for any } f \in \mathcal{F} \text{ and any } Y \in \mathcal{V}\end{array}\right] \\ \mathbf{fmap}_X^{\pi_{st}}(\phi \;\&\; X.f \doteq Y) \stackrel{\text{def}}{=} \mathbf{fmap}_X^{\pi_{st}}(\phi) \;\&\; X.\pi_{st}(f) \doteq Y & [\text{otherwise}] \end{cases} \tag{3.15}$$

A feature map $\mathbf{fmap}_X^{\pi_{st}}$ that applies to an $\mathcal{OSF}$ constraint transforming it into another $\mathcal{OSF}$ constraint is naturally extended to apply as well to a ψ-term to transform it into another ψ-term as follows:

$$\mathbf{fmap}_X^{\pi_{st}}(\psi) \stackrel{\text{def}}{=} \varphi^{-1}(\mathbf{fmap}_X^{\pi_{st}}(\varphi(\psi))); \tag{3.16}$$

that is, it is defined as the ψ-term whose dissolved form is the result of applying that same feature map to the dissolved form of the original ψ-term.[14]

When π_{st} is the identity for a pair of sorts s and t, Rule **SORT INTERSECTION MODULO FEATURE PERMUTATION** of Figure 3.9 becomes Rule **SORT INTERSECTION** of Figure 3.6. In pratice, it should be more likely that this be the case for most pairs of sorts.

§ $\mathcal{OSF}$ GENERALIZATION MODULO FEATURE PERMUTATION

The same observation can be made for $\mathcal{OSF}$ generalization modulo feature permutation as shown by Rule **UNEQUAL TAGS MODULO FEATURE PERMUTATION** of Figure 3.10. Note that this rule identifies which feature to use for the left ψ-term's subterm depending on the index of the corresponding right ψ-term's feature given by π_{st}. This feature is identified as f_i, and so $\mathbf{index}(f_i) = i$ for $i = 1, \ldots, m$. But by (3.12), inverse-consistency of the feature map π_{st}, this means that $f_i = \pi_{ts}(g_k)$, for some $k \in \{1, \ldots, p\}$. From this, it comes that, for any index $k \in \{1, \ldots, p\}$, there is a unique index $i \in \{1, \ldots, m\}$ such that $i = \mathbf{index}(\pi_{ts}(g_k))$.

Note that this rule could equivalently be replaced by its symmetric form; namely, Rule **SYMMETRIC FUZZY UNEQUAL TAGS** of Figure 3.11.

[14] See Equation 3.5 for the definition of ψ-term dissolution into an $\mathcal{OSF}$ constraint.

UNEQUAL TAGS MODULO FEATURE PERMUTATION

$$\frac{\begin{bmatrix} X \neq Y; \\ m, n, p \geq 0 \text{ and } \{h_1, \ldots, h_p\} \stackrel{\text{def}}{=} \{\pi_{st}(f_1), \ldots, \pi_{st}(f_m)\} \cap \{g_1, \ldots, g_n\} \\ \text{s.t. } h_k \stackrel{\text{def}}{=} \pi_{st}(f_k) = g_k \text{ for all } k = 1, \ldots, p; \\ \psi'_k \stackrel{\text{def}}{=} \psi_{\mathbf{index}(\pi_{ts}(g_k))} \text{ for } k = 1, \ldots, p; \\ \gamma_i^0 \stackrel{\text{def}}{=} \gamma_1 \circ \{X/Z\} \text{ and } \gamma_2^0 \stackrel{\text{def}}{=} \gamma_2 \circ \{Y/Z\}, \text{ where } Z \text{ is a new variable} \end{bmatrix} \quad \begin{pmatrix} \gamma_1^0 \\ \gamma_2^0 \end{pmatrix} \vdash \begin{pmatrix} \psi'_1 \\ \xi_1 \end{pmatrix} \uparrow \begin{pmatrix} \gamma_1^0 \\ \gamma_2^0 \end{pmatrix} \chi_1 \begin{pmatrix} \gamma_1^1 \\ \gamma_2^1 \end{pmatrix} \quad \cdots \quad \begin{pmatrix} \gamma_1^{p-1} \\ \gamma_2^{p-1} \end{pmatrix} \vdash \begin{pmatrix} \psi'_p \\ \xi_p \end{pmatrix} \uparrow \begin{pmatrix} \gamma_1^{p-1} \\ \gamma_2^{p-1} \end{pmatrix} \chi_p \begin{pmatrix} \gamma_1^p \\ \gamma_2^p \end{pmatrix}}{\begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} \vdash \begin{pmatrix} X : s\,(f_i \rightarrow \psi_i)_{i=1}^m \\ Y : t\,(g_j \rightarrow \xi_j)_{j=1}^n \end{pmatrix} Z : s \vee t\,(h_k \rightarrow \chi_k)_{k=1}^p \begin{pmatrix} \gamma_1^p \\ \gamma_2^p \end{pmatrix}}$$

Figure 3.10: $\mathcal{OSF}$ generalization modulo feature permutation

SYMMETRIC UNEQUAL TAGS MODULO FEATURE PERMUTATION

$$\frac{\begin{bmatrix} X \neq Y; \\ m, n, p \geq 0 \text{ and } \{h_1, \ldots, h_p\} \stackrel{\text{def}}{=} \{f_1, \ldots, f_m\} \cap \{\pi_{ts}(g_1), \ldots, \pi_{ts}(g_n)\} \\ \text{s.t. } h_k \stackrel{\text{def}}{=} f_k = \pi_{ts}(g_k) \text{ for all } k = 1, \ldots, p; \\ \xi'_k \stackrel{\text{def}}{=} \xi_{\mathbf{index}(\pi_{st}(f_k))} \text{ for } k = 1, \ldots, p; \\ \gamma_i^0 \stackrel{\text{def}}{=} \gamma_1 \circ \{X/Z\} \text{ and } \gamma_2^0 \stackrel{\text{def}}{=} \gamma_2 \circ \{Y/Z\}, \text{ where } Z \text{ is a new variable} \end{bmatrix} \quad \begin{pmatrix} \gamma_1^0 \\ \gamma_2^0 \end{pmatrix} \vdash \begin{pmatrix} \psi_1 \\ \xi'_1 \end{pmatrix} \uparrow \begin{pmatrix} \gamma_1^0 \\ \gamma_2^0 \end{pmatrix} \chi_1 \begin{pmatrix} \gamma_1^1 \\ \gamma_2^1 \end{pmatrix} \quad \cdots \quad \begin{pmatrix} \gamma_1^{p-1} \\ \gamma_2^{p-1} \end{pmatrix} \vdash \begin{pmatrix} \psi_p \\ \xi'_p \end{pmatrix} \uparrow \begin{pmatrix} \gamma_1^{p-1} \\ \gamma_2^{p-1} \end{pmatrix} \chi_p \begin{pmatrix} \gamma_1^p \\ \gamma_2^p \end{pmatrix}}{\begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} \vdash \begin{pmatrix} X : s\,(f_i \rightarrow \psi_i)_{i=1}^m \\ Y : t\,(g_j \rightarrow \xi_j)_{j=1}^n \end{pmatrix} Z : s \vee t\,(h_k \rightarrow \chi_k)_{k=1}^p \begin{pmatrix} \gamma_1^p \\ \gamma_2^p \end{pmatrix}}$$

Figure 3.11: Equivalent symmetric $\mathcal{OSF}$ generalization modulo feature permutation

3.2 Fuzzifying $\mathcal{OSF}$ Subsumption

We are now ready to develop the same scheme of fuzzy declensions on $\mathcal{OSF}$ terms that we performed on the $\mathcal{FOT}$s in the previous chapter's Section 2.6, however dealing with (and taking advantage of) the more general lattice-theoretic semantics we have attributed to our $\mathcal{OSF}$ algebra and its derived operational calculus. Thus, in this section, we turn to fuzzifying lattice operations over $\mathcal{OSF}$ graphs. In so doing, since $\mathcal{OSF}$ graphs generalize $\mathcal{FOT}$s, we shall proceed as we did for $\mathcal{FOT}$s. Fuzzy lattice operations on $\mathcal{OSF}$ terms are in fact very close to what was presented in the previous chapter in Section 2.6.3 for lattice operations on $\mathcal{FOT}$s with partial argument maps. So let us start by making some important observations regarding the fuzzification of $\mathcal{FOT}$ unification and generalization.

1. The term structure itself (its syntax) is not fuzzified; only a conjunctive set E of equations (pairs of first-order terms—including substitutions) is given a global similarity degree α. This is denoted as the fuzzy-weighted set E_α.[15]

2. In axioms and rules, the similarity degree of a conjunctive set of equations can never increase from prior to posterior forms.

3. There is a similarity relation $\sim$ on functors f and and g (as a half-matrix of similarity degrees in $[0, 1]$).[16]

4. For each pair of functors f/m and g/n with $f \neq g$ and $0 \leq m \leq n$, whenever $f \sim_\alpha g$ with $\alpha \in (0, 1]$ there is a one-to-one mapping $p : \{1, \ldots, m\} \to \{1, \ldots, n\}$ associating each argument position of f to a unique distinct argument position of g; this mapping is the identity on $\{1, \ldots, m\}$ by default; it is undefined for dissimilar functors; in axioms and rules, when terms with similar functors with possible arity mismatch are equated, this argument-position mapping realigns misaligned subterms; subterms in the higher-arity term that are in excess are ignored.

Then, fuzzifying lattice operations for $\mathcal{FOT}$s consisted in adapting their crisp normalization rules to carry a similarity degree according to the above observations when transforming an equation set.

When considering $\mathcal{OSF}$ terms, we can proceed similarly, but instead of $\mathcal{FOT}$ unification, let us consider what this means for the ruleset of Figure 3.6, which enforces constraint consistency when subsumption is realized by endomorphic tag mappings. The latter are sets of variable/variable equations—*i.e.*, $X \doteq Y$—respecting sorts and feature application.[17] These rules operate taking into account the following observations.

1. The $\mathcal{OSF}$ terms themselves are not fuzzified; only a conjunctive set ϕ of atomic constraints (each of either of the forms $X : s$, $X.f \doteq Y$, and $X \doteq Y$) is given a global similarity degree α as the fuzzy formula ϕ_α.

[15] Where E_0 is equivalent to $\mathfrak{false}$ for any set of equations E—see Appendix Section A.3.1.
[16] Only one direction is needed; the other is equal by symmetry—see Appendix Section A.3.2.
[17] See Figure B.2 in Appendix B.1.

2. In axioms and rules, the similarity degree of a conjunctive set of atomic $\mathcal{OSF}$ constraints can never increase from prior to posterior forms.

3. The similarity relating pairs of sorts s and s' is a half-matrix of similarity degrees in $[0, 1]$. This similarity must be consistent with the ordering $\preceq$ on sorts; that is, for all sorts s, s', t, t' in $\mathcal{S}$, the following conditions holds for **lub**s and **glb**s when they exist:

$$\textbf{if } s \sim_\alpha s' \textbf{ and } t \sim_\beta t' \textbf{ then } (s \curlywedge t) \sim_{\alpha\wedge\beta} (s' \curlywedge t'), \tag{3.17}$$

$$\textbf{if } s \sim_\alpha s' \textbf{ and } t \sim_\beta t' \textbf{ then } (s \curlyvee t) \sim_{\alpha\wedge\beta} (s' \curlyvee t'); \tag{3.18}$$

Note that the similarity degree for both foregoing lattice operations on sorts uses fuzzy conjunction $(\wedge)$ of approximation degrees, as also pictured in Figure 3.12. For example,

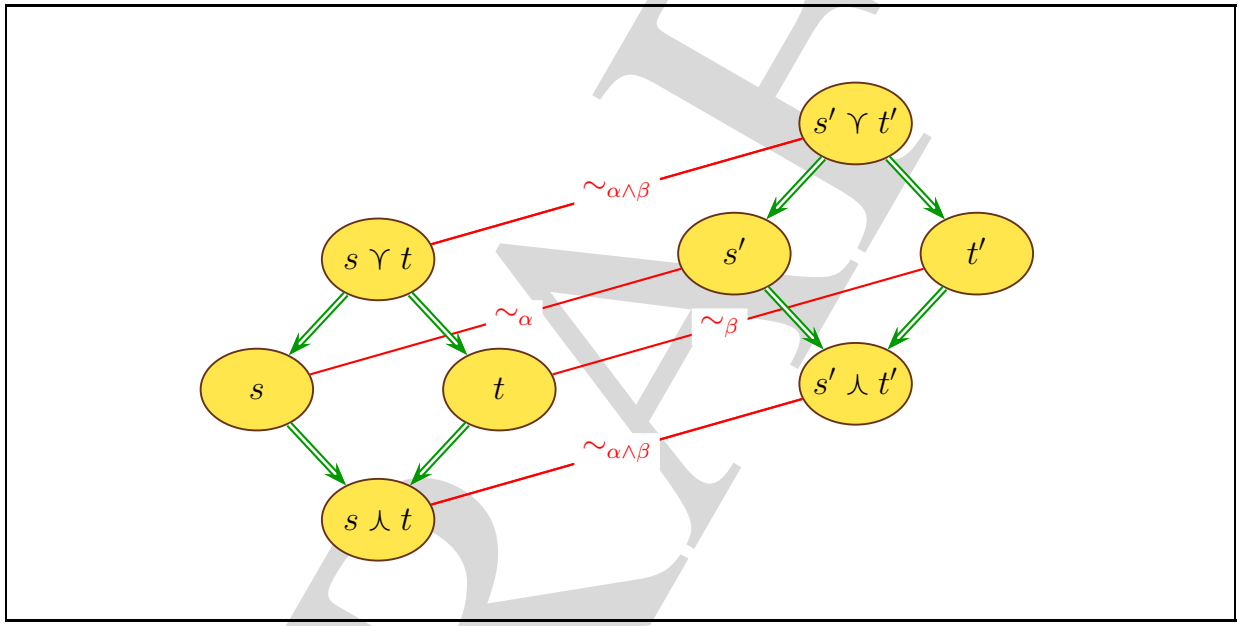


Figure 3.12: Order-consistent sort similarity

given a similarity on sorts such that:

$$\texttt{employee} \sim_{.9} \texttt{assistant}$$
$$\texttt{student} \sim_{.8} \texttt{apprentice}$$

and an ordering on sorts such that:

$$\texttt{helper} \stackrel{\text{def}}{=} \texttt{apprentice} \curlyvee \texttt{assistant}$$
$$\texttt{member} \stackrel{\text{def}}{=} \texttt{student} \curlyvee \texttt{employee}$$

and

$$\texttt{intern} \stackrel{\text{def}}{=} \texttt{apprentice} \curlywedge \texttt{assistant}$$
$$\texttt{working-student} \stackrel{\text{def}}{=} \texttt{student} \curlywedge \texttt{employee}$$

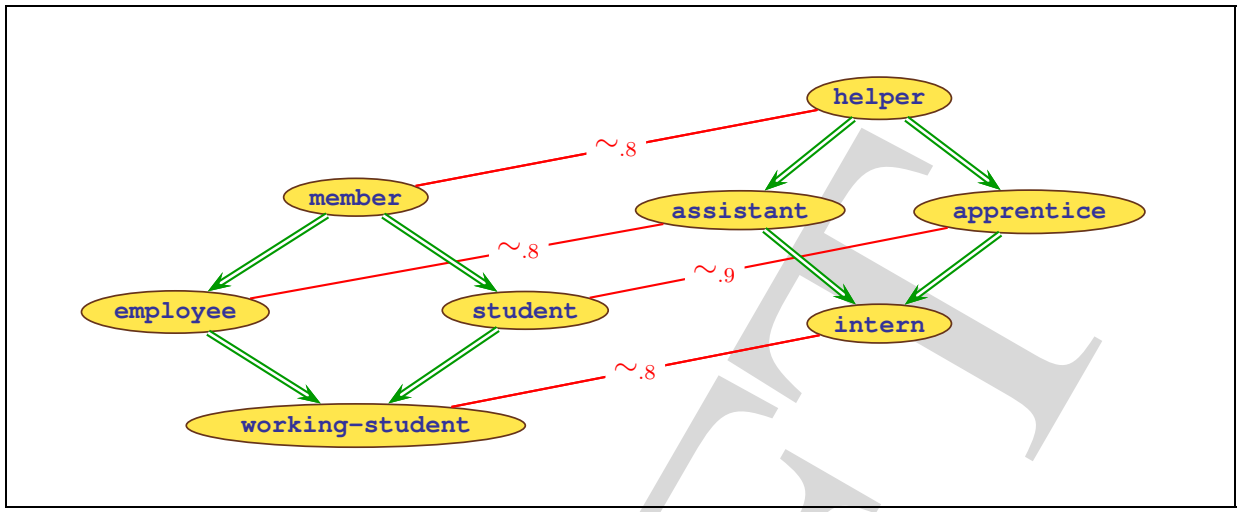


Figure 3.13: Order-consistent sort similarity example

then, necessarily for a consistent set of sorts, it must be that:

intern $\sim_{.8}$ **working-student**

and

member $\sim_{.8}$ **helper**

as illustrated in Figure 3.13.

4. The similarity relation $\sim$ is provided with a finitely non-identical feature permutation map $\pi_{st} : \mathcal{F} \to \mathcal{F}$ for each pair of sorts s and t for which $s \sim_\alpha t$ with $\alpha \in (0, 1]$. Thus, any consistent feature permutation can easily be realigned from the feature set of s to that of t with the assumed defined feature permutation π_{st}. This map is the identity on $\mathcal{F}$ by default.

   Authors' comment: *Note that thanks to its consistency properties, such a feature map π_{st} need only be specified as a finite half-matrix of bijective sets of pairs of non-identical features in $\mathcal{F} \times \mathcal{F}$.*[18] *Such a mapping is the identity by default.*

5. In axioms and rules, when a term of root sort s is compared to a term of root sort t, the one-to-one feature mapping π_{st} realigns any mismatched feature with its corresponding feature in its similar sort; this is illustrated as we shall see next in fuzzy $\mathcal{OSF}$ unification Rule **SIMILAR SORT INTERSECTION**, and in fuzzy $\mathcal{OSF}$ generalization Rule **FUZZY UNEQUAL TAGS**, where the foregoing properties of a feature map ensure that the rule is correct in both directions for similar pairs of sorts.

[18] That is, only one direction of finite permutations of the set $\mathcal{F} \times \mathcal{F} - \{\langle f, f\rangle \mid f \in \mathcal{F}\}$—one direction because it contains only one of either $\langle f, f'\rangle$ or $\langle f', f\rangle$, for all features f and f' such that $f \neq f'$. Such a matrix can easliy be generated by proper closure from a declared finite set of pairs of sorts with a finite feature mapping. See Chapter 4, Section 4.4.

SIMILAR SORT INTERSECTION

$$\frac{(\phi \;\&\; X : s \;\&\; X : t)_\alpha}{\left(\mathbf{fmap}^{\pi_{st}}_X(\phi) \;\&\; X : s \curlywedge t\right)_{\alpha\wedge\beta}} \quad \left[s \sim^{\pi_{st}}_\beta t,\ 0 \leq \beta \leq 1\right]$$

TAG ELIMINATION

$$\frac{(\phi \;\&\; X \doteq X')_\alpha}{(\phi[X'/X] \;\&\; X \doteq X')_\alpha} \quad [X' \in \mathbf{TagSet}(\phi)]$$

INCONSISTENT SORT

$$\frac{(\phi \;\&\; X \doteq \bot)_\alpha}{\mathfrak{false}_0}$$

FEATURE FUNCTIONALITY

$$\frac{(\phi \;\&\; X.f \doteq X' \;\&\; X.f \doteq X'')_\alpha}{(\phi \;\&\; X.f \doteq X' \;\&\; X' \doteq X'')_\alpha}$$

NULL SIMILARITY DEGREE

$$\frac{\phi_0}{\mathfrak{false}_0}$$

Figure 3.14: Constraint normalization rules for fuzzy $\mathcal{OSF}$ unification

3.2.1 Fuzzy $\mathcal{OSF}$ unification

With the above remarks, we may therefore proceed to fuzzifying the crisp $\mathcal{OSF}$ unification rules of Figure 3.6 into those of Figure 3.14.

In Rule **SIMILAR SORT INTERSECTION** of Figure 3.14, the function **fmap** is the same defined by Expression (3.15) in the crisp case to realign permuted features when comparing two sorts modulo a consitent feature permutation.

§ SIMILARITY OF ψ-TERMS

Let two ψ-terms ψ and ψ' defined as:

$$\psi \stackrel{\text{def}}{=} X : s(f_1 \to \psi_1, \ldots, f_n \to \psi_n)$$
$$\psi' \stackrel{\text{def}}{=} X' : s'(f'_1 \to \psi'_1, \ldots, f'_{n'} \to \psi'_{n'})$$

$(n, n' \geq 0)$.

DEFINITION 3.5 *For $\alpha \in [0,1]$, and two ψ-terms ψ and ψ' of the form above, we define recursively the fuzzy binary relation $\sim_\alpha$ on Ψ as $\psi \sim_\alpha \psi'$ iff $\alpha \stackrel{\text{def}}{=} \beta \wedge \bigwedge_{i=0}^{n} \beta_i$ where:*

$$s \sim^{\pi_{ss'}}_\beta s' \tag{3.19}$$

for some $\beta \in (0,1]$, and:

$$\psi_i \sim_{\beta_i} \mathbf{fmap}^{\pi_{ss'}}_X(\psi'_{i'}[X/X']) \tag{3.20}$$

where $\beta_i \in (0,1]$*, for any* $i \in \{1,\ldots,n\}$ *for which there a feature* $f_{i'}$*,* $i' \in \{1,\ldots,n'\}$*, such that* $f_{i'} = \pi_{ss'}(f_i)$ *for some feature* f_i*.*

THEOREM 3.3 (SIMILARITY OF ψ-TERMS) *The fuzzy binary relation* $\sim_\alpha$ *defined by Definition 3.5 is a similarity on the set of* ψ*-terms* Ψ*.*

PROOF **Reflexivity:** This follows because for any sort s, $s \sim_1^{\pi_{ss}} s$, since $\pi_{ss} = \mathbb{1}_{\mathcal{F}}$, by self-consistency of π (3.11), and because $\psi_i \sim_1 \mathbf{fmap}_X^{\pi_{ss}}(\psi_i[X/X])$ reduces to the tautology $\psi_i \sim_1 \psi_i$, for any f_i in $\mathcal{F}$ (since $f_i = \pi_{ss}(f_i)$), and any $X \in \mathcal{V}$, and thus also in particular for all $i = 1,\ldots,n$ for any $n \geq 0$.

Symmetry: by inverse-consistency (3.12) and order-consistency (3.14) of π_{st}.

Transitivity: by composition-consistency (3.13) and order-consistency (3.14) of π_{st}.

Details?
[To be completed. . .]

□

Because of Theorem 3.3, we shall say that ψ and ψ' are α*-similar* iff $\psi \sim_\alpha \psi'$.

THEOREM 3.4 (CORRECTNESS OF FUZZY $\mathcal{OSF}$ UNIFICATION) *Given a fuzzy* $\mathcal{OSF}$ *constraint* ϕ_α *with* $\alpha \in [0,1]$*, the process of non-deterministically applying to it any applicable rule shown in Figure 3.14 as long as one applies, always terminates in a fuzzy* $\mathcal{OSF}$ *constraint* $\phi'_{\alpha'}$ *such that either* $\phi' = \mathfrak{false}$ *and* $\alpha' = 0$*; or,* $0 < \alpha' \leq \alpha$ *and* $\phi \sim_{\alpha'} \phi'$*.*

PROOF Axiom **FUZZY EQUAL TAGS** is correct by reflexivity.

Correctness of Rule **FUZZY UNEQUAL TAGS** is established inductively. That is, we must prove that if we assume that all the prior fuzzy judgments of this rule are valid under all the rule's side conditions, then its posterior fuzzy judgment is valid.

[To be completed. . .] □

3.2.2 Fuzzy $\mathcal{OSF}$ generalization

Axiom **FUZZY EQUAL TAGS** in Figure 3.15 states that generalizing a pair made of the same ψ-term results in this ψ-term and the posterior tag maps and approximation degree are the same as the prior ones.

Note that, just like Rule **UNEQUAL TAGS MODULO FEATURE PERMUTATION** of Figure 3.10, Rule **FUZZY UNEQUAL TAGS** in Figure 3.15 identifies which feature f to use for the left ψ-term's subterm using its index $\mathbf{index}(f)$.

Also, Rule **FUZZY UNEQUAL TAGS** in Figure 3.15 uses a "fuzzy unapply" operation '$\uparrow_\alpha$' which takes a pair of ψ-terms with unequal root tags and an approximation degree α and returns

FUZZY EQUAL TAGS

$$\begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}_{\alpha} \vdash \begin{pmatrix}\psi\\ \psi\end{pmatrix} \psi \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}_{\alpha}$$

FUZZY UNEQUAL TAGS

$$\frac{\begin{array}{l} X \neq Y;\ s \sim_{\beta} t;\ \alpha_0 \stackrel{\text{def}}{=} \alpha \wedge \beta; \\ m, n, p \geq 0 \text{ and } \{h_1, \ldots, h_p\} \stackrel{\text{def}}{=} \{\pi_{st}(f_1), \ldots, \pi_{st}(f_m)\} \cap \{g_1, \ldots, g_n\} \\ \text{s.t. } h_k \stackrel{\text{def}}{=} \pi_{st}(f_k) = g_k \text{ for all } k = 1, \ldots, p; \\ \psi'_k \stackrel{\text{def}}{=} \psi_{\mathbf{index}(\pi_{ts}(g_k))} \text{ for } k = 1, \ldots, p; \\ \gamma_i^0 \stackrel{\text{def}}{=} \gamma_1 \circ \{X/Z\} \text{ and } \gamma_2^0 \stackrel{\text{def}}{=} \gamma_2 \circ \{Y/Z\}, \text{ where } Z \text{ is a new variable} \end{array}}{\begin{pmatrix}\gamma_1^0\\ \gamma_2^0\end{pmatrix}_{\alpha_0} \vdash \begin{pmatrix}\psi'_1\\ \xi_1\end{pmatrix} \uparrow_{\alpha_0} \begin{pmatrix}\gamma_1^0\\ \gamma_2^0\end{pmatrix} \chi_1 \begin{pmatrix}\gamma_1^1\\ \gamma_2^1\end{pmatrix}_{\alpha_1} \cdots \begin{pmatrix}\gamma_1^{p-1}\\ \gamma_2^{p-1}\end{pmatrix}_{\alpha_{p-1}} \vdash \begin{pmatrix}\psi'_p\\ \xi_p\end{pmatrix} \uparrow_{\alpha_{p-1}} \begin{pmatrix}\gamma_1^{p-1}\\ \gamma_2^{p-1}\end{pmatrix} \chi_p \begin{pmatrix}\gamma_1^{p}\\ \gamma_2^{p}\end{pmatrix}_{\alpha_p}}{\begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}_{\alpha} \vdash \begin{pmatrix}X : s\,(f_i \rightarrow \psi_i)_{i=1}^{m} \\ Y : t\,(g_j \rightarrow \xi_j)_{j=1}^{n}\end{pmatrix} Z : s \vee t\,(h_k \rightarrow \chi_k)_{k=1}^{p} \begin{pmatrix}\gamma_1^{p}\\ \gamma_2^{p}\end{pmatrix}_{\alpha_p}}$$

Figure 3.15: Fuzzy $\mathcal{OSF}$ generalization axiom and rule

a pair of (possibly identical) ψ-terms and a possibly lesser approximation degree. It is defined as follows:

$$\begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \uparrow_{\alpha} \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix} \stackrel{\text{def}}{=} \begin{cases} \begin{pmatrix}X : \ldots\\ X : \ldots\end{pmatrix}_{\alpha \wedge \alpha_1 \wedge \alpha_2} & \text{if } \exists X \text{ s.t. } \psi_i \sim_{\alpha_i} \psi'_i \text{ where:} \\ & \mathbf{ROOT}(\psi'_i) = \gamma_i(X) \text{ for } i = 1, 2; \\ \begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix}_{\alpha} & \text{otherwise.} \end{cases} \tag{3.21}$$

This has the same purpose as the fuzzy unapplication operation used in fuzzy $\mathcal{FOT}$ generalization judgments: identify in the prior pair of tag maps (γ_1, γ_2) whether or not they already map a common variable (X) to the roots of the pair of ψ-terms to be generalized (ψ_1, ψ_2) each at, respectively, approximation α_i, $i = 1, 2$. If so, the result of the unapplication is the pair made of the same ψ-term rooted in X $(X : \ldots)$ at a posterior approximation degree equal to the conjoined value of the prior approximation degree α and those; *i.e.*, $\alpha \wedge \alpha_1 \wedge \alpha_2$; if not, it is the original pair of ψ-terms (ψ_1, ψ_2) at the unchanged prior approximation degree.

Rule **FUZZY UNEQUAL TAGS** of Figure 3.15 states that generalizing two ψ-terms $\psi_1 \stackrel{\text{def}}{=} X : s\,(f_i \rightarrow \psi_i)_{i=0}^{m}$ and $\psi_2 \stackrel{\text{def}}{=} Y : t\,(g_j \rightarrow \xi_j)_{j=0}^{n}$ results in the ψ-term $\psi_1 \vee \psi_2 \stackrel{\text{def}}{=} Z : s \vee t\,(h_k \rightarrow \chi_k)_{k=0}^{p}$, where the set of features of the resulting ψ-term is the intersection of the realigned sets of features of ψ_1 and ψ_2 (*i.e.*, the realigned features they have in common), and Z is a new variable,

As was the case for $\mathcal{FOT}$s, note that fuzzy $\mathcal{OSF}$ unapplication defined by Equation (3.21) returns a pair of terms and a (possibly lesser) approximation degree, unlike crisp unapplication defined by Equation (3.6) which returns only a pair of terms. Because of this, when we write a

fuzzy $\mathcal{OSF}$ generalization judgment such as:

$$\begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}_{\alpha} \vdash \begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \uparrow_{\alpha} \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix} \psi \begin{pmatrix}\gamma'_1\\ \gamma'_2\end{pmatrix}_{\beta} \tag{3.22}$$

as we do in Rule **FUZZY UNEQUAL TAGS**, this is shorthand to indicate that the posterior similarity degree β is *at most* the one returned by the fuzzy $\mathcal{OSF}$ unapplication $\begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \uparrow_{\alpha} \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}$. Formally, the notation of the fuzzy $\mathcal{OSF}$ generalization judgment (3.22) is equivalent to:

$$\begin{pmatrix}\psi'_1\\ \psi'_2\end{pmatrix}_{\beta'} \stackrel{\text{def}}{=} \begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \uparrow_{\alpha} \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix} \quad \text{and} \quad \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}_{\beta'} \vdash \begin{pmatrix}\psi'_1\\ \psi'_2\end{pmatrix} \psi \begin{pmatrix}\gamma'_1\\ \gamma'_2\end{pmatrix}_{\beta} \tag{3.23}$$

for some β' such that $\beta \leq \beta' \leq \alpha$. This is because a fuzzy $\mathcal{OSF}$ term unapplication invoked while proving the validity of a fuzzy $\mathcal{OSF}$ generalization judgment may require, by Expression (3.21), lowering the *prior* approximation degree of the judgment. This is therefore applicable to common-feature subterms corresponds to generalizing each of the corresponding pairs of subterms under all common features. Note that this can be done in any order, as long as each subterm judgment is validated with its pair of prior tag maps equal to its pair of posterior tag maps.

As for the crisp case, note that Rule **FUZZY UNEQUAL TAGS** of Figure 3.15 could equivalently be replaced by its symmetric form; namely, Rule **SYMMETRIC FUZZY UNEQUAL TAGS** of Figure 3.16.

SYMMETRIC FUZZY UNEQUAL TAGS

$$\frac{\begin{array}{l}X \neq Y;\ s \sim_{\beta} t;\ \alpha_0 \stackrel{\text{def}}{=} \alpha \wedge \beta;\\ m, n, p \geq 0 \text{ and } \{h_1, \ldots, h_p\} \stackrel{\text{def}}{=} \{f_1, \ldots, f_m\} \cap \{\pi_{ts}(g_1), \ldots, \pi_{ts}(g_n)\}\\ \text{s.t. } h_k \stackrel{\text{def}}{=} f_k = \pi_{ts}(g_k) \text{ for all } k = 1, \ldots, p;\\ \xi'_k \stackrel{\text{def}}{=} \xi_{\mathbf{index}(\pi_{st}(f_k))} \text{ for } k = 1, \ldots, p;\\ \gamma^0_i \stackrel{\text{def}}{=} \gamma_1 \circ \{X/Z\} \text{ and } \gamma^0_2 \stackrel{\text{def}}{=} \gamma_2 \circ \{Y/Z\}, \text{ where } Z \text{ is a new variable}\end{array}\quad \begin{pmatrix}\gamma^0_1\\ \gamma^0_2\end{pmatrix}_{\alpha_0} \vdash \begin{pmatrix}\psi_1\\ \xi'_1\end{pmatrix} \uparrow_{\alpha_0} \begin{pmatrix}\gamma^0_1\\ \gamma^0_2\end{pmatrix} \chi_1 \begin{pmatrix}\gamma^1_1\\ \gamma^1_2\end{pmatrix}_{\alpha_1} \cdots \begin{pmatrix}\gamma^{p-1}_1\\ \gamma^{p-1}_2\end{pmatrix}_{\alpha_{p-1}} \vdash \begin{pmatrix}\psi_p\\ \xi'_p\end{pmatrix} \uparrow_{\alpha_{p-1}} \begin{pmatrix}\gamma^{p-1}_1\\ \gamma^{p-1}_2\end{pmatrix} \chi_p \begin{pmatrix}\gamma^p_1\\ \gamma^p_2\end{pmatrix}_{\alpha_p}}{\begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}_{\alpha} \vdash \begin{pmatrix}X : s\,(f_i \rightarrow \psi_i)_{i=1}^{m}\\ Y : t\,(g_j \rightarrow \xi_j)_{j=1}^{n}\end{pmatrix} Z : s \vee t\,(h_k \rightarrow \chi_k)_{k=1}^{p} \begin{pmatrix}\gamma^p_1\\ \gamma^p_2\end{pmatrix}_{\alpha_p}}$$

Figure 3.16: Equivalent symmetric fuzzy $\mathcal{OSF}$ generalization rule

Chapter 4

Version of January 8, 2019

Discussion

- Section 4.1 reviews other fuzzy unification work.
- Section 4.2 reviews related work:
  - Subsection 4.2.1 reviews graph-similarity measures;
  - Subsection 4.2.2 indicates some potential ties: we look at some known fuzzy data models (such as fuzzy object-oriented) and subtyping; we discuss models of fuzzy knowledge; we make a link with fuzzy automata for fuzzy string matching; we relate our work to Fuzzy Quantum Logic;
- Section 4.3 reviews a few systems implementing some notion of fuzzy unification.

4.1 Other Fuzzy Unification Work

In the course of this work, we searched the literature for "*fuzzy unification*" and "*fuzzy logic programming*," and variations thereof. Our first observation in pursuing this interest has been that, unlike existing Prolog languages (and other technology that relies on standard $\mathcal{FOT}$ unification), not all the fuzzy $\mathcal{LP}$ languages that have been proposed and/or implemented (even if only as prototypes) agree on the same fuzzy $\mathcal{FOT}$ unification operation. We have looked at some of the most prominent among those existing in an attempt to characterize their fuzzy unification operations. We next summarize some essential points that we understood of these variations on fuzzy unification from our perspective. We will proceed succintly, as it would be presumptuous of us to give an exhaustive recap of research in Fuzzy $\mathcal{LP}$. Again, our perspective is not so much Fuzzy $\mathcal{LP}$ as it is that of understanding its fuzzification of the lattice-theoretic operations on $\mathcal{FOT}$s such as $\mathcal{FOT}$ unification, which is an essential part on any $\mathcal{LP}$ system.

Before we focused on M. Sessa's fuzzy $\mathcal{FOT}$ unification algorithm, which we present in Chapter 2,[1] we looked at other work which we recall here. We also discuss our reasons for our choice to follow Sessa's approach as opposed to fuzzy Datalog or edit-distance $\mathcal{FOT}$ matching we describe next.

[1] Section 2.6.1.

§ Fuzzy Datalog Unification

Among earlier frequently cited works related to fuzzy $\mathcal{FOT}$ unification is [30], where the title leads one to expect a fuzzy term unification in a fuzzy $\mathcal{LP}$ language. It is not quite that, however, as it restricts solving similarity equations over word symbols tolerating some imprecise matches (*i.e.*, a kind of fuzzy Datalog).[2] Such mismatches could come from (possibly accidental) syntactic proximity (*e.g.*, typos, misspellings).[3] The authors propose to accumulate mismatched symbols into what they dub "*clouds*," which represent in effect similarity classes of constant symbols (nullary functors). These clouds are given a measure of "similarity" computed as the meet of those of the components—which they propose to conceive as a "cost" of how much it deviates from a perfect match (which itself has zero cost, since perfect).

Note that when the only non-variable terms defined are constants, the rules of Figure 2.3 accumulate all constant mismatches as unresolved equations. Thus, what constitutes Arcelli *et al.*'s "clouds" are the sets of constants making up the equivalence classes of the reflexive-transitive closure of the relation containing these equations.

While this could be useful as a particular fuzzy extension of Datalog, it does not address issues concerning fuzzy database representation and evaluation issues such as expounded in [48] for (crisp) Datalog. In fact, such fuzzy extensions of Datalog had already been proposed, with a straightforward fix-point semantics extending that of classical Datalog (*e.g.*, [1]). Since it limits itself to fuzzification of a Datalog-like $\mathcal{LP}$, the semantics of Arcelli *et al.*'s fuzzy $\mathcal{LP}$ language only considers approximate equations between constant symbols, whose mutual fuzzy proximity is specified as fuzzy "proximity matrices". This early form of fuzzy unification was put to use in the fuzzy $\mathcal{LP}$ language Likelog [28], [29], [31], [32].

§ Edit-Distance Fuzzy Unification

Arcelli *et al.*'s fuzzy constant unification was later elaborated to work on full $\mathcal{FOT}$s as first exposed in [30] and used in [61], and then again in [115] and [116]. The authors use the same kind of fuzzy unification on constants (*i.e.*, names formalized as symbol strings), but instead of names deemed similar (*i.e.*, in a same "cloud" or similarity class), the more classical notion of *edit distance* is used to evaluate the similarity degree of a fuzzy name match (normalized over the symbol lengths). Edit distance between two strings is the minimal number of elementary edit actions (deleting a character, inserting a character, or replacing a character for another), in either or both strings necessary to obtain a perfect match [87].[4] This fuzzy unification was put to use in biological genetics analysis with the fuzzy $\mathcal{LP}$ language FURY [61], [116].

In what follows, we use our own formal notation to summarize the essence of FURY-style fuzzy $\mathcal{FOT}$ unification [61], [115], [116]. We represent the empty string as ε, and a non-empty string as a dot-separated sequence of characters ending with the empty string (*e.g.*, `"this"` is represented as `t.h.i.s.`ε). Given a non-empty string $h.t$, we shall call its first character h its "*head*" and the substring t following it, its "*tail*." The length of a string s is its number of

[2] It is only acknowledged in the very last sentence of the paper, that the authors were yet "*aiming at extending this algorithm to the full-fledged algebra of first-order terms*" as future work.

[3] Or presumably, although they only mention it as a potential further work in their conclusion, from semantic proximity (such as could be specified as fuzzy knowledge in the form of fuzzy similarity matrices).

[4] It is used most crucially in Internet search keyword matches and DNA sequence alignment and matching.

characters denoted $|s|$. That is, the monoid homomorphism:

$$|\varepsilon| \stackrel{\text{def}}{=} 0$$
$$|h.t| \stackrel{\text{def}}{=} 1 + |t|.$$

Thus, the edit distance $\delta(s_1, s_2)$ between two strings s_1 and s_2 is derived as:[5]

$$\left.\begin{array}{rcl} \delta(\varepsilon, s) & \stackrel{\text{def}}{=} & |s| \\ \delta(s, \varepsilon) & \stackrel{\text{def}}{=} & |s| \\ \delta(h.t_1, h.t_2) & \stackrel{\text{def}}{=} & \delta(t_1, t_2) \\ \delta(h_1.t_1, h_2.t_2) & \stackrel{\text{def}}{=} & 1 + \textbf{min}\{\, \delta(t_1, h_2.t_2), \delta(h_1.t_1, t_2), \delta(t_1, t_2)\,\}, \quad \text{if } h_1 \neq h_2. \end{array}\right\} \tag{4.1}$$

These four defining equations express that the edit distance: (1) from any string to the empty string is the length of this string; (2) between two strings with equal first character, it is the distance between the remaining substrings; (3) otherwise, it is one plus the minimum of the three edit distances between one of the strings and the other string's rest, and between the two strings' rests. The latter is known as the "*Levenshtein distance*" between two strings.[6]

Because edit distance will increase with the lengths of strings, it is convenient to calibrate it over the size of the strings involved; hence the notion of "*normalized edit distance*" δ_N as in:

$$\delta_N(s_1, s_2) \stackrel{\text{def}}{=} \frac{\delta(s_1, s_2)}{\textbf{max}(|s_1|, |s_2|)}. \tag{4.2}$$

In [61], this notion of (normalized) edit distance between constant symbol strings (including the empty string ε) is extended to an edit distance between $\mathcal{FOT}$ trees. It maps two terms t_1 and t_2 to a non-negative number $\delta(t_1, t_2) \stackrel{\text{def}}{=} m^{\sigma}_{n} \in \mathbb{N}$, which denotes the minimal total number m of mismatches (edit actions necessary to go from one to the other), along with two collateral pieces of information:

- σ—a most general variable substitution that may be necessary to resolve matches upon encountering variables in the process of computing this minimal edit distance between the two terms when assimilated to strings which are sequences of the non-punctuation characters that compose their their syntax; and,
- n—a normalization factor computed as the sum of the lengths of the longest of each of pair of symbols from each term as they are matched.

[5] From which one can easily check that expected properties of a distance are verified by δ such as $\delta(s, s) = 0$, $\delta(s_1, s_2) = \delta(s_2, s_1)$, and $\delta(s_1, s_2) \leq \delta(s_1, s) + \delta(s, s_2)$, for any strings s, s_1, and s_2.

[6] The Levenshtein distance between two strings has the advantage to apply to strings of differring as well as of equal lengths. This is unlike the *Hamming distance* which is restricted to strings of equal lengths, and defined as the number of disagreeing character positions. This entails, in particular, that the Levenshtein distance between two equal-length strings is always less than or equal to their Hamming distance.

The normalization factor n is a function of the lengths of the terms when a term is seen as the string concatenation in the order they appear of the non-variable and non-punctuation symbols in it. In other words, parentheses, commas, variables, and ε are considered of length 0. Namely,

$$|t| \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } t \text{ is a variable} \\ |f| + \Sigma_{i=1}^{n} |t_i| & \text{if } t = f(t_1, \ldots, t_n),\ n \geq 0. \end{cases}$$

Compounding two $\mathcal{FOT}$ edit distances m_n^σ and p_q^θ consists in adding them up while composing their substitutions and adding their normalization factors:

$$m_n^\sigma + p_q^\theta \stackrel{\text{def}}{=} (m+p)_{n+q}^{\theta\sigma}. \tag{4.3}$$

In other words, as it sums the numbers of symbol mismatches, it also composes their associated variable substitutions and sums their normalization factors (which depend on the sizes of all the involved symbols). Note that this operation, while commutative in its numerical arguments (which are added), is not commutative in its substitution arguments (which are composed). It could also be defined by composing the substitutions in the other order if wished; but this is simpler.[7]

Given two $\mathcal{FOT}$s s and t, the edit distance between them $\delta(s,t)$ is defined as follows. If the first argument is the empty string, then:

$$\delta(\varepsilon, t) \stackrel{\text{def}}{=} |t|_{|t|}^{\emptyset} \tag{4.4}$$

meaning that the edit distance is the length of the second argument, which is also this distance's maximum known normalization factor. If the first argument is a variable, then:[8]

$$\delta(X, t) \stackrel{\text{def}}{=} 0_0^{\{t/X\}} \tag{4.5}$$

meaning that it is zero, while binding its first argument to its second argument, with a zero normalization factor. If the second argument is a variable, then:

$$\delta(t, X) \stackrel{\text{def}}{=} \delta(X, t) \tag{4.6}$$

by symmetry, which then uses the previous case. Otherwise (neither argument is a variable), let $s = g(s_1, \ldots, s_m)$ and $t = f(t_1, \ldots, t_n)$ for some $m, n \geq 0$; then:

$$\left.\begin{aligned} \delta(s,t) \stackrel{\text{def}}{=}\ & \delta(f,g)_{\max(|f|,|g|)}^{\emptyset} \\ & + \min \{\ \delta(\varepsilon, s_1) + \delta(\varepsilon(s_2, \ldots, s_m),\ \varepsilon(t_1, \ldots, t_n)) \\ & \qquad ,\ \delta(\varepsilon, t_1) + \delta(\varepsilon(s_1, \ldots, s_m),\ \varepsilon(t_2, \ldots, t_n)) \\ & \qquad ,\ \delta(s_1, t_1) + \delta(\varepsilon(s_2, \ldots, s_m)\sigma, \varepsilon(t_2, \ldots, t_n)\sigma)\ \} \\ & \text{if } \delta(s_1, t_1) = p_q^\sigma \text{ for some } p, q \geq 0 \text{ and substitution } \sigma \end{aligned}\right\} \tag{4.7}$$

[7] The authors of [61] compose their substitutions the other way, which is why they need to write their recursive 'et' rule (the last one) with the first subterms as second arguments when collecting those of the subterms.

[8] Recall that we use Prolog's convention of writing variables with capitalized symbols.

which defines a Levenshtein distance extended from strings to terms. It is equal to the edit distance between the functors plus the minimum of the three possible ways of aligning the respective sequences of subterms, composing substitutions and adding normalization factors, each set to the maximum functor length, while incrementally instantiating subterms remaining in the tails with the accumulated substitutions resulting from computing the heads' edit distance at each recursive calls.

> ***Authors' comment:*** *The above rules given as Equations (4.4)–(4.7), with our own—and simpler—notation, are adapted from Definition 5 of the Gilbert-Schroeder paper [61]. However, while they agree on the first three rules, they do not on the last one. On that last one, they only agree on the first two cases of the three recursive patterns, they differ on the last: our own rule—Equation (4.7)—propagates to the rest of the arguments the substitution resulting from computing the edit distances between the first arguments of both terms. The two other cases need not do so as either term's first argument is only paired with the empty string ε, which simply returns the identity substitution $\emptyset$—by Equation (4.4). Indeed, not propagating like in their definition of the term edit distance 'et' (Definition 5) is incorrect. Take for instance the two terms $f(a,b)$ and $g(X,X)$. According to that definition, their term edit distance is $1_3^{\{a/X\}}$. However, taking that substitution into account, it should be $2_3^{\{a/X\}}$ (since there are 2 mismatches between $f(a,b)$ and either $g(a,a)$ or $g(b,b)$: $f \neq g$ and $a \neq b$). Indeed, the definition given in [61] means that the two occurrences of X are seen as two independent variables which then get independently bound (one time to a and the other time to b), then composing the substitutions will keep only the first one ($\{a/X\}$) and not account for the argument mismatch $a \neq b$. Whereas, propagating the substitution as done in Equation (4.7) makes it possible to account for the mismatch (since a will be have been substituted for X), therefore correctly returning $2_3^{\{a/X\}}$.*
>
> *It could have been a typo or misprint in Definition 5 in [61], perhaps. But in other later papers using this unification (such as [115] and [116]), the same definition is again given. At any rate, to propagate substitutions from one matching argument to matching the rest is a simple option. Not doing it, although not optimal, does not invalidate their approach when the substitution propagation is done correctly (as done in Equation (4.7)); it just catches less mismatches in general than ought to be reported (since it has for effect to ignore potential mismatches that may come from any multiple-occurrence variable which are already bound to different symbols).*

In this manner, this accounts for the fact it may be necessary to perform variable substitutions while establishing the normalized edit distance between two terms t_1 and t_2 such that the following fuzzy equation holds whenever $\delta(t_1,t_2) = m_n^\sigma$:

$$t_1\sigma \sim t_2\sigma \left[\tfrac{n-m}{n}\right]. \tag{4.8}$$

Indeed, having m character mismatches over a maximum total length of n characters means that the rate of mismatch is $\frac{m}{n} \in [0,1]$; or, equivalently, that the rate of correctly matched characters is $1-\frac{m}{n}$; *i.e.*, $\frac{n-m}{n}$. It can then be used as a similarity degree fuzzifying the equation $t_1\sigma = t_2\sigma$. Indeed, if all the characters are mismatched, then $m = n$ and therefore the similarity degree of this equation is zero; whereas, if no characters are mismatched, then $m = 0$, and the truth value is one. As expected, the higher the number of mismatches, the fuzzier the solution.

This is a very interesting trick: "*stringifying*" the syntax of a $\mathcal{FOT}$ and then using fuzzy symbol matching while counting how many mismatches over how long symbols and substituting terms for variables as needed to resolve discrepancies in term structure. Thus, calculating the normalized edit distance between two terms with Equations (4.4)–(4.7) operates an implicit unification procedure (which we shall call Gilbert-Schroeder fuzzy unification). It has, in fact, the same recursive pattern as Robinson's procedural unification algorithm, relaxed to tolerate functor and arity inequalities [105].[9]

There are some important observations to be made at this point regarding Gilbert-Schroeder fuzzy $\mathcal{FOT}$ unification.

- It applies to conventional (crisp) Prolog terms: there is no need for "fuzzy $\mathcal{FOT}$s" whatever such may be (it is the unification that is fuzzy, not the terms).
- It is a purely lexical process: it relates strings as character sequences regardless of word meaning and/or context.
- It can always derive a minimal edit distance between two terms, however unrelated they may be—the more lexically unrelated, the larger this distance will be, although it will always be finite as it is bounded by a function of the size of the terms,[10] as well as the lengths of the functor symbols in them and the number of variable re-occurrences at leaves. Normalizing with respect to the length of concatenation of the longest of each pairs of symbols appearing in corresponding subterms gives a bounded measure in $[0, 1]$ of the character mismatch rate, therefrom a fuzzy matching measure may be derived.
- When fuzzy-unifying two non-variable non-ε terms, their arities (number of subterms) may differ as each subterm of one is unified with each subterm of the other, keeping only the minimal total number of mismatches (and collateral substitution and normalization factor)—which raises efficiency concerns. Such concerns have been addressed for tree edit distances in more recent works such as, *e.g.*, [54], although not for $\mathcal{FOT}$s which are rooted directed acyclic graphs (variables are shared nodes). Although the number of arguments of two fuzzy matching terms may differ, it must be noted that in computing edit-distance between two $\mathcal{FOT}$s, the order of argument-position is always preserved.
- It is not difficult to understand from Equations (4.1) and (4.7), that the complexity of a *naïve* implementation of this recursive scheme becomes quickly prohibitive for pragmatics. Thus, optimization methods, implementation techniques such as Dynamic Programming including specific-domain heuristics, have been the center of attention [39], [125], [124].
- More worrisome is that computing this term edit distance is not only expensive, it is also *non-deterministic*. Indeed, there may be equal minimal number of mismatches with different and incomparable variable substitutions. For example, the minimal term edit distance between `f(X,X,a)` and `g(b,Y,Y)` is 2 with either substitutions $\{\texttt{b}/\texttt{X}, \texttt{b}/\texttt{Y}\}$ or $\{\texttt{a}/\texttt{X}, \texttt{a}/\texttt{Y}\}$, which are both most general although mutually incomparable (*i.e.*, they are not alphabetical variants "up to variable renaming" of one another).

[9] *Op. cit.*, Section 5.8, Page 32.
[10] The number of functor nodes.

4.2 Related work

4.2.1 Feature Graph Similarity Measures

Work using our $\mathcal{OSF}$ formalism has also elaborated a general lattice-theoretic approach to measuring similarity over $\mathcal{OSF}$ graphs [96]. We now review this work and discuss how our approach and theirs are in fact quite compatible, the latter providing a way to derive from the structure of $\mathcal{OSF}$ graphs a similarity distance which can be used as the fuzzy information presumed available by the former.

4.2.2 Potential ties

Fuzzy data models

Fuzzy object-oriented data model [42].

Fuzzy subtyping: [44].

Fuzzy knowledge

Fuzzification of Description Logic (DL): earlier attempts such as [129]; more recently: [118, 119]. This fuzzifies DL by attaching a similarity degree to DL assertions and interpreting constraints with fuzzy connectives: infimum ($\wedge$) is **min**, supremum ($\vee$) is **max**, and complement ($\lambda\phi.\overline{\phi}$) is $\lambda w.(1-w)$. Specifying minimal and/or maximal values is used to disregard all assertions and constraints with similarity degree outside a specified interval. In this regard (setting minimal/maximal bounds), the latter is similar to such fuzzy logics as [26].

Fuzzy automata

It is easy to see that, by abstracting the algebraic operations is uses, Dijsktra's shortest path algorithm[11] is only one instance of a larger family of algorithms in an inf/sup lattice known as Warshall's algorithm working on any such algebraic structures [126].[12] This is of great benefit for software development: it is sufficient to encode only one algorithm with abstract `inf` and `sup` operations and vary the effect according any specific instantiations of these operations. Many closing algorithms in dual algebras such as (semi-) rings, (semi-) lattices, *etc.*, are of this type (see, *e.g.*, Algorithm 1).

Formalization of fuzzy NFAs:

- [46]
- [88]

General references and links on fuzzy regular expressions (to connect with $\mathcal{OSF}$ unification extended to regular path expressions [35]):

[11] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

[12] http://www.cse.chalmers.se/~coquand/AUTOMATA/over7.pdf

- Fuzzy Regular Expressions
- Fuzzy Regular Expression Matching software

Approximate string matching: [80].
More readings on fuzzy automata:

- fuzzy automata [33]
- fuzzy and non-deterministic automata [95]
- Myhill-Nerode theory for fuzzy languages and automata [71]
- Lattice-ordered monoids and automata [120, 121]

Fuzzy Quantum Logic

Also to read the intriguing interesting connections of Fuzzy Logic with Quantum Logic. To cite just a few: [41],[13] [102], [103], [53], [113]. [65]. See also: this and this. In [65], a nice algebraic summary is given as:

Logic Property	Classical Logic	Birkhoff – Von Neumann Logic	Zadeh Fuzzy Logic	Giles – Łucasiewicz Fuzzy Logic
Binary	yes	yes	no	no
Commutative	yes	no	yes	yes
Distributive	yes	no	yes	no
Excluded Middle	yes	yes	no	yes
Non-Contradiction	yes	yes	no	yes

Attributed conceptual information is the space of observations (an infinite-dimensional (quasi-Hilbert) space), and the phase space comprises functions of probable neighbor states of points in the observation space according to the axioms of Quantum Logic $\mathcal{QL}$, which is a weakening of $\mathcal{FL}$.

4.2.3 Other links

Dynamic programming

Bellman's Dynamic Programming [39] (since related to Dijkstra's shortest path algorithm).

Fuzzy FCA

See essentially Radim Bělohlávek's work and references in there.

[13] This is generally credited to be the pioneering paper.

Fuzzy GDL

Fuzzifying the Generalized Distributive Law (GDL) [22, 23] (*i.e.*, lifting it to any fuzzy lattice structure).

> ***Authors' comment:*** *Need to* ***read*** *and comment the state of the art, and infuse whether this can connect productiveley (in either directions) with $\mathcal{OSF}$ graphs seen as order-sorted FSAs (or NFAs) on regular languages of feature composition words (see, e.g., [35]).*

4.3 Fuzzy Implementations

A few fuzzy $\mathcal{LP}$ systems have been proposed and implemented using some of the fuzzy unification operations defined by the state of the art that we overviewed in Section 2.6.1, or variants thereof. The following are just some among the many one can look up, some of which may be downloadeded and used.

- FRIL [37], [36]: a LISP-based fuzzy $\mathcal{LP}$ from U. of Bristol, UK, (*ca.*, 1992), and its more recent object-oriented extension FRIL++ [106, 45, 107].
- Likelog [28], [29], [31], [32]; all using Arcelli *et al.*'s "cloud" fuzzy unification.
- FURY (using Gilbert-Schroeder fuzzy $\mathcal{FOT}$ unification) was designed and experimented with for applications in Biology [61], [116].
- Bousi∼Prolog[14] (using Sessa's "weak resolution" based on "weak unification") [117].

  Papers on Bousi∼Prolog:

  - the main system [76];
  - implementation of Sessa's "weak unification" [74];
  - in [73] a straightforward adaptation to handle "weak unification" and "weak resolution" in a fuzzy $\mathcal{LP}$ system of Prolog's WAM implementation [6]; (*viz.*, the "SWAM").

    > ***Authors' comment:*** *But this implementation uses cubic-time Warshall-Strassen transitive-closure of a fuzzy relation [72]. Couldn't this be computed faster like what Algorithm 1 does in linear time?*

- FASILL [77]: also uses Sessa's (and Bousi∼Prolog's) "weak unification" and "weak resolution," but the language has some added features. the most important being that of offering several kinds of fuzzy operators,[15]

[14] http://dectau.uclm.es/bousi/

[15] Such as those in Appendix Section A.3, Equation (A.15) and the possibility to use them simultaneously (each being a pair of a fuzzy implication and conjunction $\langle \rightarrow_i, \&_i \rangle$ for the i^{th} kind of fuzzy operators), based on the semantics of "*multi-adjoint $\mathcal{LP}$*" (MALT; see also this).

Authors' comment: *This is* not *technically speaking related to Bousi∼Prolog, although they have a common designer/author—P. Julian-Iránzo, of Universidad de Castilla, La Mancha— and belong to the "Spanish Fuzzy $\mathcal{LP}$ School" where the Bousi∼Prolog SWAM implementer—Clemente Rubio Manzano—did his PhD under Julian-Iránzo's supervision and is now at Universidad del Bio-Bio.*

- FLINT ("*Fuzzy Logic INferencing Toolkit*"): a fuzzy $\mathcal{LP}$ toolkit in LPA Prolog. What fuzzy unification algorithm does it use? None, in fact: it is a meta-interpreter implemented as an LPA Prolog library that processes Horn clause rules over fuzzy sets represented as "*fuzzy variables*" over declared value domains along with associated fuzzy qualifiers, which are membership functions over the domain of the fuzzy variable they qualify.[16] Then, a context called a "*frame*" in the form of a record structure whose fields are fuzzy variables is defined along with "*actions*" which are instructions side-effecting some of these variables, and "*daemons*" which are conditional triggers of such actions depending on the current properties of some the frame's fuzzy fields. This enables applications known as "*Mamdani-type controllers*," after Ebrahim Mamdani's original formulation ([91] and [90]) of this kind of fuzzy state controller which adapted Zadeh's earlier idea [133].

  Such essentially amounts to generic operational tools easing the specification in LPA Prolog of a primitive fuzzy expert system shell using fuzzified "condition-action" rules *à la* Business Rules. It is "primitive" in that regard since there have been several other systems supporting efficient implementation of such condition-action rules over object structures using fuzzy RETE matching (*e.g.*, FuzzyShell [98] or LISP-based FuzzyCLIPS [97]).

  Authors' comment:

  See also the following related (but only marginally) work:

  - *Tony Abou-assaleh* et al.*'s "Relaxed term unification" in Tony Abou-Assaleh, Nick Cercone, and Vlado Kešelj, "An Overview of the Theory of Relaxed Unification." in* Proceedings of the International Conference on Advances in the Internet, Processing, Systems, Interdisciplinary Research, *IPSI-2003, Sveti Stefan, Montenegro, 2003.*

    This is related but not strictly speaking fuzzy though, just one tolerating any mismatches (functor symbol and arity). It consists of mapping all terms into singleton sets which may then get augmented with contending other terms whereby unification accumulates without resolving and thus never fails. Formalizing is rather cumbersome (instead of using familiar logical variables as in Prolog, they borrow from the notation used in "Head-Phrase Structure Grammars" (HPSGs) for feature path sharing—viz., numbers in squares), and not very well researched (e.g., in this 2003 work, the authors attribute earliest ever unification operation to Robinson in 1965, and their method is definitely not declarative and based on a "recursive descent" algorithm). (They do not even cite [92], nor of course Herbrand [68].)

  - *Hashim Habiballa's "Fuzzy Predicate Logic" which is a generalized fuzzy resolution deductive system on unskolemized sentences. Fuzzification is done at the quantifier level and so guides unification of a variable depending on its quantifier and its fuzzy*

[16] Such qualifiers are known as fuzzy "*linguistic hedges*" ([132], [49]).

weight. This is like postponing skolemization (whether resolution is fuzzy or not) until a variable is accessed, the same occurrence of existentially quantified variables corresponding to the same object (even though no skolem functor for it was actually generated) and depending on all the previous universally quantified variables (all whose unquantified occurrence have been renamed within its scope) in the order of nesting.

- *Peter Vojtáš's "Fuzzy logic programming"* Fuzzy Sets and Systems *124 (2001) 361–370. (Although it says so in the abstract, there is nothing on fuzzy unification* per se*: section 5.3 "Similarities and the problem of fuzzy unification" is basically vacuous)*
- *Ekaterina Komendantskaya's "Unification by Error-Correction," (unification using Neural Nets)* Proceedings of the Fourth International Workshop on Neural-Symbolic Learning and Reasoning Patras *(NeSy'08), Greece, July 21, 2008. Edited by Artur D'Avila Garcez and Pascal Hitzler. See also her PhD thesis at Cork, 2007.*

  For general fuzzy neural nets, see Robert Keller's lecture.[17]
- *Alsinet* et al.*'s "context-dependent fuzzy unification" in "Two formalisms of extended possibilistic logic programming with context-dependent fuzzy unification: a comparative description"* Electronic Notes in Theoretical Computer Science *66(5) (2002).*

Finally, in our searching for work purporting to rely on some kind of fuzzy $\mathcal{FOT}$ unification, since $\mathcal{FOT}$ unification is an essential operation used in Logic Programming ($\mathcal{LP}$) languages such as Prolog, we naturally looked at contributions describing themselves as "*Fuzzy Logic Programming*" (or "Fuzzy Prolog"), seeking to understand what entities they fuzzified and how these were used. But $\mathcal{LP}$ as a programming model has now long been subsumed as a particular instance of Constraint Logic Programming ($\mathcal{CLP}$)—*cf.*, [79], and [70]. The $\mathcal{CLP}$ scheme is formally elegant and operationally simple means to accommodate *any* constraint system besides conventional Prolog's $\mathcal{FOT}$ unification. In $\mathcal{CLP}$, the concept of $\mathcal{FOT}$ and $\mathcal{FOT}$ unification as used in $\mathcal{LP}$, is just a particular constraint system among many others, and consists in solving systems of syntactic ($\mathcal{FOT}$) equations. One can therefore easily accommodate any other alternative, or even concomitant, constraint systems (*i.e.*, any formal data structure subject to "normalizable" constraints) in parallel, using shared variables as a natural and efficient means of communication. There are several implemented $\mathcal{CLP}$ systems—*e.g.*, CIAO.[18] This latter system, for example, provides one such fuzzy $\mathcal{LP}$ language instantiation using a particular fuzzy constraint library [123]. This is what "Fuzzy $\mathcal{CLP}$" does: it fuzzifies logical rules and facts over constrained variables; it does not fuzzify $\mathcal{FOT}$ unification *per se*—it does not need to. These languages provide fuzzy logical connectives (fuzzy **and**, **or** and **not**), and fuzzy predicates at the Horn rule/fact level. This, then, does not address our objective of fuzzifying unification and generalization as lattice operations on the $\mathcal{FOT}$ data structure.

4.4 Proposed Proofs of Concept

The most important consequence of this work, it is hoped, is that it can provide several pragmatic operational improvements on inference and learning methods from purely declarative structure-

[17] https://www.cs.hmc.edu/courses/2004/fall/cs152/slides/fuzzy.pdf
[18] https://ciao-lang.org/

oriented constraint specifications. This is meant to ease efficient implementation and the provision of software tools. In what follows we discuss a minimal set of necessary software development efforts needed to enable the expressive potential of fuzzy lattice operations on $\mathcal{FOT}$s and $\mathcal{OSF}$ graphs.

Interfaces (Java packages and libraries) for $\mathcal{FOT}$s and $\mathcal{OSF}$ graphs.

We have started an implementation in Java of the operational semantics derived from the axioms and rules that we presented and proved correct in this article which has allowed us to confirm our results on concrete examples [10].[19] This was eased by the fact that the fuzzy lattice operations do no require altering these conventional first-order structures.

§ Closing Declared Fuzzy Taxonomies

In the crisp case, declaring an ordering on sorts defines a set of pairs. The complete ordering itself is then generated as the reflexive-transitive closure of this declared set of pairs ("$s_1 \preceq s_2$"). This is taken to great advantage to compile it statically for the efficient computation of Boolean lattice operations on sorts when each sort is represented as a bit vector of as many bits as there are sorts, and carries a bit for each index of a sort it subsumes. Thus, the time and space complexity of all three Boolean lattice operations is quasi-constant on the size of the taxonomy, since this amounts to compiling each sort into a native binary word of size equal to the total number of sorts (see [12] and [11]).

For a fuzzy subsumption ordering on sorts, the same kind of reflexive-transitive closure may be statically computed. However, the information carried by each pair of the fuzzy relation is no longer $\{0,1\}$-valued but $[0,1]$-valued (the value of the similarity degree α in declaring "$s_1 \preceq_\alpha s_2$"), and may no longer be represented as a bit. Now, instead of a bit-vector, it is a fuzzy set of the form $\{\, \alpha/s \mid \alpha \in (0,1] \text{ and } s \in \mathcal{S} \,\}$. The bitwise Boolean operations on bit-vectors are now fuzzified into $\wedge$, $\vee$, and $\lambda\alpha.(1-\alpha)$ on fuzzy set elements. Each of these operations works on fuzzy sets to yield the fuzzy set of sorts obtained from applying the operation to the corresponding truth values of each sort. Namely:

$$X \wedge Y \stackrel{\text{def}}{=} \{\, (\alpha \wedge \beta)/s \mid \alpha/s \in X \text{ and } \beta/s \in Y, \text{ for all } s \in \mathcal{S} \,\} \tag{4.9}$$

$$X \vee Y \stackrel{\text{def}}{=} \{\, (\alpha \vee \beta)/s \mid \alpha/s \in X \text{ and } \beta/s \in Y, \text{ for all } s \in \mathcal{S} \,\} \tag{4.10}$$

$$\overline{X} \stackrel{\text{def}}{=} \{\, (1-\alpha)/s \mid \alpha/s \in X, \text{ for all } s \in \mathcal{S} \,\} \tag{4.11}$$

for all X and Y fuzzy sets over a reference set of sorts $\mathcal{S}$.

Similarly with composition with a similarity degree α and fuzzy set X over $\mathcal{S}$:

$$\alpha \wedge X \stackrel{\text{def}}{=} \{\, (\alpha \wedge \beta)/s \mid \beta/s \in X, \text{ for all } s \in \mathcal{S} \,\} \tag{4.12}$$

and:

$$\alpha \vee X \stackrel{\text{def}}{=} \{\, (\alpha \vee \beta)/s \mid \beta/s \in X, \text{ for all } s \in \mathcal{S} \,\}. \tag{4.13}$$

[19] See also [52], a recent extension of the Bousi-Prolog system based on our similarity-based unification tolerating functors of differing arities.

Since, by definition, we never represent explicitly a 0 similarity degree fuzzy element (*i.e.*, of the form $0/s$), we never store it explicitly either in a fuzzy set representation. Hence, in all the foregoing defining Equation (4.9)–Equation (4.13), by " *for all* $s \in \mathcal{S}$" it is assumed that whenever $\_/s \notin X$, for some sort $s \in \mathcal{S}$ and fuzzy set X on $\mathcal{S}$, this is formally equivalent to $0/s \in X$.

It will always be assumed that a top sort ("$\top$") and a bottom sort ("$\bot$") are implicitly declared such that, for all sorts s:

$$s \preceq_1 \top \tag{4.14}$$

and,

$$s \neq \top \implies \top \preceq_0 s \tag{4.15}$$

as well as:

$$\bot \preceq_1 s \tag{4.16}$$

and,

$$s \neq \bot \implies s \preceq_0 \bot \tag{4.17}$$

to express that there is no fuzziness in the sort ordering of $\top$ as the greatest all-encompassing sort and $\bot$ as the least all-excluding sort.

In [12] and [11], the encoding of crisp-ordered sorts as bit vectors is given in pseudo-code as a transitive-reflexive closure of the set of pairs of sort declarations of the form "$s_i \preceq s_j$." As expected, the process of propagating the similarity degrees declared in the fuzzy partial order of sorts is also a reflexive-transitive closure procedure. One will easily see that it is a direct homomorphic adaptation of the bit-vector procedure recalled in [11] obtained by transforming the Boolean bit-vector representation and operations into their homomorphic fuzzy-set generalizations. It is given as the pseudocode procedure `CloseFuzzyTaxonomy` expressed as Algorithm 1.

The class `Sort` is the type representing partially-ordered symbols making up a concept taxonomy. We will also assume that known sorts are stored in a global (static) hash table, called `taxonomy`, associating strings (sort names) to `Sort` objects. A global (static) method `getSort (String)` will return a sort given its name.

The class `Sort` has a field called "`children`" of type `Set⟨Sort,double⟩` containing, for any sort, the sets of sorts that are its immediate children in the taxonomy, each paired with a non-zero similarity degree. Thus, for every sort object, this set is filled with sorts by processing fuzzy "$\preceq$" expressions of the form $s_1 \preceq_\alpha s_2$ used to declare that sort s_1 is subsumed (or is a subsort of) sort s_2 with similarity degree $\alpha \in (0, 1]$; namely, $(s_1, \alpha) \in s_2$.`children`. The class `Sort` has another field called "`parents`" of type `Set⟨Sort⟩` containing, for any sort, the sets of sorts that are its immediate parents in the taxonomy. There is no need to record the similarity degrees as well in the `parents` sets because the similarity degrees will only be accessed through the sets `children` while closing a fuzzy taxonomy.

In addition, the class `Sort` has:

```
1  procedure CloseFuzzyTaxonomy ()
2  |  Set⟨Sort⟩ layer ← ⊥.parents;
3  |  while layer ≠ ∅ do
4  |  |  foreach Sort s ∈ layer do
5  |  |  |  s.fuzzyset ← {(s,1)} ∨ ⋁_{(u,α)∈s.children} (α ∧ u.fuzzyset);
6  |  |  |  s.closed ← true;
7  |  |  end
8  |  |  layer ← ⋃_{s∈layer} s.parents;
9  |  |  foreach s ∈ layer do
10 |  |  |  if ∃(u,_) ∈ s.children such that ¬u.closed then
11 |  |  |  |  layer.remove(s);
12 |  |  |  end
13 |  |  end
14 |  end
15 end
```

Algorithm 1: Encoding of a fuzzy sort taxonomy as fuzzy-set codes

- an integer field called "`index`" that is a sort's unique characteristic rank in the array `taxonomy` containing all the sorts;
- a field called "`fuzzyset`" of type `Set⟨Sort,double⟩` initialized to the empty fuzzy set (*i.e.*, equivalent to all pairs of distinct declared sorts having 0.0 similarity degree); this represents the fuzzy set computed by reflexive-transitive closure. Upon completion of the closure, it ends up containing, for each sort $s_i \in$ `taxonomy`, the similarity degree $\alpha_{ij} \in (0,1]$ of its $\preceq$ relationship with all sort $s_j \in$ `taxonomy` (*i.e.*, such that $s_i \preceq_{\alpha_{ij}} s_j$);
- a `Boolean` field called "`closed`" indicating whether this sort has been closed or not (so it is initially set to **false**).

Authors' comment: *Give an example ...*

[To be completed. . .]

§ Optimizing Closing and Lattice Operations

There is an immediate issue that the reader should have in mind with using the foregoing "sort-as-fuzzy-set" representation and the closing procedure on these fuzzy sets. Namely, while Algorithm 1 is clearly formally correct as a lattice-homomorphic image of the crisp case, the motivation for casting sorts into the Boolean lattice of bit-vector codes seems compromised in the new representation of sorts as fuzzy sets exposed in [12], and used in [11] and in [9]. After

closing it, a sort's bit-vector represents the set of its lower bounds. Indeed, recall that this enabled optimizing set lattice operations on ordered set-denoting sorts (very fast on bit vectors), with a minimal sort representation (a bit vector being essentially a non-negative integer), which can be further compacted using a given declared ***is-a*** ordering's specific topology [12]. With a fuzzy partial-order, however, a sort is no longer identified with a bit vector but with a $(0, 1]$-fuzzy set. Therefore, a compact fuzzy-set representation upon which an efficient intersection operation may be computed must be provided in order to minimize impairing the efficient-implementation motivation.

We discuss here a sensible data-structure representation for the fuzzy set encoding a fuzzy-ordered sort in a finite set of a declared fuzzy taxonomy, supporting a better-than-*naïve* implementation of its lattice operations.

We shall call "*reference base*" the set of minimal upper bounds of $\bot$; namely, the set of sorts in $\bot$`.parents`, the first `layer` in Algorithm 1. We may also refer to the reference base as the set of instances (*i.e.*, each instance identifies a singleton-denoting sort).

Note that in Algorithm 1, the class `Sort`'s field `fuzzyset` is actually a fuzzy set where the fuzzy elements are pairs α/s where s may be any sort in `taxonomy`, not just a sort in the reference base. However, each sort formally denotes a fuzzy distribution on this reference base. So we may also find it useful to identify the reference base as a global array called `base` of N non-negative integers. This number N is the number of elements in the reference base; *viz.*, $N \stackrel{\text{def}}{=} |\bot\texttt{.parents}|$. Each value `base[i]`, $\texttt{i} = 1, \ldots, N$ is the index of a sort in the static hash table `taxonomy` that is minimal (*i.e.*, in $\bot$`.parents`). Hence, rather than a field `fuzzyset`, the class `Sort` is given a field called `fuzzybase` to represent this fuzzy set as an array of $N \leq$ `taxonomy.size()` of $[0, 1]$-values for each index in `base`. In other words, for a sort s, s`.fuzzybase` is an array of N similarity degrees and s`.fuzzybase[i]` is the similarity degree of `base[i]`.

For any sort `s`, the array `s.fuzzybase` is approximated by the binary vector we shall define as a new field of type `BitCode` for the class `Sort` called `crispvalue`, a bit vector such that:[20]

$$\texttt{s.crispvalue}[i] \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } \texttt{s.fuzzybase}[i] > 0; \\ 0 & \text{otherwise.} \end{cases}$$

This information is therefore immediate to derive from any closed fuzzy sort taxonomy and can be used in the abstract interpretation of the three fuzzy Boolean lattice operations on sorts to restrict enumeration of a fuzzy set's elements only to non-zero indices using the bit-vector operations defined in [12] and [11].

4.5 Applications

Example of applications and use in knowledge and data processing (deduction and learning), linguistics (fuzzy order-sorted HPSGs).

Authors' comment: *Take a look later at applications:*

[20] For example, a Java class such as `hlt.osf.util.BitCode`, which extends the standard Java class `java.util.BitSet`.

- *Zadeh's fuzzy interpretation of linguistic hedges [132]*
- *Common Fuzzy Distributions for linguistic hedges [49]*
- *Other sorts of application (fuzzy control in particular) [134].*

4.6 Use Case

Approximate Information Retrieval.

Chapter 5

Version of January 8, 2019

Conclusion

5.1 Recapitulation

We overviewed several ways to fuzzify $\mathcal{OSF}$ constraint logic. We hope to have provided enough evidence that what we describe in this document is to benefit Information Retrieval as well as Machine Learning when fuzzy-$\mathcal{OSF}$ can provide a precious initial focusing step prior to exploiting number-analytical techniques used in Inductive Logic Programming [108] or Bayesian Nets [67].

> ***Authors' comment:*** *This last point should perhaps be detailed somewhat talking about Yutaka Sasaki's work: [111], [109], [108], [110], [21]. (Or perhaps just mention and cite his work here, and then elaborate in a separate paper on fuzzy $\mathcal{OSF}$ learning with him as co-author?)*

In this context, we focused on the lattice-theoretic properties of fuzzy $\mathcal{OSF}$ graphs. Such fuzzy structures may be put to use in approximate Knowledge Representation to offer a more flexible means to perform deduction and induction over abstract attributed objects and concepts represented as fuzzy order-sorted feature graphs.

We also discussed several fuzzy versions of related topics, from (Lattice) Algebra,to Automata Theory to (Description) Logics, to Data Structures.

5.2 Further work

The most immediate avenue of research that the issues discussed in this paper open up is the design (specification and implementation) of a $\mathcal{CLP}$ language that would be the "least upper bound" of $\mathcal{LOG}\backslash$ [15] (and later $\mathcal{LIFE}$ [7, 14]) with a fuzzy Logic Programming language such as FASILL [77]. Such a language, with access to distributed databases, would facilitate efficient approximate reasoning for query resolution as well as learning by approximate knowledge acquisition as fuzzy order-sorted feature structures. All the applications enabled by Fuzzy Logic Programming on one side and $\mathcal{OSF}$ Logic Programming on the other could then each benefit as each would thereby gain even more expressivity and flexibility for the processing of approximate

structured knowledge on massive data. While fuzzy $\mathcal{OSF}$ unification (the conjunctive connective) is the key to *deduction* (as used in logical or function rule invocation), fuzzy $\mathcal{OSF}$ generalization (the disjunctive connective) is the key to *induction* (as used in learning by extrapolating knowledge from data).

The next, of course, is the development of applications: from Information Retrieval, to Natural Language, to Knowledge Processing. The potential is immense.

Concomittantly are all the pragmatic issues: efficient implementation (preprocessing, abstract machine compiling, interfacing to constraint-solving and fuzzy-set libraries, *etc.*).

Clearly, the possibilities are legion.

Appendix A

Version of January 8, 2019

Background Material

In this appendix, we provide a succint summary of formal material, terminology, and notation constituting background for the issues developed in this work.

- Appendix A.1 reviews basic lattice-theoretic notions and notation. Section A.1.1 gives the essentials. Section A.1.2 explains modularity and distributivity.
- Appendix A.2 gives the basic formal set-theoretic characterization properties of relations as sets of pairs.
- Appendix A.3 extends these to their corresponding fuzzy notions: Section A.3.1 to fuzzy relations; Section A.3.2 to fuzzy equivalence relations (called similarities); and, Section A.3.3 to fuzzy partial orders.
- Appendix A.4 recalls basic definitions and properties of $\mathcal{FOT}$ substitutions represented, as we do in this work, as finitely non-indentical variable-to-terms mappings.
- Appendix A.5 contains the procedural $\mathcal{FOT}$ generalization algorithms as formulated in 1970, one by John Reynolds and the other by Gordon Plotkin.
- Appendix A.6 presents the clause-driven $\mathcal{OSF}$-generalization rules given in [21].
- Appendix A.7 discusses what "up to tag renaming" can allow in the process of $\mathcal{OSF}$ clause normalization.

A.1 Basic Lattice Theory

We recall here basic background from Lattice Theory that we rely on in this work. We restrict ourselves to notions that are relevant to our presentation with the aim to make it as self-contained as possible. For a comprehensive treatise on the subject, the definitive reference is Birkhoff's book [40].

A.1.1 Essentials

Let $L, \leq$ be a poset.

If all pairs of elements x and y in L admit a unique greatest lower bound (**glb**) in L, noted $x \wedge y$, then $L, \leq, \wedge$ is called a *(lower) semi-lattice*. Dually, if all pairs of elements x and y in L admit a unique least upper bound (**lub**) in L, noted $x \vee y$, then $L, \leq, \vee$ is called an *(upper) semi-lattice*.

DEFINITION A.1 (LATTICE) *A* lattice $L, \leq, \wedge, \vee$ *is a poset such that* $L, \leq, \wedge$ *is a lower semi-lattice and* $L, \leq, \vee$ *is an upper semi-lattice.*

A lower semi-lattice $L, \leq, \wedge$ is called *complete* if *all* (*i.e.*, even non-finite) subsets $X \subseteq L$ admit a **glb** in L. Dually, an upper semi-lattice $L, \leq, \vee$ is called a *complete*, if *all* subsets $X \subseteq L$ admit a **lub** in L.

A lattice $L, \leq, \wedge, \vee$ is *lower-complete* iff $L, \leq, \wedge$ is a complete lower semi-lattice. Dually, it is *upper-complete* iff $L, \leq, \vee$ is a complete upper semi-lattice. A lattice in *complete* if it both lower-complete and upper-complete.

When L has a greatest element, we call it "*top*" and write it "$\top$." Note that by definition $\top \stackrel{\text{def}}{=} \bigvee L$. Dually, when L has a least element, we call it "*bottom*" and write it "$\bot$." Also note that by definition $\bot \stackrel{\text{def}}{=} \bigwedge L$.

DEFINITION A.2 (MODULAR INEQUALITY) *In any lattice* $L, \leq, \wedge, \vee$, *the following holds:*

$$\text{if } x \leq y \text{ then } x \vee (z \wedge y) \;\leq\; (x \vee z) \wedge y \tag{A.1}$$

for all x, y, z *in* L.

DEFINITION A.3 (DISTRIBUTIVE INEQUALITIES) *In any lattice* $L, \leq, \wedge, \vee$, *the following two inequalities hold:*

$$x \wedge (y \vee z) \;\geq\; (x \wedge y) \vee (x \wedge y), \tag{A.2}$$

$$x \vee (y \wedge z) \;\leq\; (x \vee y) \wedge (x \vee y) \tag{A.3}$$

for all x, y *in* L.

DEFINITION A.4 (COMPLEMENTED LATTICE) *A* complemented lattice *is a lattice* $L, \leq, \wedge, \vee$ *with top* $\top$ *and bottom* $\bot$ *in which for any element* $x \in L$, *there is a unique* complement $\overline{x} \in L$ *verifying:*

$$x \vee \overline{x} \;=\; \top \tag{A.4}$$

and:

$$x \wedge \overline{x} \;=\; \bot \tag{A.5}$$

DEFINITION A.5 *A* Boolean Lattice *is a distributive complemented lattice.*

A.1.2 Modularity, Distributivity

Intuitively, when thinking of an ordering as comparing information contents, submodularity—*i.e.*, Inequality (A.1)—and subdistributivity—*i.e.*, Inequality (A.2) or Inequality (A.3)—express the fact that there may be non-uniform distribution of information either horizontally (modularity) or vertically (distributivity). When equalities rather than inequalities are required to hold always, this restricts the class of lattices to fully modular and fully distributive lattices.

DEFINITION A.6 (MODULAR LATTICE) *A* modular lattice $L, \leq, \wedge, \vee$ *is a lattice in which the modular inequality* (A.1) *becomes an equality everywhere; viz.,*

$$\text{if } x \leq y \text{ then } x \vee (z \wedge y) \;=\; (x \vee z) \wedge y \tag{A.6}$$

for all x, y, z *in* L.

A useful way to visualize what makes a lattice be modular is that, for all pair of elements x and y, the interval between x and $x \vee y$ ($=$ **lub**(x, y)) is order-isomorphic with the interval between $x \wedge y$ ($=$ **glb**(x, y)) and y. In other words, in a modular lattice, information contents varies isomorphically along edges of diamond-shaped order diagrams. This is why modularity is often referred to as the "diamond isomorphism" property.[1] Formally, this means that for any pair x and y, the two functions $\lambda u.\big(u \vee x\big)$ and $\lambda v.\big(v \wedge y\big)$ are mutually inverse order isomorphisms.

The following result provides a simple test of modularity.

THEOREM A.1 (MODULARITY CONDITION) *Any lattice that admits a sublattice isomorphic to the 5-element lattice on the left side of Figure A.1 is* not *modular.*

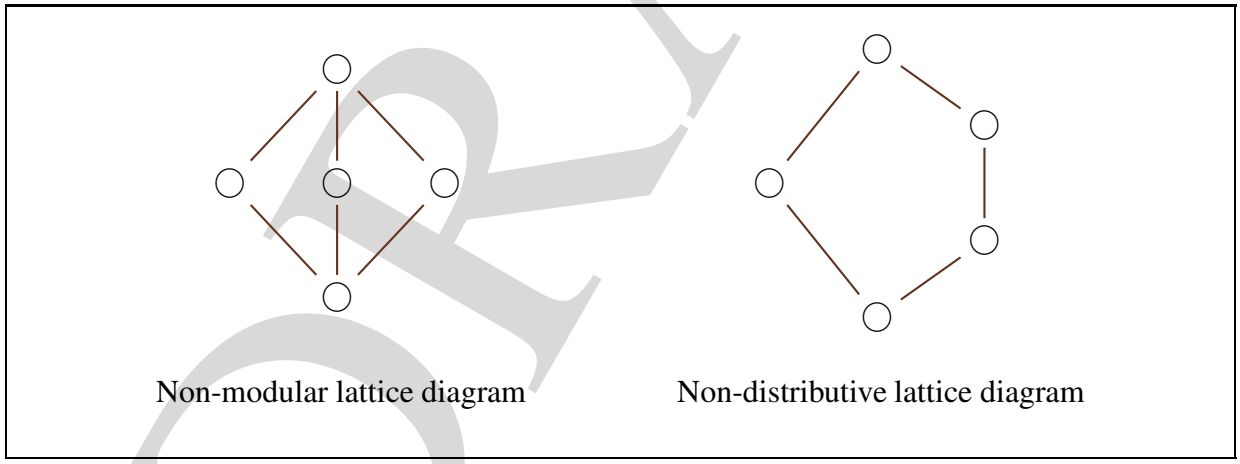


Figure A.1: Non-modular and non-distributive lattice diagrams

DEFINITION A.7 (DISTRIBUTIVE LATTICE) *A* distributive lattice $L, \leq, \wedge, \vee$ *is a lattice in which the inequality in condition* (A.2)*—or, equivalently the inequality in condition* (A.3)*—is strengthened into an* equality *everywhere.*

[1] https://en.wikipedia.org/wiki/Modular_lattice#Diamond_isomorphism_theorem

Note that:

THEOREM A.2 *Every distributive lattice is also modular.*

A way to visualize whether a lattice is distributive is when all paths between two related elements have equal lengths; namely, information contents varies uniformly vertically. This is captured by the following properties.

The following result provides a simple test of distributivity.

THEOREM A.3 (DISTRIBUTIVITY CONDITION) *Any lattice that admits a sublattice isomorphic to the 5-element lattice on the right side of Figure A.1 is* not *distributive.*

A.2 Crisp Relations

In (crisp) Set Theory, a binary relation r on a set S is a subset of $S \times S$. We say that r is:

- *reflexive* iff:

$$\mathbb{1}_{S\times S} \subseteq r \tag{A.7}$$

where $\mathbb{1}_S$ is the *identity* relation on S defined as: $\mathbb{1}_{S\times S} \stackrel{\text{def}}{=} \{\langle x, x\rangle \mid x \in S\}$;

- *symmetric* iff:

$$r = r^{-1} \tag{A.8}$$

where r^{-1} is the *inverse* relation of r defined as: $r^{-1} \stackrel{\text{def}}{=} \{\langle y, x\rangle \in S \times S \mid \langle x, y\rangle \in r\}$;

- *antisymmetric* iff:

$$r \cap r^{-1} \subseteq \mathbb{1}_{S\times S} \tag{A.9}$$

where $r \cap r'$ is the intersection of r and r'; *viz.*, the relation on S defined as: $r \cap r' \stackrel{\text{def}}{=} \{\langle x, y\rangle \in S \times S \mid \langle x, y\rangle \in r$ **and** $\langle x, y\rangle \in r'\}$;

- *transitive* iff:

$$r \circ r \subseteq r \tag{A.10}$$

where $r \circ r'$ is the composition of r and r'; *viz.*, the relation on S defined as: $r \circ r' \stackrel{\text{def}}{=} \{\langle x, y\rangle \in S \times S \mid \langle x, z\rangle \in r$ **and** $\langle z, y\rangle \in r'$ for some $z \in S\}$.

DEFINITION A.8 (PREORDER) *A relation r on a set S is a preorder on S iff it is reflexive and transitive; i.e., iff r verifies conditions* (A.7) *and* (A.10).

DEFINITION A.9 (EQUIVALENCE) *A symmetric preorder r on a set S is called an equivalence on S; that is, r verifies conditions* (A.7), (A.8), *and* (A.10).

Such an equivalence relation $\equiv$ on a set S defines a partition of this set; namely, a collection of non-empty subsets S_i, $1 \leq i \leq \mathbf{I}_{\equiv} \in \mathbb{N}$, of S (the equivalence classes) such that:

$$1 \leq i \neq j \leq \mathbf{I}_{\equiv} \implies S_i \cap S_j = \emptyset \tag{A.11}$$

and:

$$S = \bigcup_{i \leq \mathbf{I}_{\equiv}} S_i \tag{A.12}$$

where $\mathbf{I}_{\equiv}$, the *index* of $\equiv$, is the number of equivalence classes of $\equiv$ forming the partition of S. The equivalence class of an element of $x \in S$ is denoted $[x]^{\equiv}$ and is defined as:

$$[x]^{\equiv} \stackrel{\text{def}}{=} \{\, y \in S \mid x \equiv y \,\}. \tag{A.13}$$

DEFINITION A.10 (PARTIAL ORDER) *A relation r on a set S is a partial order on S iff it is an antisymmetric preorder on S; i.e., iff r verifies conditions* (A.7), (A.9), *and* (A.10).

A.3 Fuzzy Set Algebra

In this section, we recall some essential terminology and notation on Fuzzy Set algebra used in this document. The fuzzy operator symbol on $[0,1]^2$ we shall use for fuzzy conjunction, also called T-norm,[2] is $\wedge$ (resp., $\vee$ for its fuzzy dual operation). This is generally interpreted as **min** (resp., **max**); *e.g.*, in Zadeh's seminal paper [130]. But other fuzzy operation interpretations can be considered depending on the desired effect.

Thus, all the issues considered in this document are generic in the choice of fuzzy operators. Indeed, in fuzzifying $\mathcal{OSF}$ terms, or anything for that matter, it is important to realize that many kinds of fuzziness may be obtained depending on the choice of these operators. In this section, we make some general points regarding the interpretations of fuzzy operators over the continuous interval $[0,1]$ other than the classical **min** and **max**.

A conventional set S of elements of a universe $\mathcal{U}$ is identified to its Boolean-valued characteristic function $\mathbf{1}_S : \mathcal{U} \to \{\mathfrak{false}, \mathfrak{true}\}$ so that for all x in $\mathcal{U}$, $x \in S$ iff $\mathbf{1}_S(x) = \mathfrak{true}$. This defines a Boolean algebra isomorphism between the set of subsets of elements of $\mathcal{U}$ and $\{\mathfrak{false}, \mathfrak{true}\}$-valued functions on $\mathcal{U}$. When identifying a set to its Boolean characteristic function, set operations become logical operations: intersection becomes conjunction [$\mathbf{1}_{S \cap S'}(x) \stackrel{\text{def}}{=} \mathbf{1}_S(x)$ **and** $\mathbf{1}_{S'}(x)$], union becomes disjunction [$\mathbf{1}_{S \cup S'}(x) \stackrel{\text{def}}{=} \mathbf{1}_S(x)$ **or** $\mathbf{1}_{S'}(x)$], and complementation becomes negation [$\mathbf{1}_{\overline{S}}(x) \stackrel{\text{def}}{=}$ **not** $\mathbf{1}_S(x)$]. Another equivalent Boolean algebra isomorphism is the one identifying the logical constants $\mathfrak{false}$ and $\mathfrak{true}$ to the numerical values 0 an 1, respectively, and the logical operations **and**, **or**, and **not**, to the numerical operations **min**, **max**, and $\lambda x.(1-x)$, respectively. This is because these numerical operations on the numbers 0 and 1 stay isomorphically internal to $\{0,1\} \subset [0,1]$ with $0 < 1$. Formally, this amounts to plunging the discrete 2-valued poset $(\{0,1\}, \leq)$ homomorphically into the continuous unit interval $([0,1], \leq)$ with the identical numerical operations (**min**, **max**, and $\lambda x.(1-x)$). Hence, when

[2] See https://en.wikipedia.org/wiki/T-norm.

seen as $[0,1]$-valued functions, these latter operations are a homomorphic extension of conventional Boolean algebra. This was the interpretation proposed originally by Zadeh in his seminal article on Fuzzy Sets [130].

However, there are many other ways in which conventional two-valued Boolean $\{0,1\}$-Logic may be fuzzified by being extended into a multiple-valued logic depending on whether it allows multiple discrete or continuous similarity degrees in $[0,1]$ (or any suitable "*L-structure*" [63, 56]).

In essence, *any* Boolean *Logic* can be fuzzified by extending its basic Boolean connectives (1) **and**, (2) **or**, and (3) **not**, on the $\{0,1\}$ similarity degrees of characteristic function (and thus the set operations (1) intersection, (2) union, and (3) complementation), into generic Boolean *Lattice* operations $\wedge$, $\vee$, $\lambda x.\overline{x}$ on $[0,1]$-valued functions whereby:

- $\mathbf{1}_{S\cap S'}(x) \stackrel{\text{def}}{=} \mathbf{1}_S(x) \wedge \mathbf{1}_{S'}(x)$,
- $\mathbf{1}_{S\cup S'}(x) \stackrel{\text{def}}{=} \mathbf{1}_S(x) \vee \mathbf{1}_{S'}(x)$,
- $\mathbf{1}_{\overline{S}}(x) \stackrel{\text{def}}{=} \overline{\mathbf{1}_S}(x)$.

This may look innocuous a precision, but it turns out that, as thoroughly explained and illustrated in particular in Dubois and Prade's comprehensive treatise on Fuzzy Sets and Systems [56], there are many other possible choices for the lattice operations on $[0,1]$ for $\wedge$, $\vee$, and $\lambda x.\overline{x}$ besides **min**, **max**. It is for this reason, and with no loss of generality, that we use the former three operations rather than the latter in this document.

All we need are lattice operations $\wedge$ and $\vee$ (and $\lambda x.\overline{x}$) on $[0,1]$, which can then be made to apply to fuzzy sets defined as functions in $[0,1]^{\mathcal{U}}$ by pointwise extension. Namely:

$$\begin{array}{rcl} \wedge & : & [0,1]^{\mathcal{U}} \times [0,1]^{\mathcal{U}} \rightarrow [0,1]^{\mathcal{U}} \\ \vee & : & [0,1]^{\mathcal{U}} \times [0,1]^{\mathcal{U}} \rightarrow [0,1]^{\mathcal{U}} \\ \lambda x.\overline{x} & : & [0,1]^{\mathcal{U}} \rightarrow [0,1]^{\mathcal{U}} \\ \top & \stackrel{\text{def}}{=} & \lambda x.1 \\ \bot & \stackrel{\text{def}}{=} & \lambda x.0 \end{array} \tag{A.14}$$

so that:

- $[0,1]^{\mathcal{U}}, \wedge, \top$ is a commutative monoid,
- $[0,1]^{\mathcal{U}}, \vee, \bot$ is a commutative monoid,
- $\wedge$, $\vee$ are mutually distributive,
- $\overline{\bot} = \top$,
- $\overline{\top} = \bot$,

and, for all fuzzy sets ϕ, ϕ_1, ϕ_2:

- $\phi \wedge \bot = \bot$,
- $\phi \vee \top = \top$,

- $\overline{\phi_1 \wedge \phi_2} \;=\; \overline{\phi_1} \vee \overline{\phi_2}$,
- $\overline{\phi_1 \vee \phi_2} \;=\; \overline{\phi_1} \wedge \overline{\phi_2}$,
- $\overline{\overline{\phi}} \;=\; \phi$.

From this, a plethora of familiar algebraic and order-theoretic properties ensue [40]. In the literature, the "$\wedge$" operator is sometimes called "*T-norm*" (for "*triangular* norm"), while the "$\vee$" operator is sometimes called "*T-conorm*" or "*S-norm*." They can be derived from one another by duality using:

$$\begin{array}{rcl} \phi_1 \wedge \phi_2 & = & \overline{\big(\overline{\phi_1} \vee \overline{\phi_2}\big)} \\ \phi_1 \vee \phi_2 & = & \overline{\big(\overline{\phi_1} \wedge \overline{\phi_2}\big)}. \end{array} \tag{A.15}$$

In particular, with $\overline{\phi} \stackrel{\text{def}}{=} 1 - \phi$:

$$\begin{array}{rcl} \phi_1 \wedge \phi_2 & = & 1 - \big((1-\phi_1) \vee (1-\phi_2)\big) \\ \phi_1 \vee \phi_2 & = & 1 - \big((1-\phi_1) \wedge (1-\phi_2)\big). \end{array} \tag{A.16}$$

For example, here are three popular such $\wedge$ and $\vee$ operations on $[0,1]$ used in practice [56], [77]:

- "Gödel" fuzzy operators:

$$\left\{ \begin{array}{rcl} \alpha_1 \wedge_{_G} \alpha_2 & \stackrel{\text{def}}{=} & \mathbf{min}(\alpha_1, \alpha_2) \\ \alpha_1 \vee_{_G} \alpha_2 & \stackrel{\text{def}}{=} & \mathbf{max}(\alpha_1, \alpha_2) \end{array} \right. \tag{A.17}$$

- "Product" (or "probabilistic") fuzzy operators:

$$\left\{ \begin{array}{rcl} \alpha_1 \wedge_{_P} \alpha_2 & \stackrel{\text{def}}{=} & \alpha_1 \alpha_2 \\ \alpha_1 \vee_{_P} \alpha_2 & \stackrel{\text{def}}{=} & \alpha_1 + \alpha_2 - \alpha_1 \alpha_2 \end{array} \right. \tag{A.18}$$

- "Łukasiewicz" fuzzy operators:

$$\left\{ \begin{array}{rcl} \alpha_1 \wedge_{_L} \alpha_2 & \stackrel{\text{def}}{=} & \mathbf{max}(0, \alpha_1 + \alpha_2 - 1) \\ \alpha_1 \vee_{_L} \alpha_2 & \stackrel{\text{def}}{=} & \mathbf{min}(\alpha_1 + \alpha_2, 1) \end{array} \right. \tag{A.19}$$

Choosing any of these, or others, will determine how fuzzy inference is affected by each argument. For example, contrary to the "Gödel" fuzzy conjunction $\wedge_G$ that imposes the value of one over the other of two truth values, the "Product" version $\wedge_P$ is less "drastic" and will take a more balanced consideration of the values of both arguments. There are a few other fuzzy operators that have been given specific denominations that correspond to particular situations.[3] But one may design their adequate $\wedge$ operator (or $\vee$ operator since one can be derived from the other by duality).[4]

However, in all the actual numerical examples provided in this document for illustration, we use **min** (resp., **max**).

[3] See, *e.g.*: http://www.nicodubois.com/bois5.2.htm.

[4] Designing specific fuzzy norms can be done visually in 3D using publicly available tools such as, *e.g.*, http://www.math.uri.edu/~bkaskosz/flashmo/graph3d2/.

A.3.1 Fuzzy relation

Let us now fuzzify the conventional set theoretic definitions recalled in A.2. It is a straightforward homomorphic extension of the conventional view of (crisp) sets as $\{0,1\}$-valued functions to $[0,1]$-valued functions. Indeed, the former are just a particular case of the more general (fuzzy) sets seen as $[0,1]$-valued characteristic functions.[5] That is, all the fuzzy notions are obtained as straightforward extensions of their crisp counterparts through a Boolean lattice homomorphism. The advantage of the fuzzy extension over conventional sets is that, being structurally richer, it is more expressive. It is a homomorphic extension insofar as all the formal algebraic properties of fuzzy sets and fuzzy-set connectives reduce to their conventional crisp versions when unfuzzifying the truth value $\phi(x)$ of every fuzzy element $\phi(x)/x$ of a fuzzy set ϕ into a crisp value in $\{0,1\}$ for truth values which, when compared to a given value α in $[0,1]$, are either strictly less (assimilated to 0), or greater or equal (assimilated to 1). Informally, this is the crisp set of elements with "*at least*" α as fuzzy truth value. This is called a fuzzy set's "*α-cut*" ϕ_α such that $\phi_\alpha(x) \stackrel{\text{def}}{=} 0$ whenever $\phi(x) < \alpha$ and $\phi_\alpha(x) \stackrel{\text{def}}{=} 1$ whenever $\phi(x) \geq \alpha$, for any *truth threshold* α in $[0,1]$.

DEFINITION A.11 (FUZZY RELATION) *A fuzzy relation on a set S is a fuzzy set on $S \times S$.*

The following properties generalize those of crisp binary relations seen in A.2. Like in the crisp case, we will look closer at essentially two kinds of fuzzy binary relations: fuzzy orders and fuzzy equivalences.[6]

Recall that a fuzzy set ϕ on a set S is a function $\phi : S \to [0,1]$. Let $\rho : S \times S \to [0,1]$ be a fuzzy relation on S. We say that ρ is:

- *reflexive* iff:

$$\mathbb{1}_{S\times S} \leq \rho \tag{A.20}$$

  where $\mathbb{1}_{S\times S}$ is the *fuzzy identity* relation on S defined as: $\mathbb{1}_{S\times S}(x,y) = 1$ if $x = y$ and 0 if $x \neq y$, for all x and y in S; and $\leq$ is *fuzzy set inclusion* defined as: $\rho \leq \rho'$ iff $\rho(x,y) \leq \rho'(x,y)$, for all x and y in S;

- *symmetric* iff:

$$\rho = \rho^{-1} \tag{A.21}$$

  where ρ^{-1} is the *fuzzy inverse* of ρ; *viz.*, the fuzzy relation on S defined as: $\rho^{-1}(x,y) \stackrel{\text{def}}{=} \rho(y,x)$, for all x and y in S;

- *antisymmetric* iff:

$$\rho \wedge \rho^{-1} \leq \mathbb{1}_{S\times S} \tag{A.22}$$

  where the *fuzzy meet* $\rho \wedge \rho'$ is the fuzzy relation on S defined as: $\big(\rho \wedge \rho'\big)(x,y) \stackrel{\text{def}}{=} \rho(x,y) \wedge \rho'(x,y)$, for all x and y in S;

[5] Such a fuzzy characteristic function is called a "*membership function*" in the literature following Zadeh's original terminology [130].

[6] See [128] for even finer and more expressive kinds of useful fuzzy relations that can be defined algebraically.

- *transitive* iff:

$$\rho \circ \rho \;\leq\; \rho \tag{A.23}$$

where the *fuzzy composition* $\rho \circ \rho'$ is the fuzzy relation on S defined as: $\big(\rho \circ \rho'\big)(x, y) \stackrel{\text{def}}{=} \bigvee_{z \in S} \big(\rho(x, z) \wedge \rho'(z, y)\big)$, for all x and y in S.

DEFINITION A.12 (FUZZY PREORDER) *A fuzzy relation ρ on a set S is a fuzzy preorder on S iff it is reflexive and transitive; i.e., iff r verifies conditions* (A.20) *and* (A.23).

A.3.2 Similarity

DEFINITION A.13 (FUZZY EQUIVALENCE) *A fuzzy equivalence ρ on a set S is a fuzzy relation on S which is a symmetric fuzzy preorder on S—that is, ρ verifies conditions* (A.20), (A.21), *and* (A.23).

A fuzzy equivalence relation is also called "*similarity*" relation in the literature [77]. For this reason, we speak of "*similarity degree*" to denote the truth value of a pair so related.

A similarity relation $\sim$ on a set S is a fuzzy equivalence relation on S; *i.e.*, a fuzzy set of pairs of $S \times S$. When S is a finite discrete set, say indexed over $\{1, \ldots, n\}$, since a similarity relation $\sim$ on S is a fuzzy subset of $S \times S$, the three conditions of an equivalence can be visualized on a square $n \times n$ matrix $\sim \in \{1, \ldots, n\}^2 \to [0, 1]$ as follows. For all $i, j, k = 1, \ldots, n$:

- *reflexivity*: $i \sim i = 1$ (*i.e.*, entries on the diagonal are equal to 1);
- *symmetry*: $i \sim j = j \sim i$ (*i.e.*, all symmetric entries on either side of the diagonal are equal);
- *transitivity*: $i \sim k \wedge k \sim j \;\leq\; i \sim j$, for any $k \in \{1, \ldots, n\}$ (*i.e.*, going via an intermediate element will always result in a smaller or equal similarity degree than going directly).[7]

Given a similarity relation $\sim$ on a set S, the subset of $[0, 1]$ denoted $\textbf{DEGREES}^{\sim}$ and defined as $\textbf{DEGREES}^{\sim} \stackrel{\text{def}}{=} \{\, \alpha \in [0, 1] \mid x \sim_\alpha y, \text{ for some } x, y \in S \,\}$ is called the "*similarity degree set*" of $\sim$. A similarity degree $\alpha \in \textbf{DEGREES}^{\sim}$ can thus be used as an approximation threshold, and a similarity can be rendered a crisp equivalence on S by keeping only pairs in $\sim$ with similarity degree greater than or equal to α (*i.e.*, the α-cut of the similarity).

The similarity class $[x]^{\sim}_{\alpha}$ of an element $x \in S$ at an approximation threshold α in $[0, 1]$ given a similarity $\sim$ on S is defined as:

$$[x]^{\sim}_{\alpha} \stackrel{\text{def}}{=} \{\, y \in S \mid x \sim_\beta y, \text{ for some } \beta \in [\alpha, 1] \,\}.$$

Thus, as the similarity degree α decreases from 1 down to 0, more similarities appear in $\sim_\alpha$ between pairs of distinct elements of S that were not related in α-cuts of $\sim$ at greater thresholds. In other words, as α decreases, the equivalence classes of $\sim_\alpha$ grow larger by coalescing classes

[7] Here and elsewhere in this article, we shall use $\wedge/\vee$ for fuzzy conjunction/disjunction in generic formulas. We prefer using these more general symbols in our formalization since which specific fuzzy operations are used is irrelevant. However, we shall use **min**/**max** in all the illustrative examples we give that use actual numbers.

of lesser similarity degrees; that is, for any $x \in S$, if $\alpha \leq \beta$, then $[x]^{\sim}_{\beta} \subseteq [x]^{\sim}_{\alpha}$. In particular, it is always the case that $[x]^{\sim}_{0.0} = S$; indeed, then, all elements are indistinguishable. Note finally that, for any pair $\langle x, y \rangle$ in $S \times S$ and similarity $\sim\colon S \times S \to [0,1]$, if $x \sim_{\alpha} y$ for some α in $[0,1]$, then $x \sim_{\beta} y$ for all $\beta \in [0, \alpha]$. For this reason, unless otherwise specified, *whenever we use an approximation degree as a subscript, we mean the* ***greatest*** *such degree.*

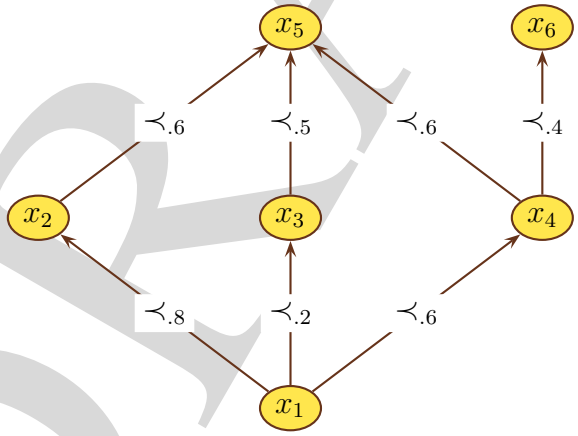
A.3.3 Fuzzy partial order

DEFINITION A.14 (FUZZY PARTIAL ORDER) *A fuzzy relation ρ on a set S is a fuzzy partial order on S iff it is an antisymmetric fuzzy preorder; i.e., iff ρ verifies conditions* (A.20), (A.22), *and* (A.23).

For a fuzzy partial order, as in the case of a fuzzy equivalence relation, when S is a finite discrete set $\{x_1, \ldots, x_n\}$, the three conditions of the above definition can be visualized on a square $n \times n$ matrix $\preceq \; in\, [0,1]^2$ as follows:

- reflexivity and transitivity (just as for a similarity matrix);
- antisymmetry: the matrix must be triangular (up to reordering of columns and lines); this is because $\preceq_{ij} > 0$ implies $\preceq_{ji} = 0$, for all $i, j = 1, \ldots, n$ (*i.e.*, all symmetric entries on either side of the diagonal may not be both non-zero).

For example, the fuzzy binary relation $\preceq$ on the 6-element set $\{x_1, \ldots, x_6\}$ defined as the fuzzy **min/max** reflexive-transitive closure of the following weighted acyclic graph:[8]



corresponds to the following fuzzy matrix:

$$\preceq \;\stackrel{\text{def}}{=}\; \begin{bmatrix} 1 & 0.8 & 0.2 & 0.6 & 0.6 & 0.4 \\ 0 & 1 & 0 & 0 & 0.6 & 0 \\ 0 & 0 & 1 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 & 0.6 & 0.4 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.24}$$

[8] This example is from [131].

upon which these conditions can be verified—which means that the fuzzy relation $\preceq$ so defined is a fuzzy partial order on the set $\{x_1, x_2, x_3, x_4, x_5, x_6\}$.

Note that, just as in the crisp case, any fuzzy preorder $\preceq$ on a set S (*i.e.*, a fuzzy relation on S that is reflexive and transitive) always implicitly defines the following fuzzy relations:

- a similarity $\sim$ on S defined, for any $\alpha \in [0,1]$, as:

$$\sim_\alpha \stackrel{\text{def}}{=} \preceq_\alpha \wedge \succeq_\alpha \tag{A.25}$$

  where $\succeq_\alpha$ is the fuzzy relation defined as: $\succeq_\alpha \stackrel{\text{def}}{=} \preceq_\alpha^{-1}$;

- a fuzzy partial order $\preceq$, a fuzzy set of partial orders $\preceq_\alpha$ on each partition $\Pi_\alpha^\sim$ of S generated by $\sim$ in the fuzzy partition $\Pi^\sim \stackrel{\text{def}}{=} \{\, \Pi_\theta^\sim \mid \theta \in \textbf{DEGREES}^\sim \,\}$, such that:

$$[x]_\alpha^\sim \preceq_\alpha [y]_\alpha^\sim \text{ iff } x \sim_\alpha x' \text{ and } x' \preceq_\alpha y' \text{ and } y' \sim_\alpha y \tag{A.26}$$

  for some $x' \in [x]_\alpha^\sim$ and some $y' \in [y]_\alpha^\sim$.

A.4 First-Order Term Substitutions

This section gives basic terminology and properties of $\mathcal{FOT}$ substitutions as defined in 2.2 where the set-theoretic definition of substitutions as finitely non-identical variable-to-term mappings is given as Expression (2.2).

LEMMA A.1 *Given two substitutions σ and θ in* **SUBST**$_\mathcal{T}$*, the operation defined by Expression (2.2) always results in a substitution in* **SUBST**$_\mathcal{T}$.

> **PROOF** It must be verified that, given σ and θ two finitely non-identical mappings from $\mathcal{V}$ to $\mathcal{T}$, the notation $\sigma\theta$ defined in set-theoretic terms from the set structure of σ and θ by Expression (2.2) always results in a finitely non-identical mapping from $\mathcal{V}$ to $\mathcal{T}$. This is an elementary exercise from the very set-theoretic definition of substitution composition given as Expression (2.2). □

LEMMA A.2 *For any term t in $\mathcal{T}$ and any substitutions σ and θ in* **SUBST**$_\mathcal{T}$*, the expression $\sigma\theta$ defined by Expression (2.2) is a substitution that has the same effect as first applying σ to t, and then applying θ to the result; that is, $\forall t \in \mathcal{T}, \forall \sigma \in$* **SUBST**$_\mathcal{T}, \forall \theta \in$ **SUBST**$_\mathcal{T}$*, $\sqcup(\sigma\theta) = (\sqcup\sigma)\theta$.*

> **PROOF** Expression (2.2) consists of two parts of a (disjoint) set union. The first part of this union consists in the set of pairs t/X in σ transformed into the set of pairs $t\theta/X$ for each each pair t/X in σ. This has for effect to "capture" any potential variables in $\textbf{var}(t\sigma) \cap \textbf{dom}(\theta)$ by mapping directly to $t\sigma\theta$ any variable mapped to t by σ. This corresponds to precomputing the necessary "shortcut" of instantiating X directly into to $t\sigma\theta$ for all such concerned variables in $\textbf{dom}(\theta)$. Note that since this may possibly introduce identical pairs X/X, which must then be eliminated.
>
> The second part of the union in Expression (2.2) simply completes the resulting substitution with pairs t/Y in θ concerning those variables Y which are not affected by σ (*i.e.*, all $Y \in \textbf{dom}(\theta)$ such

that $Y \notin \mathbf{dom}(\sigma)$). Indeed, these variables are taken care of in the first part in the terms mapping the variables in $\mathbf{dom}(X)$ by further instantiating by θ as need be.

These two cases clearly cover the only possibilties for variable mapping by σ and θ, and by construction in each case, this results in a finite set of term/variable pairs, thus completely specified by Expression (2.2) on all $\mathcal{V}$, when applied to any term t, has the same effect of first applying σ to t and then applying θ to the result. □

COROLLARY A.1 *Substitution composition as defined by Expression (2.2) is an associative operation; i.e., for all σ, θ, and δ in* **SUBST**$_{\mathcal{T}}$, $\sigma(\theta\delta) = (\sigma\theta)\delta$.

PROOF Let t be any term in $\mathcal{T}$, and σ, θ, and δ be three substitutions in **SUBST**$_{\mathcal{T}}$. Applying Lemma A.2 successively, we have $t(\sigma(\theta\delta)) = (t\sigma)(\theta\delta) = ((t\sigma)\theta)\delta = (t(\sigma\theta))\delta = t((\sigma\theta)\delta)$. Since both sides applied to any term are equal, this means that $\sigma(\theta\delta) = (\sigma\theta)\delta$. □

Note that, as a set of term/variable pairs, the substitution which is the identity everywhere on $\mathcal{V}$ is the empty set of pairs—which is why it is called the empty substitution and denoted as the empty set $\emptyset$. It is easy to verify that this empty substitution is also the unique identity element on **SUBST**$_{\mathcal{T}}$. Namely, for all substitution $\sigma \in$ **SUBST**$_{\mathcal{T}}$, $\sigma\emptyset = \emptyset\sigma = \sigma$ and if $\sigma\theta = \theta\sigma = \sigma$ for some $\theta \in$ **SUBST**$_{\mathcal{T}}$, then $\theta = \emptyset$. Therefore, **SUBST**$_{\mathcal{T}}$ with composition and $\emptyset$ is a monoid. Note finally that substitution composition is not commutative since in general $\sigma\theta \neq \theta\sigma$.[9] Therefore, the set **SUBST**$_{\mathcal{T}}$ with substitution composition is a non-commutative monoid.

Like all monoids, the set **SUBST**$_{\mathcal{T}}$ of substitutions inherits a relation $\preceq$ defined as follows.

DEFINITION A.15 $\sigma \preceq \theta$ *iff* $\exists\, \delta \in$ **SUBST**$_{\mathcal{T}}$ *s.t.* $\sigma = \theta\delta$.

The expression "$\sigma \preceq \theta$" is read "σ *refines* θ" or "θ *is more general than* σ."

LEMMA A.3 *The relation $\preceq$ is a preorder on the set of first-order term substitutions* **SUBST**$_{\mathcal{T}}$.

PROOF We must show that $\preceq$ is reflexive and transitive. **Reflexivity:** For any $\sigma \in$ **SUBST**$_{\mathcal{T}}$, there exists $\delta = \emptyset$ such that $\sigma = \sigma\delta$, which means by definition of $\preceq$ that $\sigma \preceq \sigma$. **Transitivity:** Assume $\sigma_1 \preceq \sigma_2$ and $\sigma_2 \preceq \sigma_3$; this means that there exist δ_1 and δ_2 such that $\sigma_1 = \sigma_2\delta_1$ and $\sigma_2 = \sigma_3\delta_2$. Replacing σ_2 by its value in the expression of σ_1, it comes as a result that $\sigma_1 = \sigma_3\delta_2\delta_1$. And so, there exists $\delta_3 = \delta_2\delta_1$ such that $\sigma_1 = \sigma_3\delta_3$; which means that $\sigma_1 \preceq \sigma_3$. □

Note that $\preceq$ is not an order relation because it is not anti-symmetric. Indeed, if we have both $\sigma \preceq \theta$ and $\theta \preceq \sigma$, this does not necessarily imply that $\sigma = \theta$. However, this defines an equivalence relation on substitutions.

LEMMA A.4 *The relation $\simeq \stackrel{def}{=} \preceq \cap \preceq^{-1}$ is an equivalence on the set of substitutions* **SUBST**$_{\mathcal{T}}$.

[9] Take for example $\sigma = \{\, a/X \,\}$ and $\theta = \{\, b/X \,\}$, for which $\sigma\theta = \{\, a/X\}$ and $\theta\sigma = \{\, b/X\}$.

PROOF Let us verify that $\simeq$ has three properties of an equivalence. **Reflexivity:** Clearly, for any $\sigma \in \textbf{SUBST}_{\mathcal{T}}$, $\sigma \simeq \sigma$ since this is equivalent to $\sigma \preceq \sigma$ and $\sigma \preceq \sigma$, which is always true since $\preceq$ is reflexive because it is a preorder. **Symmetry:** Also, for any $\sigma \in \textbf{SUBST}_{\mathcal{T}}$ and $\theta \in \textbf{SUBST}_{\mathcal{T}}$, if $\sigma \simeq \theta$, this is equivalent by definition to $\sigma \preceq \theta$ and $\theta \preceq \sigma$; which is also equivalent to $\theta \simeq \sigma$. Therefore, $\simeq$ is symmetric. **Transitivity:** Let us now assume that (1) $\sigma \simeq \theta$ and (2) $\theta \simeq \delta$. This implies in particular, by definition of $\simeq$ and $\simeq$: (1) ($\sigma \preceq \theta$ and $\theta \preceq \delta$), which by transitivity of $\preceq$ implies $\sigma \preceq \delta$; and (2) ($\delta \preceq \theta$ and $\theta \preceq \sigma$); which by transitivity of $\preceq$ implies $\delta \preceq \sigma$. Hence, we have both $\sigma \preceq \delta$ and $\delta \preceq \sigma$, which is equivalent to $\sigma \simeq \delta$. Therefore, $\simeq$ is transitive. □

DEFINITION A.16 *A variable renaming ρ is a substitution in $\textbf{SUBST}_{\mathcal{T}} \cap (\mathcal{V} \to \mathcal{V})$ that is injective. That is,*

- $\rho = \{X'_i/X_i\}_{i=1}^{n}$ *with* $X_i \in \mathcal{V}$ *and* $X'_i \in \mathcal{V}$*; and,*
- *if* $X_i \neq X_j$ *then* $X'_i \neq X'_j$*, for any* $i, j = 1, \ldots, n$ *such that* $i \neq j$*.*

COROLLARY A.2 *If both $\sigma \preceq \theta$ and $\theta \preceq \sigma$, this entails that σ and θ are equal up to a renaming of their variables. Namely, $\exists\, \rho : \mathcal{V} \to \mathcal{V}$ bijective such that $\theta = \rho\sigma$ and $\sigma = \rho^{-1}\theta$.*

PROOF If $\sigma \preceq \theta$ and $\theta \preceq \sigma$ then, by definition, there exist two substitutions ρ and ρ' such that $\sigma = \theta\rho$ and $\theta = \sigma\rho'$. In other words:

$$\begin{cases} \sigma = \sigma\,\rho\,\rho', \\ \theta = \theta\,\rho'\rho; \end{cases} \text{ which is equivalent to: } \begin{cases} \rho\,\rho' = \emptyset, \\ \rho'\rho = \emptyset; \end{cases} \text{ and therefore to: } \begin{cases} \rho = \rho'^{-1}, \\ \rho' = \rho^{-1}. \end{cases}$$

Note also that since ρ and ρ' are mutual inverses on $\mathcal{V}$, it must be that ρ and ρ' are injective. This follows from the axiom of functionality for ρ and ρ', which states that for every pair of variables X and X' in $\mathcal{V}$, if $X = X'$ then necessarily $X\rho = X'\rho$ and $X\rho' = X'\rho'$. But since ρ are mutual inverses on $\mathcal{V}$, this means that whenever $Y\rho' = Y'\rho'$ for any pair of variables Y and Y' in $\mathcal{V}$, then necessarily $Y\rho'\rho = Y'\rho'\rho$; *i.e.*, $Y\emptyset = Y'\emptyset$, and thus $Y = Y'$, which means that ρ' must be injective. The same reasoning in the other direction will entail that ρ must be injective as well. Note finally that ρ is also surjective on $\mathcal{V}$, since any variable $X \in \mathcal{V}$ is such that $X\rho'\rho = X$, therefore there exists $Y = X\rho'$ such that $Y\rho = X$. The same applies to ρ' in the other direction. Therefore, ρ and ρ' are bijective inverses. □

A.5 Reynolds-Plotkin $\mathcal{FOT}$ Generalization

The two essentially identical algorithms (up to notation) for the generalization of two $\mathcal{FOT}$s given by Reynolds [104] and Plotkin [101] in the same volume are reproduced verbatim in Figure A.2 and Figure A.3. As can be seen in these figures, each describes a procedural method computing the most specific $\mathcal{FOT}$ subsuming two given $\mathcal{FOT}$s in finitely many steps by comparing them simultaneously, and generating a pair of generalizing substitutions from a fresh variable wherever they disagree being scanned from left to right, each time replacing the disagreeing terms by the new variable everywhere they both occur in each term.

(4) If A and B are CAFs beginning with the same predicate symbol, then let $Z_1, Z_2, \ldots$ be a sequence of variables which do not occur in A or B, and obtain $A \sqcup B$ by the following *Anti-unification Algorithm*:*

(a) Set the variables $\bar{A}$ to A, $\bar{B}$ to B, ζ and η to the empty substitution, and i to zero.

(b) If $\bar{A}=\bar{B}$, exit with $A \sqcup B=\bar{A}=\bar{B}$.

(c) Let k be the index of the first symbol position at which $\bar{A}$ and $\bar{B}$ differ, and let S and T be the terms which occur, beginning in the kth position, in $\bar{A}$ and $\bar{B}$ respectively.

(d) If, for some j such that $1 \leqslant j \leqslant i$, $Z_j\zeta=S$ and $Z_j\eta=T$, then alter $\bar{A}$ by replacing the occurrence of S beginning in the kth position by Z_j, alter $\bar{B}$ by replacing the occurrence of T beginning in the kth position by Z_j, and go to step (b).

(e) Otherwise, increase i by one, alter $\bar{A}$ by replacing the occurrence of S beginning in the kth position by Z_i, alter $\bar{B}$ by replacing the occurrence of T beginning in the kth position by Z_i, replace ζ by $\zeta \cup \{S/Z_i\}$, replace η by $\eta \cup \{T/Z_i\}$, and go to step (b).

* This algorithm has been discovered independently by Mr Gordon Plotkin of the University of Edinburgh.

Figure A.2: Reynolds's $\mathcal{FOT}$ "anti-unification" algorithm ([104], pages 138–139)

Let W_1, W_2 be any two compatible words. The following algorithm terminates at stage *3*, and the assertion made there is then correct.

1. Set V_i to $W_i (i=1,2)$. Set ε_i to $\varepsilon (i=1,2)$. ε is the empty substitution.

2. Try to find terms t_1, t_2 which have the same place in V_1, V_2 respectively and such that $t_1 \neq t_2$ and either t_1 and t_2 begin with different function letters or else at least one of them is a variable.

3. If there are no such t_1, t_2 then halt. V_1 is a least generalization of $\{W_1, W_2\}$ and $V_1=V_2$, $V_i\varepsilon_i=W_i (i=1,2)$.

4. Choose a variable x distinct from any in V_1 or V_2 and wherever t_1 and t_2 occur in the same place in V_1 and V_2, replace each by x.

5. Change ε_i to $\{t_i|x\}\varepsilon_i (i=1,2)$.

6. Go to *2*.

Figure A.3: Plotkin's $\mathcal{FOT}$ "least generalization" algorithm ([101], page 155)

A.6 Clause-driven $\mathcal{OSF}$-generalization

We will express this problem formally as one of deriving the most specific ψ-term t (up to variable renaming) generalizing two ψ-terms t_1 and t_2, together with two tag mappings γ_i : $\mathbf{var}(t) \mapsto$

var(t_i) such that $\gamma_i(t) = t_i$, $(i = 1, 2)$.

To view this as a constraint-solving problem, as before, a ψ-term t is first dissolved into its rooted $\mathcal{OSF}$ constraint form $\varphi(t)$. Thereupon, ψ-term generalization translates as deriving a rooted $\mathcal{OSF}$ constraint ϕ generalizing two rooted $\mathcal{OSF}$ constraints ϕ_1 and ϕ_2, together with two tag mappings γ_i : **var**$(\phi) \mapsto$ **var**(ϕ_i) such that $\gamma_i(\phi) = \phi_i$, $(i = 1, 2)$. This problem can thus be expressed declaratively as that of solving effectively a particular $\mathcal{OSF}$ constraint using appropriate $\mathcal{OSF}$ constraint normalization rules. We shall call such a rule an $\mathcal{OSF}$ generalization rule; it obeys the following pattern:

(LABEL): **RULE NAME**

$$\frac{(\textit{Prior Left Tag Map})\ \gamma_1, \quad (\textit{Prior Right Tag Map})\ \gamma_2 \ \vdash\ (\textit{Prior Constraint})\ \phi}{(\textit{Posterior Left Tag Map})\ \gamma_1', (\textit{Posterior Right Tag Map})\ \gamma_2' \ \vdash\ (\textit{Posterior Constraint})\ \phi'}$$

if (*Optional Metacondition*)

where a "*tag map*" γ is a set of pairs of $\mathcal{OSF}$ tags, each of the form $Y = X$, and denotes a mapping of tags being inferred whereby $\gamma(Y) \stackrel{\text{def}}{=} X$.[10] These tag maps appearing in such rules will be also referred to as "*contexts*." Seen as a "fraction notation" (such as $\frac{N}{D}$) each rule transforms the antecedent (the "numerator;" *i.e.*, N) into the consequent (the "denominator;" *i.e.*, D), whenever the prior patterns match and the (optional) metacondition holds.

The constraint ϕ is of the form $\phi' \left[\phi_1 \ \| \ \phi_2\right]$, where each of ϕ' and ϕ_i, $i = 1, 2$, is a (possibly empty) conjunction of basic $\mathcal{OSF}$ constraints in solved form; moreover, ϕ' is rooted (*i.e.*, it is a dissolved ψ-term). The rules for performing $\mathcal{OSF}$ term generalization are given in Figure A.4. These rules are explained informally as follows.

- Rule **SORT INDUCTION** — when both sides of the disjunction "$\_\|\_$" in the prior constraint contain each an atomic constraint of the form $X_i : s_i$ where tag X is bound to each tag X_i in both prior contexts for $i = 1, 2$, the pattern $X : s_1 \vee s_2$ may then be inferred as a generalizing pattern and conjoined to ϕ. This may be safely done only if the constraint ϕ to generalize does not already contain a constraint of the form $X : s$ for the tag X and some sort s, as specified by the side condition (otherwise, this rule would keep being applied on the same pattern).

- Rule **FEATURE INDUCTION** — when both sides of the disjunction "$\_ \ \| \ \_$" in the prior constraint contain each an atomic constraint of the form $X_i.f \doteq Y_i$ for the same feature symbol f where a same tag X is bound to each tag X_i in both prior contexts for $i = 1, 2$, the pattern $X.f \doteq Y$ may then be inferred as a generalizing pattern and conjoined to ϕ, where Y is a new tag now bound to each tag Y_i in each posterior contexts, for $i = 1, 2$. The side condition also stipulates that, for this rule to be applicable, there should not already exist any tag bound to Y_i in either prior contexts, for $i = 1, 2$. For when such is the case, this means that there is an "orphan" feature constraint for f out of either X_1 or X_2 in one

[10] Operationally, $Y = X$ reads "*tag Y is substituted for by tag X*," or equivalently, when tags as seen as logical variables: "*X is bound to Y*." This amounts to replacing X with Y everywhere X occurs besides the lefthand side of the equational constraint $X \doteq Y$. See, *e.g.*, in Figure 3.5: *viz.*, γ_1 : **TagSet**$(t) \mapsto$ **TagSet**(t_1) and γ_2 : **TagSet**$(t) \mapsto$ **TagSet**(t_2), where $t =$ **lub**(t_1, t_2).

(SI): **SORT INDUCTION**

$$\frac{\{\, X_1/X \,\} \cup \gamma_1, \{\, X_2/X \,\} \cup \gamma_2 \;\vdash\; \phi \left[(X_1 : s_1 \;\&\; \phi_1) \;\|\; (X_2 : s_2 \;\&\; \phi_2)\right]}{\{\, X_1/X \,\} \cup \gamma_1, \{\, X_2/X \,\} \cup \gamma_2 \;\vdash\; \phi \;\&\; X : s_1 \vee s_2 \left[(X_1 : s_1 \;\&\; \phi_1) \;\|\; (X_2 : s_2 \;\&\; \phi_2)\right]}$$

if $\neg\exists\, s$ **s.t.** $X : s \in \phi$

(FI): **FEATURE INDUCTION**

$$\frac{\{\, X_1/X \,\} \cup \gamma_1, \qquad \{\, X_2/X \,\} \cup \gamma_2 \;\vdash\; \phi \left[(X_1.f \doteq Y_1 \;\&\; \phi_1) \;\|\; (X_2.f \doteq Y_2 \;\&\; \phi_2)\right]}{\{\, X_1/X, Y_1/Y \,\} \cup \gamma_1, \{\, X_2/X, Y_2/Y \,\} \cup \gamma_2 \;\vdash\; \phi \;\&\; X.f \doteq Y \left[(X_1.f \doteq Y_1 \;\&\; \phi_1) \;\|\; (X_2.f \doteq Y_2 \;\&\; \phi_2)\right]}$$

if $\neg\exists\, Z$ s.t. $Y_1/Z \in \{\, X_1/X \,\} \cup \gamma_1$ **and** $\neg\exists\, Z$ s.t. $Y_2/Z \in \{\, X_2/X \,\} \cup \gamma_2$; **and where** Y is a new tag

(CI): **COREFERENCE INDUCTION**

$$\frac{\{\, X_1/X, Y_1/Y \,\} \cup \gamma_1, \{\, X_2/X, Y_2/Y \,\} \cup \gamma_2 \;\vdash\; \phi \left[(X_1.f \doteq Y_1 \;\&\; \phi_1) \;\|\; (X_2.f \doteq Y_2 \;\&\; \phi_2)\right]}{\{\, X_1/X, Y_1/Y \,\} \cup \gamma_1, \{\, X_2/X, Y_2/Y \,\} \cup \gamma_2 \;\vdash\; \phi \;\&\; X.f \doteq Y \left[(X_1.f \doteq Y_1 \;\&\; \phi_1) \;\|\; (X_2.f \doteq Y_2 \;\&\; \phi_2)\right]}$$

Figure A.4: Constraint normalization rules for $\mathcal{OSF}$ generalization [21]

but not the other disjunct, which may not be generalized as a common property since not present in both disjuncts.

- Rule **COREFERENCE INDUCTION** — this rule is almost identical to the previous one except that it is precisely when its side condition does not hold because there is already a tag Y simultaneously bound to X_i in each prior contexts for $i = 1, 2$. In this case, the inference may be made to materialize $X.f \doteq Y$ in the induced constraint from the fact that the same feature exists in both sides of the disjunct between corresponding tags in both prior contexts.

The initial constraint to be normalized in order to use these rules to infer the most specific generalizer (*i.e.*, the least upper bound) of two ψ-terms $\mathbf{t}_1$ and $\mathbf{t}_2$ whose tags have been renamed apart is:

$$\{\, \mathbf{ROOT}(\mathbf{t}_1)/X \,\}, \{\, \mathbf{ROOT}(\mathbf{t}_2)/X \,\} \;\vdash\; \left[\varphi(\mathbf{t}_1) \;\|\; \varphi(\mathbf{t}_2)\right] \tag{A.27}$$

where X is a new tag symbol generated for $\mathbf{ROOT}(\mathbf{lub}(\mathbf{t}_1, \mathbf{t}_2))$; *i.e.*,

$$\mathbf{ROOT}(\mathbf{lub}(\mathbf{t}_1, \mathbf{t}_2)) \stackrel{\text{def}}{=} X. \tag{A.28}$$

Then, starting with initial tag maps:

$$\gamma_i(X) \stackrel{\text{def}}{=} \mathbf{ROOT}(\mathbf{t}_i), \quad \text{for } i = 1, 2, \tag{A.29}$$

we proceed applying the rules of Figure A.4 to the $\mathcal{OSF}$ generalization constraint formula (A.27) until none applies. This always terminates, resulting in an expression of the form: $\phi \left[\phi_1 \;\|\; \phi_2\right]$

where ϕ and ϕ_i, $(i = 1, 2)$, are (possibly empty) conjunctive $\mathcal{OSF}$ clauses in solved form; moreover ϕ is rooted (*i.e.*, it is a dissolved ψ-term). Whatever remains, if anything, in the disjunction (*i.e.*, $\left[\phi_1 \parallel \phi_2\right]$) is a pair of differential $\mathcal{OSF}$ constraints such that:

$$\gamma_1(\phi) \,\&\, \phi_1 \simeq \varphi(\mathbf{t}_1) \tag{A.30}$$

$$\gamma_2(\phi) \,\&\, \phi_2 \simeq \varphi(\mathbf{t}_2) \tag{A.31}$$

and:

$$\phi \simeq \varphi(\mathbf{lub}(\mathbf{t}_1, \mathbf{t}_2)) \tag{A.32}$$

where "$\simeq$" means "*equal up to a consistent tag renaming*."

For a detailed rule-application trace illustrating how this constraint normalization computes the two endomorphic mappings realizing the **lub** of two $\mathcal{OSF}$ terms, see Appendix B.1, Example B.3, Page 134.

A.7 $\mathcal{OSF}$-term tag renamimg

Since unification of two ψ-terms is up to tag renaming, it actually computes a **glb** not in the set of ψ-terms, but in the quotient lattice $\Psi/_{\cong}$ of sets of congruent ψ-terms, where two ψ-terms are congruent whenever they are isomorphic. This means that ψ-terms are congruent when they are two rooted sorted $\mathcal{OSF}$ graphs with a bijection between their respective sets of tags, each pair of bijective tags bearing the same sort in their respective ψ-terms. Renaming the tags of a ψ-term simply consists of defining such a sort-preserving bijection between the tag names of one term to those of another, thus defining an $\mathcal{OSF}$ isomorphism. We are then free to replace all tags of a ψ-term with the new ones given by the bijective tag mapping, the result will remain in the same tag-renaming congruence class.

Therefore, in order to provide the new ψ-term resulting from the unification of two ψ-terms its own independent set of variables, it is sufficient to replace $\mathcal{OSF}$ unification rule **FEATURE FUNCTIONALITY** of Figure 3.6 with the rule **TAG-RENAMING FEATURE FUNCTIONALITY** of Figure A.5. This rule systematically introduces a new tag for any feature that yields two distinct tags and generates to new equations binding the new tag to each of the two tags in order to eliminate these to be replaced with the new tag by the **TAG ELIMINATION** of Figure 3.6.

TAG-RENAMING FEATURE FUNCTIONALITY:

$$\frac{\phi \,\&\, X.f \doteq Y \,\&\, X.f \doteq Y'}{\phi \,\&\, X.f \doteq Z \,\&\, Z \doteq Y \,\&\, Z \doteq Y'}\ [Z \text{ new}]$$

Figure A.5: Tag-renaming feature functionality for $\mathcal{OSF}$ unification by normalization

In this way, we can ensure that tags in the original ψ-terms being unified may only be bound to newly introduced tags, thus allowing to define an endomorphic mapping γ as follows. If an

$\mathcal{OSF}$ constraint ϕ in solved form that is not $\mathfrak{false}$ contains a tag equality constraint $X \doteq Y$, then necessarily Y occurs nowhere else in ϕ, and we can define the mapping $\gamma : \mathcal{V} \to \mathcal{V}$ with:

$$\gamma(Y) \stackrel{\text{def}}{=} \begin{cases} X & \text{if } X \doteq Y \in \phi; \\ Y & \text{otherwise.} \end{cases} \tag{A.33}$$

This mapping is akin to "binding" a tag X to a tag $\gamma(X)$.

THEOREM A.4 *The mapping γ defined by Equation A.33 derived from an $\mathcal{OSF}$ constraint in solved form ϕ which is the $\mathcal{OSF}$ normal form of the $\mathcal{OSF}$ constraint:*

$$\hat{\phi} \stackrel{\text{def}}{=} X \doteq \mathbf{ROOT}(t_1) \,\&\, X \doteq \mathbf{ROOT}(t_2) \,\&\, \varphi(t_1) \,\&\, \varphi(t_2) \tag{A.34}$$

where X is a new variable, is such that γ defines an $\mathcal{OSF}$ endormorphic mapping whereby $\gamma(\hat{\phi}) = \phi$.

PROOF The mapping γ defined by Equation A.33 is necessarily functional—*i.e.*, $X = X' \to \gamma(X) = \gamma(X')$—because otherwise this would entail that there are two constraints $X \doteq X'$ and $X \doteq X''$ in ϕ, for some tags X' and X'' such that $X' = X''$. However, if this was the case, then Rule **TAG ELIMINATION** would eliminate either X' for X'' or X' for X'' from anywhere else besides one or the other of these two equality constraints, thereby contradicting the assumption that ϕ is in normal form. Therefore γ must be functional. Moreover,

1. repeated applications of Rule **SORT INTERSECTION** can only decrease the sort of any tag so that eventually $X : s \in \hat{\phi} \to \gamma(X) : s' \in \phi$ with $s' \preceq s$;
2. repeated applications of Rule **FEATURE FUNCTIONALITY** enforce eventual equality in the solved form ϕ of two image tags of a same feature out of any tag, always ending up eliminated for the other by application of Rule **TAG ELIMINATION**—in other words:

$$X.f \doteq Y \in \hat{\phi} \;\to\; \gamma(X).f \doteq \gamma(Y) \in \phi.$$

By definition, these are the two properties required for γ to be an $\mathcal{OSF}$ endomorphism. □

Appendix B

Version of January 8, 2019

$\mathcal{OSF}$ Examples and Extensions

In this appendix, we provide examples and extensions of $\mathcal{OSF}$ constraints.

- Appendix B.1 gives detailed examples of $\mathcal{OSF}$ lattice operations.
- Appendix B.2 gives three useful additional decidable $\mathcal{OSF}$ constraints: partial features, extensional (*i.e.*, element-denoting) sort symbols, and aggregation. The first and second of these constraints convey implicit additional axioms that $\mathcal{FOT}$s verify as $\mathcal{OSF}$ terms.
- Appendix B.3 shows explicitly how a $\mathcal{FOT}$ is a special case of $\mathcal{OSF}$ term using partial features and extensional sorts.

B.1 Examples of $\mathcal{OSF}$ Lattice Operations

In this chapter, we will give detailed examples illustrating (crisp) $\mathcal{OSF}$ unification and generalization using the sort taxonomy in Figure B.1, which corresponds to a possible description of a partial school population with the following subconcept and instance declarations.

- set-denoting subconcepts:
  - a `student` is a `person`
  - an `employee` is a `person`
  - a `staff` is an `employee`
  - a `faculty` is an `employee`
  - an `intern` is a `student`
  - an `intern` is a `staff`
- individual-denoting subconcepts:
  - `nassim` is a `student`

- **jayd** is a **student**
- **elies** is a **student**

- **ali** is an **intern**
- **hanaan** is an **intern**

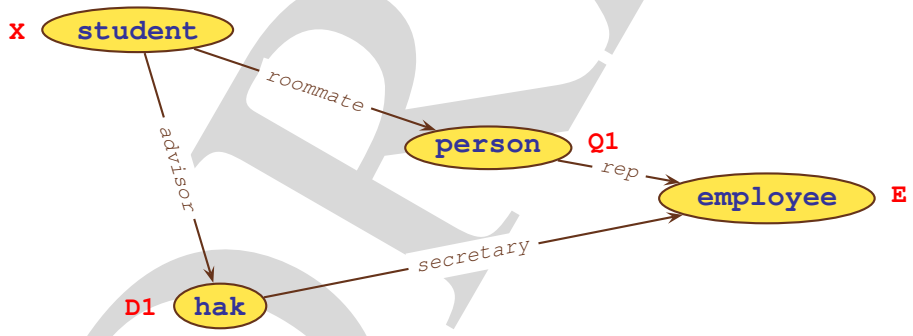
- **javier** is a **staff**
- **eta** is a **staff**

- **hak** is a **faculty**
- **hussein** is a **faculty**
- **fatima** is a **faculty**

Example B.1 $\mathcal{OSF}$ **lattice operations** — Consider for example the ψ-term $\mathbf{t}_1$:

$$\begin{aligned}\mathbf{t}_1 \;=\; & \mathtt{X{:}student} \\ & (\,\mathit{roommate} \rightarrow \mathtt{person}(\mathit{rep} \rightarrow \mathtt{E{:}employee}) \\ & ,\,\mathit{advisor} \rightarrow \mathtt{hak}(\mathit{secretary} \rightarrow \mathtt{E})\,)\end{aligned}$$

corresponding to the $\mathcal{OSF}$-graph:[1]



and the ψ-term $\mathbf{t}_2$:

$$\begin{aligned}\mathbf{t}_2 \;=\; & \mathtt{Y{:}employee} \\ & (\,\mathit{advisor} \rightarrow \mathtt{hak}(\mathit{assistant} \rightarrow \mathtt{A}) \\ & ,\,\mathit{roommate} \rightarrow \mathtt{S{:}student}(\mathit{rep} \rightarrow \mathtt{S}) \\ & ,\,\mathit{helper} \rightarrow \mathtt{ali}(\mathit{spouse} \rightarrow \mathtt{A})\,)\end{aligned}$$

[1] In this and all similar examples, we shall generate new tag names that do not occur in the ψ-terms equivalent to $\mathcal{OSF}$-graphs for root nodes of tagless subterms—*e.g.*, **Q1** and **D1** in this example.

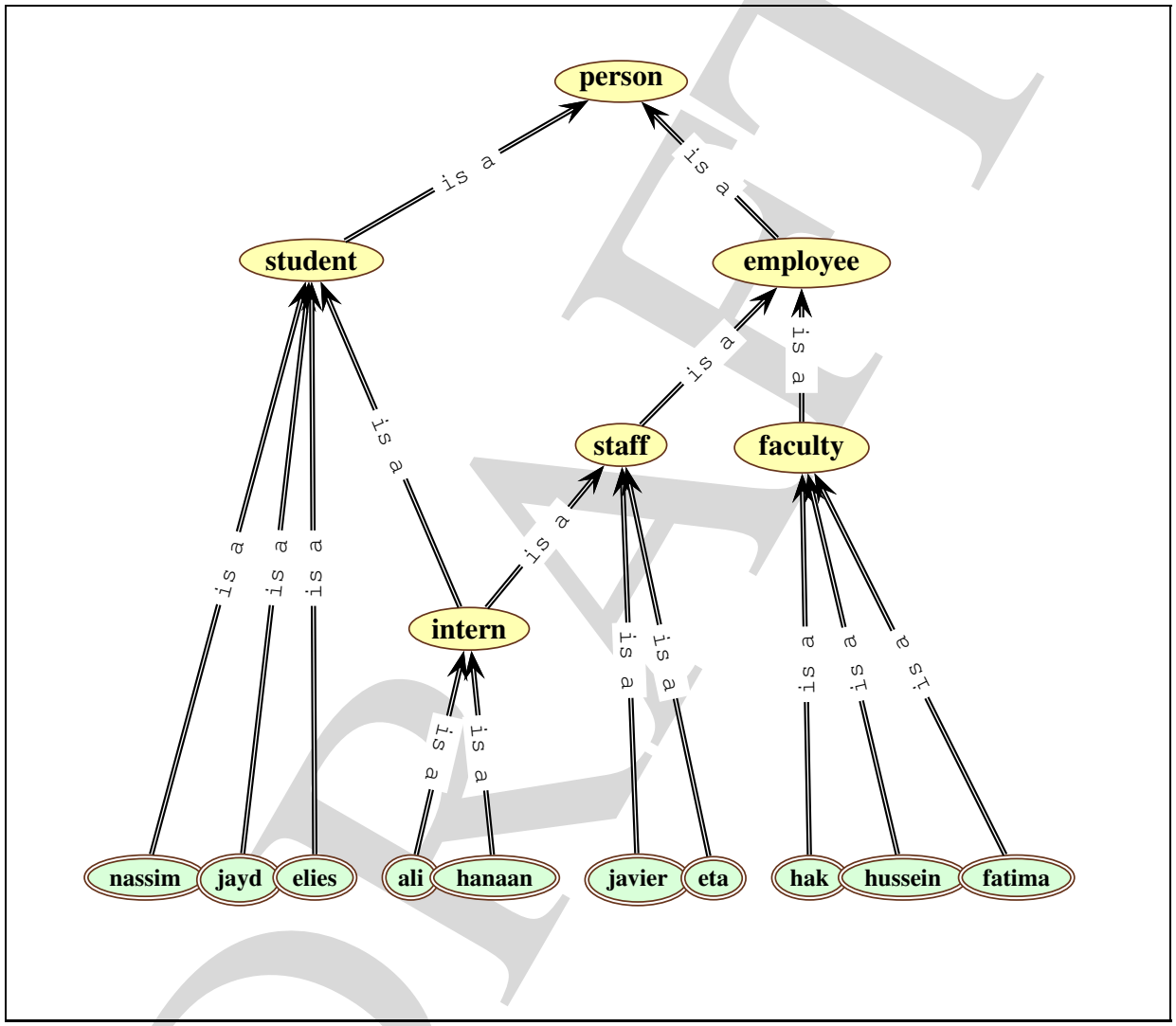
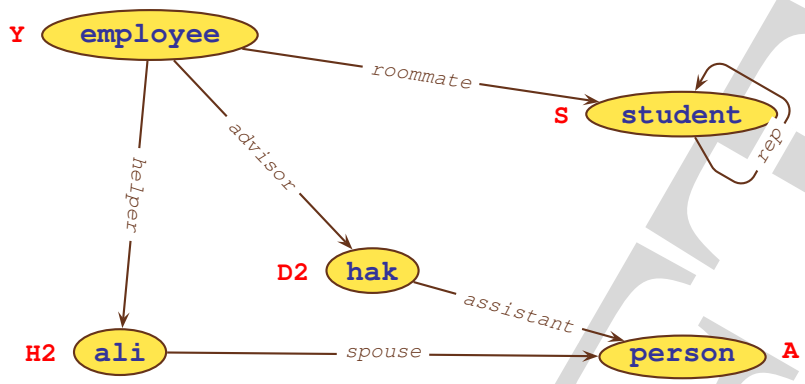


Figure B.1: “School example” concept taxonomy

corresponding to the $\mathcal{OSF}$-graph:

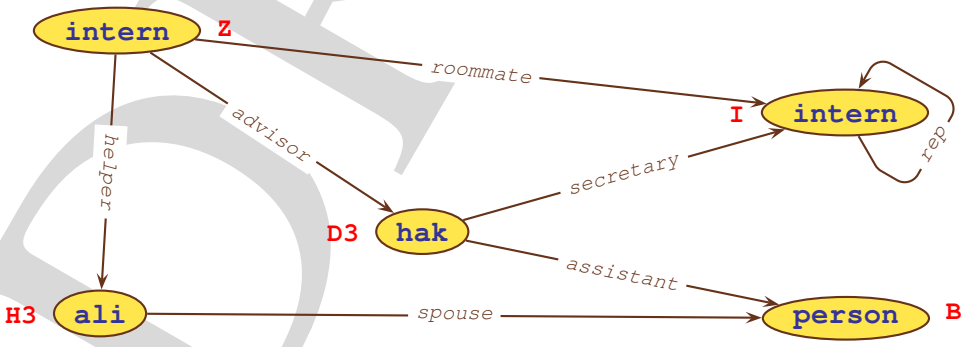


in the context of the sort partial order ("concept taxonomy") shown in Figure B.1.

Endomorphic mappings γ, γ_1, and γ_2, can be computed to exhibit the lattice structure of $\mathcal{OSF}$ terms. Given the two terms $\mathbf{t}_1$ and $\mathbf{t}_2$ shown above, their greatest lower bound is the ψ-term $\underline{\mathbf{t}}$:

$$
\begin{aligned}
\underline{\mathbf{t}} \;=\; & \mathtt{Z{:}intern} \\
& (\,\mathit{advisor} \rightarrow \mathtt{hak}(\mathit{assistant} \rightarrow \mathtt{B}, \\
& \qquad\qquad \mathit{secretary} \rightarrow \mathtt{I}) \\
& ,\,\mathit{helper} \rightarrow \mathtt{ali}(\mathit{spouse} \rightarrow \mathtt{B}) \\
& ,\,\mathit{roommate} \rightarrow \mathtt{I{:}intern}(\mathit{rep} \rightarrow \mathtt{I}))
\end{aligned}
$$

corresponding to the graph:



given by their $\mathcal{OSF}$ unification realized by the endomorphic mapping γ such that:

$$\gamma(\mathbf{t}_1) = \gamma(\mathbf{t}_2) = \underline{\mathbf{t}}.$$

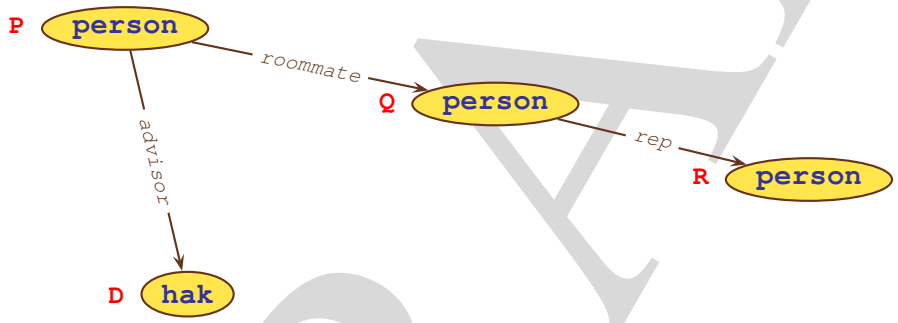
It corresponds to the tag mapping:

$$\begin{array}{ll}\gamma(\mathtt{X}) &= \mathtt{Z},\\ \gamma(\mathtt{Q1}) &= \mathtt{I},\\ \gamma(\mathtt{E}) &= \mathtt{I},\\ \gamma(\mathtt{D1}) &= \mathtt{D3},\\ \\ \gamma(\mathtt{Y}) &= \mathtt{Z},\\ \gamma(\mathtt{S}) &= \mathtt{I},\\ \gamma(\mathtt{D2}) &= \mathtt{D3},\\ \gamma(\mathtt{H2}) &= \mathtt{H3},\\ \gamma(\mathtt{A}) &= \mathtt{B}.\end{array}$$

Dually, their least upper bound is the ψ-term $\mathbf{t}$:

$$\begin{array}{l}\mathbf{t} \;=\; \mathtt{P}\!:\!\mathbf{person}\\ \qquad\quad (\;\mathit{roommate} \rightarrow \mathbf{person}(\mathit{rep} \rightarrow \mathbf{person})\\ \qquad\quad ,\;\mathit{advisor} \rightarrow \mathbf{hak})\end{array}$$

corresponding to the $\mathcal{OSF}$-graph:



given by their $\mathcal{OSF}$ generalization realized by the two endomorphic mappings $\langle\gamma_1, \gamma_2\rangle$ whereby:

$$\begin{array}{l}\gamma_1(\mathbf{t}) = \mathbf{t}_1,\\ \gamma_2(\mathbf{t}) = \mathbf{t}_2.\end{array}$$

They correspond to the tag mappings γ_1 and γ_2 defined as:

$$\begin{array}{ll}\gamma_1(\mathtt{P}) = \mathtt{X}, & \gamma_2(\mathtt{P}) = \mathtt{Y},\\ \gamma_1(\mathtt{Q}) = \mathtt{Q1}, & \gamma_2(\mathtt{Q}) = \mathtt{S},\\ \gamma_1(\mathtt{R}) = \mathtt{E}, & \gamma_2(\mathtt{R}) = \mathtt{S},\\ \gamma_1(\mathtt{D}) = \mathtt{D1}; & \gamma_2(\mathtt{D}) = \mathtt{D2}.\end{array}$$

Starting with:

$$\{\,\mathtt{P} = \textsc{root}(\mathbf{t}_1)\,\}, \{\,\mathtt{P} = \textsc{root}(\mathbf{t}_2)\,\} \;\vdash\; \left[\varphi(\mathbf{t}_1) \;\|\; \varphi(\mathbf{t}_2)\right]$$

where $\mathtt{P}$ is a new tag symbol for $\textsc{root}(\mathbf{lub}(\mathbf{t}_1, \mathbf{t}_2))$, we proceed applying the rules of Figure A.4 until none applies.

We next give the detailed rule application traces for both unification and generalization of $\mathcal{OSF}$ terms.

Example B.2 $\mathcal{OSF}$ **unification** — Here is a step-by-step trace of constraint normalization computing the unification $\mathbf{glb}(\mathbf{t}_1, \mathbf{t}_2)$ of the $\mathcal{OSF}$ terms $\mathbf{t}_1$ and $\mathbf{t}_2$ defined in Example B.1. The ψ-term $\underline{\mathbf{t}} = \mathbf{glb}(\mathbf{t}_1, \mathbf{t}_2)$ together with the endomorphism $\gamma : \mathbf{TagSet}(\mathbf{t}_1) \cup \mathbf{TagSet}(\mathbf{t}_1) \mapsto \mathbf{TagSet}(\underline{\mathbf{t}})$ are computed using the $\mathcal{OSF}$ unification rules of Figure 3.6 as follows.

We start by dissolving $\mathbf{t}_1$ and $\mathbf{t}_2$:

- $\varphi(\mathbf{t}_1) = \mathtt{X} : \mathbf{student} \;\&\; \mathtt{X} \,.\, \mathit{roommate} \doteq \mathtt{Q1} \;\&\; \mathtt{X} \,.\, \mathit{advisor} \doteq \mathtt{D1} \;\&\; \mathtt{Q1} : \mathbf{person} \;\&\; \mathtt{Q1} \,.\, \mathit{rep} \doteq \mathtt{E} \;\&\; \mathtt{D1} : \mathbf{hak} \;\&\; \mathtt{D1} \,.\, \mathit{secretary} \doteq \mathtt{E} \;\&\; \mathtt{E} : \mathbf{employee}$
- $\varphi(\mathbf{t}_2) = \mathtt{Y} : \mathbf{employee} \;\&\; \mathtt{Y} \,.\, \mathit{roommate} \doteq \mathtt{S} \;\&\; \mathtt{Y} \,.\, \mathit{advisor} \doteq \mathtt{D2} \;\&\; \mathtt{Y} \,.\, \mathit{helper} \doteq \mathtt{H2} \;\&\; \mathtt{S} : \mathbf{student} \;\&\; \mathtt{S} \,.\, \mathit{rep} \doteq \mathtt{S} \;\&\; \mathtt{D2} : \mathbf{hak} \;\&\; \mathtt{D2} \,.\, \mathit{assistant} \doteq \mathtt{A} \;\&\; \mathtt{H2} : \mathbf{ali} \;\&\; \mathtt{H2} \,.\, \mathit{spouse} \doteq \mathtt{A} \;\&\; \mathtt{A} : \mathbf{person}$

Then, we keep applying any applicable $\mathcal{OSF}$ unification rule of Figure 3.6 in any order until none applies to the following initial constraint:

$$\mathtt{Z} \doteq \mathtt{X} \;\&\; \mathtt{Z} \doteq \mathtt{Y} \;\&\; \varphi(\mathbf{t}_1) \;\&\; \varphi(\mathbf{t}_2). \tag{B.1}$$

where $\mathtt{Z}$ is a newly generated tag for $\textsc{root}(\mathbf{glb}(\mathbf{t}_1, \mathbf{t}_2))$. This normalization proceeds as follows:[2]

1. apply Rule **Tag Elimination** (twice—once eliminating $\mathtt{X}$ replacing with $\mathtt{Z}$, and the other eliminating $\mathtt{Y}$ also replacing it with $\mathtt{Z}$):[3]
   - $\{\; \underline{\mathtt{Z} \doteq \mathtt{X}} \;\&\; \underline{\mathtt{Z} \doteq \mathtt{Y}}$ $\&$ $\mathtt{X} : \mathbf{student} \;\&\; \mathtt{X} \,.\, \mathit{roommate} \doteq \mathtt{Q1} \;\&\; \mathtt{X} \,.\, \mathit{advisor} \doteq \mathtt{D1} \;\&\; \mathtt{Q1} : \mathbf{person} \;\&\; \mathtt{Q1} \,.\, \mathit{rep} \doteq \mathtt{E} \;\&\; \mathtt{D1} : \mathbf{hak} \;\&\; \mathtt{D1} \,.\, \mathit{secretary} \doteq \mathtt{E} \;\&\; \mathtt{E} : \mathbf{employee}$ $\&$ $\mathtt{Y} : \mathbf{employee} \;\&\; \mathtt{Y} \,.\, \mathit{roommate} \doteq \mathtt{S} \;\&\; \mathtt{Y} \,.\, \mathit{advisor} \doteq \mathtt{D2} \;\&\; \mathtt{Y} \,.\, \mathit{helper} \doteq \mathtt{H2} \;\&\; \mathtt{S} : \mathbf{student} \;\&\; \mathtt{S} \,.\, \mathit{rep} \doteq \mathtt{S} \;\&\; \mathtt{D2} : \mathbf{hak} \;\&\; \mathtt{D2} \,.\, \mathit{assistant} \doteq \mathtt{A} \;\&\; \mathtt{H2} : \mathbf{ali} \;\&\; \mathtt{H2} \,.\, \mathit{spouse} \doteq \mathtt{A} \;\&\; \mathtt{A} : \mathbf{person} \;\}$;
   - endomorphic mapping: $\gamma(\mathtt{X}) \stackrel{\text{def}}{=} \mathtt{Z}$, $\gamma(\mathtt{Y}) \stackrel{\text{def}}{=} \mathtt{Z}$;
2. apply Rule **Sort Intersection**:
   - $\{\; \underline{\mathtt{Z} : \mathbf{student}} \;\&\; \mathtt{Z} \,.\, \mathit{roommate} \doteq \mathtt{Q1} \;\&\; \mathtt{Z} \,.\, \mathit{advisor} \doteq \mathtt{D1} \;\&\; \mathtt{Q1} : \mathbf{person} \;\&\; \mathtt{Q1} \,.\, \mathit{rep} \doteq \mathtt{E} \;\&\; \mathtt{D1} : \mathbf{hak} \;\&\; \mathtt{D1} \,.\, \mathit{secretary} \doteq \mathtt{E} \;\&\; \mathtt{E} : \mathbf{employee}$ $\&$ $\underline{\mathtt{Z} : \mathbf{employee}} \;\&\; \mathtt{Z} \,.\, \mathit{roommate} \doteq \mathtt{S} \;\&\; \mathtt{Z} \,.\, \mathit{advisor} \doteq \mathtt{D2} \;\&\; \mathtt{Z} \,.\, \mathit{helper} \doteq \mathtt{H2} \;\&\; \mathtt{S} : \mathbf{student} \;\&\; \mathtt{S} \,.\, \mathit{rep} \doteq \mathtt{S} \;\&\; \mathtt{D2} : \mathbf{hak} \;\&\; \mathtt{D2} \,.\, \mathit{assistant} \doteq \mathtt{A} \;\&\; \mathtt{H2} : \mathbf{ali} \;\&\; \mathtt{H2} \,.\, \mathit{spouse} \doteq \mathtt{A} \;\&\; \mathtt{A} : \mathbf{person} \;\}$;

[2] We shall underline the parts of a constraint matching a unification rule prior constraint pattern, which rule is then applied next in the application trace—which is just one among all equivalent other possible non-deterministic traces for what concerns any particular order of rule application.

[3] Also, since they are recorded alongside with the tag map γ, we will also erase the corresponding tag equation in the constraint when applying Rule **Tag Elimination**.

- endomorphic mapping: $\gamma(\mathtt{X}) = \mathtt{Z}$, $\gamma(\mathtt{Y}) = \mathtt{Z}$;

3. apply Rule **RENAMING TAGS APART**:

   - $\{\ \mathtt{Z} : \mathtt{intern}$
     $\&$
     $\underline{\mathtt{Z}\,.\,\mathit{roommate} \doteq \mathtt{Q1}}\ \&\ \mathtt{Z}\,.\,\mathit{advisor} \doteq \mathtt{D1}\ \&\ \mathtt{Q1} : \mathtt{person}\ \&\ \mathtt{Q1}\,.\,\mathit{rep} \doteq \mathtt{E}\ \&$
     $\mathtt{D1} : \mathtt{hak}\ \&\ \mathtt{D1}\,.\,\mathit{secretary} \doteq \mathtt{E}\ \&\ \mathtt{E} : \mathtt{employee}$
     $\&$
     $\underline{\mathtt{Z}\,.\,\mathit{roommate} \doteq \mathtt{S}}\ \&\ \mathtt{Z}\,.\,\mathit{advisor} \doteq \mathtt{D2}\ \&\ \mathtt{Z}\,.\,\mathit{helper} \doteq \mathtt{H2}\ \&\ \mathtt{S} : \mathtt{student}\ \&$
     $\mathtt{S}\,.\,\mathit{rep} \doteq \mathtt{S}\ \&\ \mathtt{D2} : \mathtt{hak}\ \&\ \mathtt{D2}\,.\,\mathit{assistant} \doteq \mathtt{A}\ \&\ \mathtt{H2} : \mathtt{ali}\ \&\ \mathtt{H2}\,.\,\mathit{spouse} \doteq \mathtt{A}$
     $\&\ \mathtt{A} : \mathtt{person}\ \}$;
   - endomorphic mapping: $\gamma(\mathtt{X}) = \mathtt{Z}$, $\gamma(\mathtt{Y}) = \mathtt{Z}$;

4. apply Rule **TAG ELIMINATION**:

   - $\{\ \mathtt{Z} : \mathtt{intern}\ \&\ \mathtt{Z}\,.\,\mathit{roommate} \doteq \mathtt{I}\ \&\ \underline{\mathtt{I} \doteq \mathtt{Q1}}\ \&\ \underline{\mathtt{I} \doteq \mathtt{S}}$
     $\&$
     $\mathtt{Z}\,.\,\mathit{advisor} \doteq \mathtt{D1}\ \&\ \mathtt{Q1} : \mathtt{person}\ \&\ \mathtt{Q1}\,.\,\mathit{rep} \doteq \mathtt{E}\ \&\ \mathtt{D1} : \mathtt{hak}\ \&\ \mathtt{D1}\,.\,\mathit{secretary} \doteq \mathtt{E}$
     $\&\ \mathtt{E} : \mathtt{employee}$
     $\&$
     $\mathtt{Z}\,.\,\mathit{advisor} \doteq \mathtt{D2}\ \&\ \mathtt{Z}\,.\,\mathit{helper} \doteq \mathtt{H2}\ \&\ \mathtt{S} : \mathtt{student}\ \&\ \mathtt{S}\,.\,\mathit{rep} \doteq \mathtt{S}\ \&$
     $\mathtt{D2} : \mathtt{hak}\ \&\ \mathtt{D2}\,.\,\mathit{assistant} \doteq \mathtt{A}\ \&\ \mathtt{H2} : \mathtt{ali}\ \&\ \mathtt{H2}\,.\,\mathit{spouse} \doteq \mathtt{A}\ \&\ \mathtt{A} : \mathtt{person}$
     $\}$;
   - endomorphic mapping: $\gamma(\mathtt{X}) = \mathtt{Z}$, $\gamma(\mathtt{Y}) = \mathtt{Z}$, $\gamma(\mathtt{Q1}) \stackrel{\text{def}}{=} \mathtt{I}$, $\gamma(\mathtt{S}) \stackrel{\text{def}}{=} \mathtt{I}$;

5. apply Rule **RENAMING TAGS APART**:

   - $\{\ \mathtt{Z} : \mathtt{intern}\ \&\ \mathtt{Z}\,.\,\mathit{roommate} \doteq \mathtt{I}$
     $\&$
     $\underline{\mathtt{Z}\,.\,\mathit{advisor} \doteq \mathtt{D1}}\ \&\ \mathtt{I} : \mathtt{person}\ \&\ \mathtt{I}\,.\,\mathit{rep} \doteq \mathtt{E}\ \&\ \mathtt{D1} : \mathtt{hak}\ \&\ \mathtt{D1}\,.\,\mathit{secretary} \doteq \mathtt{E}$
     $\&\ \mathtt{E} : \mathtt{employee}$
     $\&$
     $\underline{\mathtt{Z}\,.\,\mathit{advisor} \doteq \mathtt{D2}}\ \&\ \mathtt{Z}\,.\,\mathit{helper} \doteq \mathtt{H2}\ \&\ \mathtt{I} : \mathtt{student}\ \&\ \mathtt{I}\,.\,\mathit{rep} \doteq \mathtt{I}\ \&$
     $\mathtt{D2} : \mathtt{hak}\ \&\ \mathtt{D2}\,.\,\mathit{assistant} \doteq \mathtt{A}\ \&\ \mathtt{H2} : \mathtt{ali}\ \&\ \mathtt{H2}\,.\,\mathit{spouse} \doteq \mathtt{A}\ \&\ \mathtt{A} : \mathtt{person}$
     $\}$;
   - endomorphic mapping: $\gamma(\mathtt{X}) = \mathtt{Z}$, $\gamma(\mathtt{Y}) = \mathtt{Z}$, $\gamma(\mathtt{Q1}) = \mathtt{I}$, $\gamma(\mathtt{S}) = \mathtt{I}$;

6. apply Rule **TAG ELIMINATION**:

   - $\{\ \mathtt{Z} : \mathtt{intern}\ \&\ \mathtt{Z}\,.\,\mathit{roommate} \doteq \mathtt{I}\ \&\ \mathtt{Z}\,.\,\mathit{advisor} \doteq \mathtt{D3}\ \&\ \underline{\mathtt{D3} \doteq \mathtt{D1}}\ \&\ \underline{\mathtt{D3} \doteq \mathtt{D2}}$
     $\&$
     $\mathtt{I} : \mathtt{person}\ \&\ \mathtt{I}\,.\,\mathit{rep} \doteq \mathtt{E}\ \&\ \mathtt{D1} : \mathtt{hak}\ \&\ \mathtt{D1}\,.\,\mathit{secretary} \doteq \mathtt{E}\ \&\ \mathtt{E} : \mathtt{employee}$
     $\&$
     $\mathtt{Z}\,.\,\mathit{helper} \doteq \mathtt{H2}\ \&\ \mathtt{I} : \mathtt{student}\ \&\ \mathtt{I}\,.\,\mathit{rep} \doteq \mathtt{I}\ \&\ \mathtt{D2} : \mathtt{hak}\ \&\ \mathtt{D2}\,.\,\mathit{assistant} \doteq \mathtt{A}$
     $\&\ \mathtt{H2} : \mathtt{ali}\ \&\ \mathtt{H2}\,.\,\mathit{spouse} \doteq \mathtt{A}\ \&\ \mathtt{A} : \mathtt{person}\ \}$;

- endomorphic mapping: $\gamma(\mathtt{X}) = \mathtt{Z}$, $\gamma(\mathtt{Y}) = \mathtt{Z}$, $\gamma(\mathtt{Q1}) = \mathtt{I}$, $\gamma(\mathtt{S}) = \mathtt{I}$;

7. apply Rule SORT INTERSECTION:

   - $\{\ \mathtt{Z} : \mathtt{intern}\ \&\ \mathtt{Z}\ .\ \mathit{roommate} \doteq \mathtt{I}\ \&\ \mathtt{Z}\ .\ \mathit{advisor} \doteq \mathtt{D3}$
     $\&$
     $\underline{\mathtt{I} : \mathtt{person}}\ \&\ \mathtt{I}\ .\ \mathit{rep} \doteq \mathtt{E}\ \&\ \mathtt{D3} : \mathtt{hak}\ \&\ \mathtt{D3}\ .\ \mathit{secretary} \doteq \mathtt{E}\ \&\ \mathtt{E} : \mathtt{employee}$
     $\&$
     $\mathtt{Z}\ .\ \mathit{helper} \doteq \mathtt{H2}\ \&\ \underline{\mathtt{I} : \mathtt{student}}\ \&\ \mathtt{I}\ .\ \mathit{rep} \doteq \mathtt{I}\ \&\ \mathtt{D3} : \mathtt{hak}\ \&\ \mathtt{D3}\ .\ \mathit{assistant} \doteq \mathtt{A}$
     $\&\ \mathtt{H2} : \mathtt{ali}\ \&\ \mathtt{H2}\ .\ \mathit{spouse} \doteq \mathtt{A}\ \&\ \mathtt{A} : \mathtt{person}\ \}$;
   - endomorphic mapping: $\gamma(\mathtt{X}) = \mathtt{Z}$, $\gamma(\mathtt{Y}) = \mathtt{Z}$, $\gamma(\mathtt{Q1}) = \mathtt{I}$, $\gamma(\mathtt{S}) = \mathtt{I}$, $\gamma(\mathtt{D1}) \stackrel{\text{def}}{=} \mathtt{D3}$, $\gamma(\mathtt{D2}) \stackrel{\text{def}}{=} \mathtt{D3}$;

8. apply Rule FEATURE FUNCTIONALITY:

   - $\{\ \mathtt{Z} : \mathtt{intern}\ \&\ \mathtt{Z}\ .\ \mathit{roommate} \doteq \mathtt{I}\ \&\ \mathtt{Z}\ .\ \mathit{advisor} \doteq \mathtt{D3}\ \&\ \mathtt{I} : \mathtt{student}$
     $\&$
     $\underline{\mathtt{I}\ .\ \mathit{rep} \doteq \mathtt{E}}\ \&\ \mathtt{D3} : \mathtt{hak}\ \&\ \mathtt{D3}\ .\ \mathit{secretary} \doteq \mathtt{E}\ \&\ \mathtt{E} : \mathtt{employee}$
     $\&$
     $\mathtt{Z}\ .\ \mathit{helper} \doteq \mathtt{H2}\ \&\ \underline{\mathtt{I}\ .\ \mathit{rep} \doteq \mathtt{I}}\ \&\ \mathtt{D3} : \mathtt{hak}\ \&\ \mathtt{D3}\ .\ \mathit{assistant} \doteq \mathtt{A}\ \&\ \mathtt{H2} : \mathtt{ali}$
     $\&\ \mathtt{H2}\ .\ \mathit{spouse} \doteq \mathtt{A}\ \&\ \mathtt{A} : \mathtt{person}\ \}$;
   - endomorphic mapping: $\gamma(\mathtt{X}) = \mathtt{Z}$, $\gamma(\mathtt{Y}) = \mathtt{Z}$, $\gamma(\mathtt{Q1}) = \mathtt{I}$, $\gamma(\mathtt{S}) = \mathtt{I}$, $\gamma(\mathtt{D1}) = \mathtt{D3}$, $\gamma(\mathtt{D2}) = \mathtt{D3}$;

9. apply Rule TAG ELIMINATION:

   - $\{\ \mathtt{Z} : \mathtt{intern}\ \&\ \mathtt{Z}\ .\ \mathit{roommate} \doteq \mathtt{I}\ \&\ \mathtt{Z}\ .\ \mathit{advisor} \doteq \mathtt{D3}\ \&\ \mathtt{I} : \mathtt{student}\ \&$
     $\mathtt{I}\ .\ \mathit{rep} \doteq \mathtt{I}\ \&\ \underline{\mathtt{E} \doteq \mathtt{I}}$
     $\&$
     $\mathtt{D3} : \mathtt{hak}\ \&\ \mathtt{D3}\ .\ \mathit{secretary} \doteq \mathtt{E}\ \&\ \mathtt{E} : \mathtt{employee}$
     $\&$
     $\mathtt{Z}\ .\ \mathit{helper} \doteq \mathtt{H2}\ \&\ \mathtt{D3} : \mathtt{hak}\ \&\ \mathtt{D3}\ .\ \mathit{assistant} \doteq \mathtt{A}\ \&\ \mathtt{H2} : \mathtt{ali}\ \&\ \mathtt{H2}\ .\ \mathit{spouse} \doteq \mathtt{A}$
     $\&\ \mathtt{A} : \mathtt{person}\ \}$;
   - endomorphic mapping: $\gamma(\mathtt{X}) = \mathtt{Z}$, $\gamma(\mathtt{Y}) = \mathtt{Z}$, $\gamma(\mathtt{Q1}) = \mathtt{I}$, $\gamma(\mathtt{S}) = \mathtt{I}$, $\gamma(\mathtt{D1}) = \mathtt{D3}$, $\gamma(\mathtt{D2}) = \mathtt{D3}$;

10. apply Rule SORT INTERSECTION:

    - $\{\ \mathtt{Z} : \mathtt{intern}\ \&\ \mathtt{Z}\ .\ \mathit{roommate} \doteq \mathtt{I}\ \&\ \mathtt{Z}\ .\ \mathit{advisor} \doteq \mathtt{D3}\ \&\ \mathtt{I} : \mathtt{student}\ \&$
      $\mathtt{I}\ .\ \mathit{rep} \doteq \mathtt{I}$
      $\&$
      $\underline{\mathtt{D3} : \mathtt{hak}}\ \&\ \mathtt{D3}\ .\ \mathit{secretary} \doteq \mathtt{I}\ \&\ \mathtt{I} : \mathtt{employee}$
      $\&$
      $\mathtt{Z}\ .\ \mathit{helper} \doteq \mathtt{H2}\ \&\ \underline{\mathtt{D3} : \mathtt{hak}}\ \&\ \mathtt{D3}\ .\ \mathit{assistant} \doteq \mathtt{A}\ \&\ \mathtt{H2} : \mathtt{ali}\ \&\ \mathtt{H2}\ .\ \mathit{spouse} \doteq \mathtt{A}$
      $\&\ \mathtt{A} : \mathtt{person}\ \}$;

- endomorphic mapping: $\gamma(\mathtt{X}) = \mathtt{Z}$, $\gamma(\mathtt{Y}) = \mathtt{Z}$, $\gamma(\mathtt{Q1}) = \mathtt{I}$, $\gamma(\mathtt{S}) = \mathtt{I}$, $\gamma(\mathtt{D1}) = \mathtt{D3}$, $\gamma(\mathtt{D2}) = \mathtt{D3}$, $\gamma(\mathtt{E}) \stackrel{\text{def}}{=} \mathtt{I}$;

11. apply Rule **SORT INTERSECTION**:

- $\{\ \mathtt{Z} : \mathtt{intern}\ \&\ \mathtt{Z}\,.\,\mathit{roommate} \doteq \mathtt{I}\ \&\ \mathtt{Z}\,.\,\mathit{advisor} \doteq \mathtt{D3}\ \&\ \underline{\mathtt{I} : \mathtt{student}}\ \&\ \mathtt{I}\,.\,\mathit{rep} \doteq \mathtt{I}\ \&\ \mathtt{D3} : \mathtt{hak}$ $\&$ $\mathtt{D3}\,.\,\mathit{secretary} \doteq \mathtt{I}\ \&\ \underline{\mathtt{I} : \mathtt{employee}}$ $\&$ $\mathtt{Z}\,.\,\mathit{helper} \doteq \mathtt{H2}\ \&\ \mathtt{D3}\,.\,\mathit{assistant} \doteq \mathtt{A}\ \&\ \mathtt{H2} : \mathtt{ali}\ \&\ \mathtt{H2}\,.\,\mathit{spouse} \doteq \mathtt{A}\ \&\ \mathtt{A} : \mathtt{person}\ \}$;
- endomorphic mapping: $\gamma(\mathtt{X}) = \mathtt{Z}$, $\gamma(\mathtt{Y}) = \mathtt{Z}$, $\gamma(\mathtt{Q1}) = \mathtt{I}$, $\gamma(\mathtt{S}) = \mathtt{I}$, $\gamma(\mathtt{D1}) = \mathtt{D3}$, $\gamma(\mathtt{D2}) = \mathtt{D3}$, $\gamma(\mathtt{E}) = \mathtt{I}$;

12. apply Rule **RENAMING TAGS APART**:

- $\{\ \mathtt{Z} : \mathtt{intern}\ \&\ \mathtt{Z}\,.\,\mathit{roommate} \doteq \mathtt{I}\ \&\ \mathtt{Z}\,.\,\mathit{advisor} \doteq \mathtt{D3}\ \&\ \mathtt{I} : \mathtt{intern}\ \&\ \mathtt{I}\,.\,\mathit{rep} \doteq \mathtt{I}\ \&\ \mathtt{D3} : \mathtt{hak}$ $\&$ $\mathtt{D3}\,.\,\mathit{secretary} \doteq \mathtt{I}$ $\&$ $\underline{\mathtt{Z}\,.\,\mathit{helper} \doteq \mathtt{H2}}\ \&\ \mathtt{D3}\,.\,\mathit{assistant} \doteq \mathtt{A}\ \&\ \mathtt{H2} : \mathtt{ali}\ \&\ \underline{\mathtt{H2}\,.\,\mathit{spouse} \doteq \mathtt{A}}\ \&\ \underline{\mathtt{A} : \mathtt{person}}\ \}$;
- endomorphic mapping: $\gamma(\mathtt{X}) = \mathtt{Z}$, $\gamma(\mathtt{Y}) = \mathtt{Z}$, $\gamma(\mathtt{Q1}) = \mathtt{I}$, $\gamma(\mathtt{S}) = \mathtt{I}$, $\gamma(\mathtt{D1}) = \mathtt{D3}$, $\gamma(\mathtt{D2}) = \mathtt{D3}$, $\gamma(\mathtt{E}) = \mathtt{I}$, $\gamma(\mathtt{H2}) \stackrel{\text{def}}{=} \mathtt{H3}$, $\gamma(\mathtt{A}) \stackrel{\text{def}}{=} \mathtt{B}$;

13. normal form:

- $\{\ \mathtt{Z} : \mathtt{intern}\ \&\ \mathtt{Z}\,.\,\mathit{roommate} \doteq \mathtt{I}\ \&\ \mathtt{Z}\,.\,\mathit{advisor} \doteq \mathtt{D3}\ \&\ \mathtt{I} : \mathtt{intern}\ \&\ \mathtt{I}\,.\,\mathit{rep} \doteq \mathtt{I}\ \&\ \mathtt{D3} : \mathtt{hak}$ $\&$ $\mathtt{D3}\,.\,\mathit{secretary} \doteq \mathtt{I}$ $\&$ $\mathtt{Z}\,.\,\mathit{helper} \doteq \mathtt{H3}\ \&\ \mathtt{D3}\,.\,\mathit{assistant} \doteq \mathtt{B}\ \&\ \mathtt{H2} : \mathtt{ali}\ \&\ \mathtt{H3}\,.\,\mathit{spouse} \doteq \mathtt{B}\ \&\ \mathtt{B} : \mathtt{person}\ \}$;
- endomorphic mapping: $\gamma(\mathtt{X}) = \mathtt{Z}$, $\gamma(\mathtt{Y}) = \mathtt{Z}$, $\gamma(\mathtt{Q1}) = \mathtt{I}$, $\gamma(\mathtt{S}) = \mathtt{I}$, $\gamma(\mathtt{D1}) = \mathtt{D3}$, $\gamma(\mathtt{D2}) = \mathtt{D3}$, $\gamma(\mathtt{E}) = \mathtt{I}$, $\gamma(\mathtt{H2}) = \mathtt{H3}$, $\gamma(\mathtt{A}) = \mathtt{B}$.

This normal form is that of $\underline{\mathbf{t}} = \mathbf{glb}(\mathbf{t}_1, \mathbf{t}_2)$ shown in Figure B.2, with the corresponding endomorphic mapping γ such that $\underline{\mathbf{t}} = \gamma(\mathbf{t}_1) = \gamma(\mathbf{t}_2)$.

Example B.3 $\mathcal{OSF}$ **generalization** — Here is a step-by-step trace of constraint normalization computing the generalization $\mathbf{lub}(\mathbf{t}_1, \mathbf{t}_2)$ of the $\mathcal{OSF}$ terms $\mathbf{t}_1$ and $\mathbf{t}_2$ defined in Example B.1. It computes

their **lub**, along with the corresponding endomorphic mappings $\gamma_i : \mathbf{TagSet}(\mathbf{t}) \mapsto \mathbf{TagSet}(\mathbf{t}_i)$, for $i = 1, 2$, using the $\mathcal{OSF}$ generalization rules of Figure A.4 as follows.

We start by dissolving $\mathbf{t}_1$ and $\mathbf{t}_2$:

- $\varphi(\mathbf{t}_1) = \mathtt{X} : \mathtt{student}\ \&\ \mathtt{X}\,.\,\mathit{roommate} \doteq \mathtt{Q1}\ \&\ \mathtt{X}\,.\,\mathit{advisor} \doteq \mathtt{D1}\ \&\ \mathtt{Q1} : \mathtt{person}\ \&\ \mathtt{Q1}\,.\,\mathit{rep} \doteq \mathtt{E}\ \&\ \mathtt{D1} : \mathtt{hak}\ \&\ \mathtt{D1}\,.\,\mathit{secretary} \doteq \mathtt{E}\ \&\ \mathtt{E} : \mathtt{employee}$
- $\varphi(\mathbf{t}_2) = \mathtt{Y} : \mathtt{employee}\ \&\ \mathtt{Y}\,.\,\mathit{roommate} \doteq \mathtt{S}\ \&\ \mathtt{Y}\,.\,\mathit{advisor} \doteq \mathtt{D2}\ \&\ \mathtt{Y}\,.\,\mathit{helper} \doteq \mathtt{H2}\ \&\ \mathtt{S} : \mathtt{student}\ \&\ \mathtt{S}\,.\,\mathit{rep} \doteq \mathtt{S}\ \&\ \mathtt{D2} : \mathtt{hak}\ \&\ \mathtt{D2}\,.\,\mathit{assistant} \doteq \mathtt{A}\ \&\ \mathtt{H2} : \mathtt{ali}\ \&\ \mathtt{H2}\,.\,\mathit{spouse} \doteq \mathtt{A}\ \&\ \mathtt{A} : \mathtt{person}$

We then initiate the normalization process as indicated by Expression (A.27) on Page 123. Starting with:

$$\{\mathtt{P} = \mathbf{ROOT}(\mathbf{t}_1)\}, \{\mathtt{P} = \mathbf{ROOT}(\mathbf{t}_2)\} \ \vdash\ [\varphi(\mathbf{t}_1) \parallel \varphi(\mathbf{t}_2)]$$

where $\mathtt{P}$ is a new tag symbol for $\mathbf{ROOT}(\mathbf{lub}(\mathbf{t}_1, \mathbf{t}_2))$, and defining:

$$\gamma_1(\mathtt{P}) \stackrel{\text{def}}{=} \mathbf{ROOT}(\mathbf{t}_1) = \mathtt{X}$$
$$\gamma_2(\mathtt{P}) \stackrel{\text{def}}{=} \mathbf{ROOT}(\mathbf{t}_2) = \mathtt{Y}$$

we proceed applying the rules of Figure A.4 until none applies. This produces the following trace.[4]

1. Start with:
   - $[\ \underline{\mathtt{X} : \mathtt{student}}\ \&\ \mathtt{X}\,.\,\mathit{roommate} \doteq \mathtt{Q1}\ \&\ \mathtt{X}\,.\,\mathit{advisor} \doteq \mathtt{D1}\ \&\ \mathtt{Q1} : \mathtt{person}\ \&\ \mathtt{Q1}\,.\,\mathit{rep} \doteq \mathtt{E}\ \&\ \mathtt{D1} : \mathtt{hak}\ \&\ \mathtt{D1}\,.\,\mathit{secretary} \doteq \mathtt{E}\ \&\ \mathtt{E} : \mathtt{employee}$
     $\parallel$
     $\underline{\mathtt{Y} : \mathtt{employee}}\ \&\ \mathtt{Y}\,.\,\mathit{roommate} \doteq \mathtt{S}\ \&\ \mathtt{Y}\,.\,\mathit{advisor} \doteq \mathtt{D2}\ \&\ \mathtt{Y}\,.\,\mathit{helper} \doteq \mathtt{H2}\ \&\ \mathtt{S} : \mathtt{student}\ \&\ \mathtt{S}\,.\,\mathit{rep} \doteq \mathtt{S}\ \&\ \mathtt{D2} : \mathtt{hak}\ \&\ \mathtt{D2}\,.\,\mathit{assistant} \doteq \mathtt{A}\ \&\ \mathtt{H2} : \mathtt{ali}\ \&\ \mathtt{H2}\,.\,\mathit{spouse} \doteq \mathtt{A}\ \&\ \mathtt{A} : \mathtt{person}\ ]$
   - $\gamma_1 = \{\mathtt{P} = \mathtt{X}\}$
   - $\gamma_2 = \{\mathtt{P} = \mathtt{Y}\}$
2. apply Rule **SORT INDUCTION**:
   - $\mathtt{P} : \mathtt{person}$
     $[\underline{\mathtt{X}\,.\,\mathit{roommate} \doteq \mathtt{Q1}}\ \&\ \mathtt{X}\,.\,\mathit{advisor} \doteq \mathtt{D1}\ \&\ \mathtt{Q1} : \mathtt{person}\ \&\ \mathtt{Q1}\,.\,\mathit{rep} \doteq \mathtt{E}\ \&\ \mathtt{D1} : \mathtt{hak}\ \&\ \mathtt{D1}\,.\,\mathit{secretary} \doteq \mathtt{E}\ \&\ \mathtt{E} : \mathtt{employee}$
     $\parallel$
     $\underline{\mathtt{Y}\,.\,\mathit{roommate} \doteq \mathtt{S}}\ \&\ \mathtt{Y}\,.\,\mathit{advisor} \doteq \mathtt{D2}\ \&\ \mathtt{Y}\,.\,\mathit{helper} \doteq \mathtt{H2}\ \&\ \mathtt{S} : \mathtt{student}\ \&\ \mathtt{S}\,.\,\mathit{rep} \doteq \mathtt{S}\ \&\ \mathtt{D2} : \mathtt{hak}\ \&\ \mathtt{D2}\,.\,\mathit{assistant} \doteq \mathtt{A}\ \&\ \mathtt{H2} : \mathtt{ali}\ \&\ \mathtt{H2}\,.\,\mathit{spouse} \doteq \mathtt{A}\ \&\ \mathtt{A} : \mathtt{person}\ ]$
   - $\gamma_1 = \{\mathtt{P} = \mathtt{X}\}$

[4] As for $\mathcal{OSF}$ unification, we shall underline the parts of a constraint matching a generalization rule's prior pattern, which rule is then applied next.

- $\gamma_2 = \{\mathtt{P} = \mathtt{Y}\}$

3. apply Rule **FEATURE INDUCTION**:

   - $\mathtt{P} : \mathtt{person} \;\&\; \mathtt{P}\,.\,\mathit{roommate} \doteq \mathtt{S}$
     $[\underline{\mathtt{X}\,.\,\mathit{advisor} \doteq \mathtt{D1}} \;\&\; \mathtt{Q1} : \mathtt{person} \;\&\; \mathtt{Q1}\,.\,\mathit{rep} \doteq \mathtt{E} \;\&\; \mathtt{D1} : \mathtt{hak} \;\&\; \mathtt{D1}\,.\,\mathit{secretary} \doteq \mathtt{E}$
     $\;\&\; \mathtt{E} : \mathtt{employee}$
     $\|$
     $\underline{\mathtt{Y}\,.\,\mathit{advisor} \doteq \mathtt{D2}} \;\&\; \mathtt{Y}\,.\,\mathit{helper} \doteq \mathtt{H2} \;\&\; \mathtt{S} : \mathtt{student} \;\&\; \mathtt{S}\,.\,\mathit{rep} \doteq \mathtt{S} \;\&$
     $\mathtt{D2} : \mathtt{hak} \;\&\; \mathtt{D2}\,.\,\mathit{assistant} \doteq \mathtt{A} \;\&\; \mathtt{H2} : \mathtt{ali} \;\&\; \mathtt{H2}\,.\,\mathit{spouse} \doteq \mathtt{A} \;\&\; \mathtt{A} : \mathtt{person}$
     $]$
   - $\gamma_1 = \{\mathtt{P} = \mathtt{X}, \mathtt{Q} = \mathtt{Q1}\}$
   - $\gamma_2 = \{\mathtt{P} = \mathtt{Y}, \mathtt{Q} = \mathtt{S}\}$

4. apply Rule **FEATURE INDUCTION**:

   - $\mathtt{P} : \mathtt{person} \;\&\; \mathtt{P}\,.\,\mathit{roommate} \doteq \mathtt{Q} \;\&\; \mathtt{P}\,.\,\mathit{advisor} \doteq \mathtt{D}$
     $[\underline{\mathtt{Q1} : \mathtt{person}} \;\&\; \mathtt{Q1}\,.\,\mathit{rep} \doteq \mathtt{E} \;\&\; \mathtt{D1} : \mathtt{hak} \;\&\; \mathtt{D1}\,.\,\mathit{secretary} \doteq \mathtt{E} \;\&\; \mathtt{E} : \mathtt{employee}$
     $\|$
     $\mathtt{Y}\,.\,\mathit{helper} \doteq \mathtt{H2} \;\&\; \underline{\mathtt{S} : \mathtt{student}} \;\&\; \mathtt{S}\,.\,\mathit{rep} \doteq \mathtt{S} \;\&\; \mathtt{D2} : \mathtt{hak} \;\&\; \mathtt{D2}\,.\,\mathit{assistant} \doteq \mathtt{A}$
     $\;\&\; \mathtt{H2} : \mathtt{ali} \;\&\; \mathtt{H2}\,.\,\mathit{spouse} \doteq \mathtt{A} \;\&\; \mathtt{A} : \mathtt{person}]$
   - $\gamma_1 = \{\mathtt{P} = \mathtt{X}, \mathtt{Q} = \mathtt{Q1}, \mathtt{D} = \mathtt{D1}\}$
   - $\gamma_2 = \{\mathtt{P} = \mathtt{Y}, \mathtt{Q} = \mathtt{S}, \mathtt{D} = \mathtt{D2}\}$

5. apply Rule **SORT INDUCTION**:

   - $\mathtt{P} : \mathtt{person} \;\&\; \mathtt{P}\,.\,\mathit{roommate} \doteq \mathtt{Q} \;\&\; \mathtt{P}\,.\,\mathit{advisor} \doteq \mathtt{D} \;\&\; \mathtt{Q} : \mathtt{person}$
     $[\mathtt{Q1}\,.\,\mathit{rep} \doteq \mathtt{E} \;\&\; \underline{\mathtt{D1} : \mathtt{hak}} \;\&\; \mathtt{D1}\,.\,\mathit{secretary} \doteq \mathtt{E} \;\&\; \mathtt{E} : \mathtt{employee}$
     $\|$
     $\mathtt{Y}\,.\,\mathit{helper} \doteq \mathtt{H2} \;\&\; \mathtt{S}\,.\,\mathit{rep} \doteq \mathtt{S} \;\&\; \underline{\mathtt{D2} : \mathtt{hak}} \;\&\; \mathtt{D2}\,.\,\mathit{assistant} \doteq \mathtt{A} \;\&\; \mathtt{H2} : \mathtt{ali}$
     $\;\&\; \mathtt{H2}\,.\,\mathit{spouse} \doteq \mathtt{A} \;\&\; \mathtt{A} : \mathtt{person}]$
   - $\gamma_1 = \{\mathtt{P} = \mathtt{X}, \mathtt{Q} = \mathtt{Q1}, \mathtt{D} = \mathtt{D1}\}$
   - $\gamma_2 = \{\mathtt{P} = \mathtt{Y}, \mathtt{Q} = \mathtt{S}, \mathtt{D} = \mathtt{D2}\}$

6. apply Rule **SORT INDUCTION**:

   - $\mathtt{P} : \mathtt{person} \;\&\; \mathtt{P}\,.\,\mathit{roommate} \doteq \mathtt{Q} \;\&\; \mathtt{P}\,.\,\mathit{advisor} \doteq \mathtt{D} \;\&\; \mathtt{Q} : \mathtt{person} \;\&\; \mathtt{D} : \mathtt{hak}$
     $[\underline{\mathtt{Q1}\,.\,\mathit{rep} \doteq \mathtt{E}} \;\&\; \mathtt{D1} : \mathtt{hak} \;\&\; \mathtt{D1}\,.\,\mathit{secretary} \doteq \mathtt{E} \;\&\; \mathtt{E} : \mathtt{employee}$
     $\|$
     $\mathtt{Y}\,.\,\mathit{helper} \doteq \mathtt{H2} \;\&\; \underline{\mathtt{S}\,.\,\mathit{rep} \doteq \mathtt{S}} \;\&\; \mathtt{D2}\,.\,\mathit{assistant} \doteq \mathtt{A} \;\&\; \mathtt{H2} : \mathtt{ali} \;\&\; \mathtt{H2}\,.\,\mathit{spouse} \doteq \mathtt{A}$
     $\;\&\; \mathtt{A} : \mathtt{person}]$
   - $\gamma_1 = \{\mathtt{P} = \mathtt{X}, \mathtt{Q} = \mathtt{Q1}, \mathtt{D} = \mathtt{D1}\}$
   - $\gamma_2 = \{\mathtt{P} = \mathtt{Y}, \mathtt{Q} = \mathtt{S}, \mathtt{D} = \mathtt{D2}\}$

7. apply Rule **FEATURE INDUCTION**:

   - $\mathtt{P} : \mathtt{person} \;\&\; \mathtt{P} \,.\, \mathit{roommate} \doteq \mathtt{Q} \;\&\; \mathtt{P} \,.\, \mathit{advisor} \doteq \mathtt{D} \;\&\; \mathtt{Q} : \mathtt{person} \;\&\; \mathtt{D} : \mathtt{hak} \;\&\; \mathtt{Q} \,.\, \mathit{rep} \doteq \mathtt{R}$
     $\big[\, \mathtt{D1} \,.\, \mathit{secretary} \doteq \mathtt{E} \;\&\; \mathtt{E} : \mathtt{employee}$
     $\|$
     $\mathtt{Y} \,.\, \mathit{helper} \doteq \mathtt{H2} \;\&\; \mathtt{S} : \mathtt{student} \;\&\; \mathtt{S} \,.\, \mathit{rep} \doteq \mathtt{S} \;\&\; \mathtt{D2} : \mathtt{hak} \;\&\; \mathtt{D2} \,.\, \mathit{assistant} \doteq \mathtt{A} \;\&\; \mathtt{H2} : \mathtt{ali} \;\&\; \mathtt{H2} \,.\, \mathit{spouse} \doteq \mathtt{A} \;\&\; \mathtt{A} : \mathtt{person} \,\big]$
   - $\gamma_1 = \{\mathtt{P} = \mathtt{X}, \mathtt{Q} = \mathtt{Q1}, \mathtt{D} = \mathtt{D1}, \mathtt{R} = \mathtt{E}\}$
   - $\gamma_2 = \{\mathtt{P} = \mathtt{Y}, \mathtt{Q} = \mathtt{S}, \mathtt{D} = \mathtt{D2}, \mathtt{R} = \mathtt{S}\}$

8. apply Rule **SORT INDUCTION**:

   - $\mathtt{P} : \mathtt{person} \;\&\; \mathtt{P} \,.\, \mathit{roommate} \doteq \mathtt{Q} \;\&\; \mathtt{P} \,.\, \mathit{advisor} \doteq \mathtt{D} \;\&\; \mathtt{Q} : \mathtt{person} \;\&\; \mathtt{D} : \mathtt{hak} \;\&\; \mathtt{Q} \,.\, \mathit{rep} \doteq \mathtt{R} \;\&\; \mathtt{R} : \mathtt{person}$
     $\big[\, \mathtt{D1} \,.\, \mathit{secretary} \doteq \mathtt{E}$
     $\|$
     $\mathtt{Y} \,.\, \mathit{helper} \doteq \mathtt{H2} \;\&\; \mathtt{D2} \,.\, \mathit{assistant} \doteq \mathtt{A} \;\&\; \mathtt{H2} : \mathtt{ali} \;\&\; \mathtt{H2} \,.\, \mathit{spouse} \doteq \mathtt{A} \;\&\; \mathtt{A} : \mathtt{person} \,\big]$
   - $\gamma_1 = \{\mathtt{P} = \mathtt{X}, \mathtt{Q} = \mathtt{Q1}, \mathtt{D} = \mathtt{D1}, \mathtt{R} = \mathtt{E}\}$
   - $\gamma_2 = \{\mathtt{P} = \mathtt{Y}, \mathtt{Q} = \mathtt{S}, \mathtt{D} = \mathtt{D2}, \mathtt{R} = \mathtt{S}\}$

All the above can be summarized as illustrated by the lattice diagrams shown in Figure B.2.

B.2 Other Decidable $\mathcal{OSF}$ Constraints

The set of $\mathcal{OSF}$-constraints normalization rules presented thus far may be extended with useful additional axioms that enable other constraints commonly enforced in object/class-based systems; *viz.*., *partial features*, *element constructors*, and *aggregation*. We next describe additional rules that enforce such additional axioms.

§ PARTIAL FEATURES

Let $\mathbf{dom} : \mathcal{F} \mapsto \mathbf{2}^{\mathcal{S}}$ associate to a feature f its *domain* $\mathbf{dom}(f)$, the set of maximal sorts in $\mathcal{S}$ for which f is defined. A feature f is said to be:

- *total* when $\mathbf{dom}(f) = \{\top\}$;
- *undefined* when $\mathbf{dom}(f) = \{\bot\}$;
- *partial* when it is neither undefined nor total.

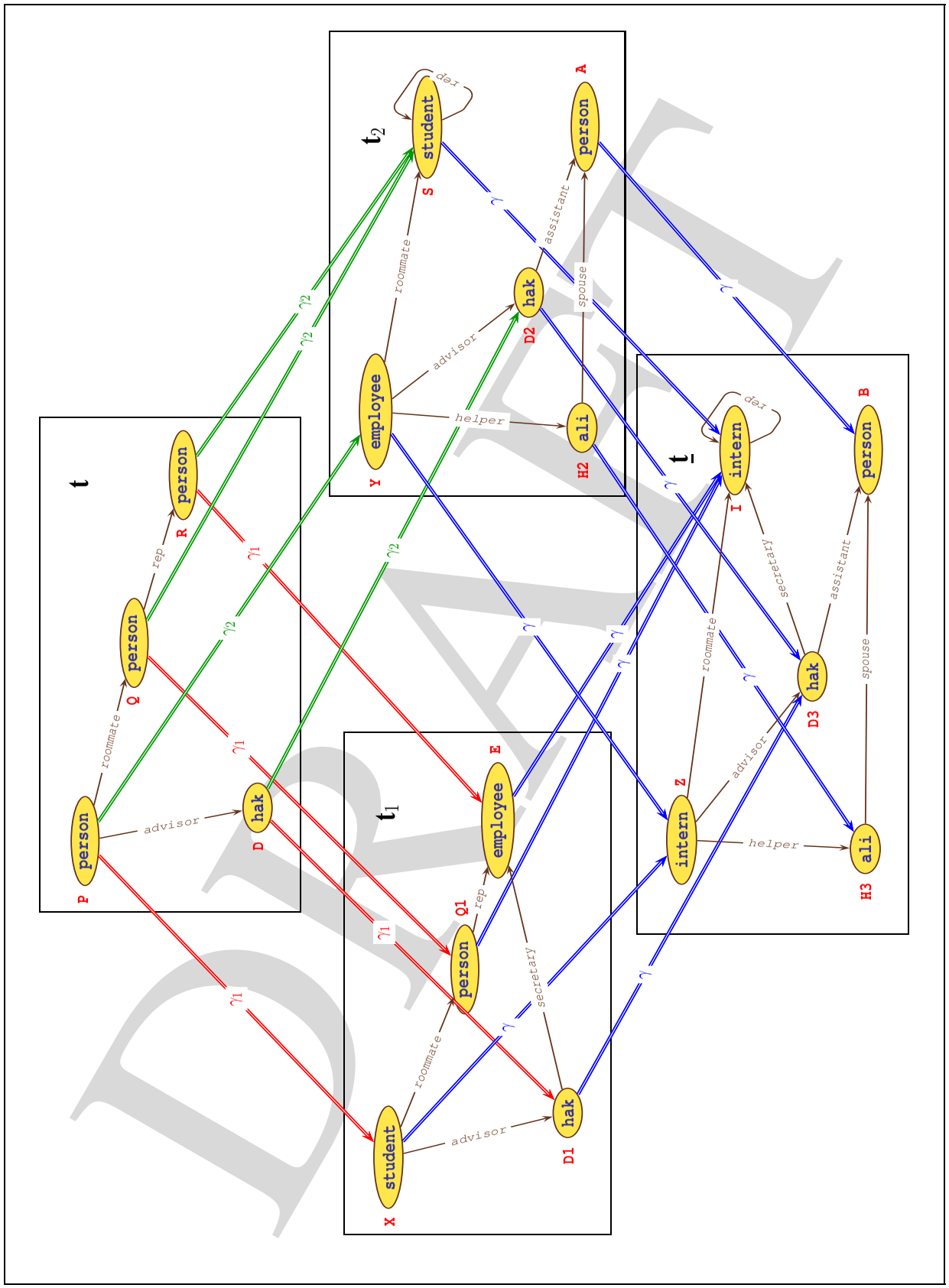


Figure B.2: Example of $\mathcal{OSF}$ subsumption lattice operations

Given a feature $f \in \mathcal{F}$, for each sort $s \in \mathbf{dom}(f)$, the *range* of f over s, denoted as $\mathbf{ran}_s(f) \in \mathcal{S}$, is the set of all maximal sorts of the possible values that feature f can take on sort s.[5] A possible $\mathcal{OSF}$-constraint normalization rule correctly enforcing such partial features is shown as Rule **PARTIAL FEATURE** in Figure B.3.

PARTIAL FEATURE:

$$\frac{\phi \;\&\; X.f \doteq X'}{\phi \;\&\; X.f \doteq X' \;\&\; X : s \;\&\; X' : s'} \; [s \in \mathbf{dom}(f);\;\; s' \in \mathbf{ran}_s(f)]$$

Figure B.3: Partial Feature

Note however that Rule **PARTIAL FEATURE** is non-deterministic, since there may be several incomparable maximal sorts making up the domain (or, for a sort in its domain, several ranges) for a partial feature when it has multiple domains and corresponding ranges (even for a single domain sort). One way to ease this issue is to introduce a new supersort (or use one if there is one) of all the domain sorts of a feature, and a new supersort (or use one if there is one) for all its range sorts. However, note that, semantically, this is not equivalent since there may be a loss of information since $f : \bigvee_{k\geq 2} d_k \to \bigvee_{k\geq 2} r_k$ is algebraically coarser than $f : \bigvee_{k\geq 2}(d_k \to r_k)$.[6]

This way of enforcing, even if only partially, known or declared domains and ranges for features is to generate constraints to that effect systematically when dissolving an $\mathcal{OSF}$-term. Indeed, we can also redefine the $\mathcal{OSF}$-dissolution function φ defined by Equation (3.5) on Page 71, with:

$$\begin{array}{rcl} \varphi(t) & \stackrel{\text{def}}{=} & X : s \;\&\; X.f_1 \doteq X_1 \;\&\; \ldots \;\&\; X.f_n \doteq X_n \\ & & \;\&\; X : s_1 \;\&\; \ldots \;\&\; X : s_n \\ & & \;\&\; X_1 : s'_1 \;\&\; \ldots \;\&\; X_n : s'_n \\ & & \;\&\; \varphi(t_1) \;\&\; \ldots \;\&\; \varphi(t_n) \end{array} \tag{B.2}$$

where $s_i \stackrel{\text{def}}{=} \bigvee \mathbf{dom}(f_i)$ (*i.e.*, the **lub** of all the known or declared maximal domains of f_i), and $s'_i \stackrel{\text{def}}{=} \bigvee_{s_i} \mathbf{ran}_{s_i}(f_i)$ (*i.e.*, the **lub** of all the known or declared ranges of f_i on domain s_i), for $i = 1, \ldots, n$.[7] This will include explicitly the most general feature domain/range constraints. Then, normalizing with the rules of Figure B.4 **PARTIAL FEATURE DOMAIN NARROWING** and **PARTIAL FEATURE RANGE NARROWING** will "finish the job," as each rule uses sort information on a feature's domain (resp., range) to further constrain its range (resp., domain) whenever this information is available. Importantly, the third side condition of each rule is to ensure finite termination of normalization by preventing repeated introductions of useless sort constraints, as is the case for example for Rule **PARTIAL FEATURE**—which, while being correct, is also non-deterministic as well as non-terminating!

[5] Computational linguists, who have borrowed heavily from the $\mathcal{OSF}$ formalism to express HPSG grammars for natural-language processing, call this category of axioms "*feature appropriateness*" axioms (see, *e.g.* [47]).

[6] Can you see why? (*Hint:* a range sort depends on its feature's domain—try for $k = 2$.)

[7] Again, if an actual sort symbol does not exist in $\mathcal{S}$ for either **lub** expressions, a new one is introduced for s_i or

PARTIAL FEATURE DOMAIN NARROWING:

$$\frac{\phi \;\&\; X.f \doteq X' \;\&\; X' : s'}{\phi \;\&\; X : s \;\&\; X.f \doteq X' \;\&\; X' : s'} \; [\mathbf{dom}(f) = \{s\};\ \mathbf{ran}_s(f) = \{s'\};\ X : s'' \notin \phi \text{ with } s'' \preceq s]$$

PARTIAL FEATURE RANGE NARROWING:

$$\frac{\phi \;\&\; X : s \;\&\; X.f \doteq X'}{\phi \;\&\; X : s \;\&\; X.f \doteq X' \;\&\; X' : s'} \; [\mathbf{dom}(f) = \{s\};\ \mathbf{ran}_s(f) = \{s'\};\ X' : s'' \notin \phi \text{ with } s'' \preceq s']$$

Figure B.4: Partial Feature Narrowing

All this special handling of partial features in the $\mathcal{OSF}$ machinery is done perhaps with the necessity of introducing many new symbols for missing unique **lub**s, but it can be done automatically and incurs no penalty for execution nor space in sort ordering checking nor lattice operations thanks to bit encoding techniques [12]. On the other hand, in practice, this narrows features to their appropriate domain and range sorts, including eliminating spurious feature applications, even trivial ones such as, *e.g.*, typos.

§ ELEMENT CONSTRUCTORS

A sort denotes a set of values of the domain of interpretation. When this set is a singleton, the sort is assimilated to the value contained in the denoted singleton (*e.g.*, a number, a string, *etc.*). However, such data may also have structure; then, it is assimilated to a data constructor. Such a structure denotes an individual element only when all its subterms under a set of specific features do as well. For example, a pair constructor "**pair**" with features "**left**" and "**right**" will denote a single pair object only when both subterms under these two features denote each a single individual; otherwise, it denotes a set. Thus, the $\mathcal{OSF}$ term $\mathtt{pair}(\mathtt{left} \to \mathtt{1}, \mathtt{right} \to \mathtt{2})$ denotes the *individual* object $\langle 1, 2\rangle$, whereas if the sort **nat** denotes the set of natural numbers $\mathbb{N}$, then the term $\mathtt{pair}(\mathtt{left} \to \mathtt{nat}, \mathtt{right} \to \mathtt{nat})$ denotes the *set* of all pairs whose left and right subterms are natural numbers (*viz.*, $\{\, \langle m, n\rangle \mid m \in \mathbb{N},\ n \in \mathbb{N} \,\}$). For such sorts, we must then ensure that only individual-denoting terms are uniquely represented.

Let $\mathcal{E}$ (for "*element*," or "*extensional*," sorts) be the set of sorts in $\mathcal{S}$ that are element constructors. Define the *arity* $\mathbf{arity}(e)$ of such an element sort e giving its *feature arity* as a set of features—*i.e.*, $\mathbf{arity} : \mathcal{E} \mapsto \mathbf{2}^{\mathcal{F}}$. The set $\mathbf{arity}(e)$ is the set of features that completely determine the unique element of sort e. In other words, whenever all features of $\mathbf{arity}(e)$ denote singletons, then so does e. All such values ought to be uniquely identified. Note in passing that all atomic constants in $\mathcal{E}$ always have empty arity. For example, for any number n, $\mathbf{arity}(n) = \emptyset$. The $\mathcal{OSF}$-constraint normalization rule that enforces this uniqueness axiom on element sorts is called Rule **WEAK EXTENSIONALITY** as shown in Figure B.5.

s'_i.

WEAK EXTENSIONALITY:

$$\frac{\phi \;\&\; X : s \;\&\; X' : s}{\phi \;\&\; X : s \;\&\; X \doteq X'} \left[s \in \mathcal{E};\ \ \forall f \in \mathbf{arity}(s), \{X.f \doteq Y, X'.f \doteq Y\} \subseteq \phi\right]$$

Figure B.5: Weak Extensionality

With this rule, for example, if $\mathcal{S} = \{\top, \bot, \mathtt{nil}, \mathtt{cons}, \mathtt{list}, \mathtt{nat}, 0, 1, 2, \ldots\}$ such that $\mathtt{nil} < \mathtt{list}$, $\mathtt{cons} < \mathtt{list}$, $n < \mathtt{nat}$ for $n \in \mathbb{N}$ (where $<$ is the subsort ordering). Let $\mathcal{E} = \{\mathtt{nil}, \mathtt{cons}, n\},\ (n \in \mathbb{N})$, such that $\mathbf{arity}(\mathtt{nil}) = \emptyset$, $\mathbf{arity}(\mathtt{cons}) = \{\mathtt{head}, \mathtt{tail}\}$, and $\mathbf{arity}(n) = \emptyset$ for $n \in \mathbb{N}$. Then, the $\mathcal{OSF}$ term:

$$X : \mathtt{cons}(\mathtt{head} \to \mathtt{1}, \mathtt{tail} \to \mathtt{nil}) \;\&\; Y : \mathtt{cons}(\mathtt{head} \to \mathtt{1}, \mathtt{tail} \to \mathtt{nil})$$

is normalized into:

$$X : \mathtt{cons}(\mathtt{head} \to \mathtt{1}, \mathtt{tail} \to \mathtt{nil}) \;\&\; X \doteq Y$$

This rule is called *"weak"* because it can only enforce uniqueness of *acyclic* elements. Rules with a stronger condition working for cyclic terms are given next.

§ STRONG EXTENSIONALITY

Basically, the reason why Rule **WEAK EXTENSIONALITY** of Figure B.5 does not recognize singletons that are cyclic terms is that it works *inductively*. Doing so, it is well-founded only because it proceeds from leaves to their roots. However, for cyclic terms, there may be no leaf to proceed from. Consider, for example, an extensional sort $s \in \mathcal{E}$ such that $\mathbf{arity}(s) = \{f\}$, and the conjunction of cyclic terms:

$$X : s(f \to X) \;\&\; X' : s(f \to X') \tag{B.3}$$

or, even better, that of the mutually cyclic terms:

$$X : s(f \to X') \;\&\; X' : s(f \to X). \tag{B.4}$$

Now, $\mathbf{arity}(s) = \{f\}$ means that *"s denotes a singleton sort whenever its f feature denotes one as well."* Semantically, in both examples, variables X and X' denote therefore the same element (due to *all* the features in $\mathbf{arity}(s)$ being consistently sorted as singletons). However, the Rule **WEAK EXTENSIONALITY** does not transform either term (B.3) or term (B.4) into one where X and X' are equal as they should be as per the semantics of arity and extensionality. Therefore, this inductive manner of proceeding will not work for cyclic extensional terms such as these.

The alternative is to proceed *coinductively*, from roots to leaves or previously processed nodes, while keeping a record of which extensional sorts appear with which variables, since such sorts denote single element. This is done by carrying an *occurrence context* as a set Γ of

elements of the form $s : \{X_1, \ldots, X_n\}$, where $X_i \in \mathcal{V}$, for $i = 1, \ldots, n, (n \geq 0)$, where $s \in \mathcal{E}$ is extensional, and such that each such s may not occur more than once in any such occurrence context Γ. A *contexted rule* is one of the form:

(Rule Number) **RULE NAME** :

$$\frac{\mathit{Prior\ Context} \vdash \mathit{Prior\ Form}}{\mathit{Posterior\ Context} \vdash \mathit{Posterior\ Form}} \ [\mathit{Condition}]$$

Appropriate extensional sort occurrences record-keeping is thus achieved using contexted Rule *Extensional Variable* of Figure B.6. The "real" work is then done by contexted Rule **STRONG EXTENSIONALITY**. Using these two rules on weak normal forms will work as expected; *viz.*, it will merge any remaining potential cyclic extensional elements that denote the same individual.

EXTENSIONAL VARIABLE:

$$\frac{\Gamma \uplus \{s : V, \ldots,\} \ \vdash \ \phi \ \& \ X : s}{\Gamma \uplus \{s : V \cup \{X\}, \ldots\} \ \vdash \ \phi \ \& \ X : s} \left[\begin{array}{l} X \notin V; \ \ s \in \mathcal{E}; \ \ s' \in \mathcal{E}; \\ \forall f \in \mathbf{arity}(s), \ \{X.f \doteq X', X' : s'\} \subseteq \phi \end{array}\right]$$

STRONG EXTENSIONALITY:

$$\frac{\Gamma \uplus \{s : \{X, X', \ldots\} \ \vdash \ \phi}{\Gamma \uplus \{s : \{X, \ldots\} \ \vdash \ \phi \ \& \ X \doteq X'} \ [s \in \mathcal{E}]$$

Figure B.6: Strong extensionality

§ RELATIONAL FEATURES AND AGGREGATION

The $\mathcal{OSF}$ formalism deals with functional features. However, relational features may also come handy. A relational feature is a binary relation or, equivalently, a set-valued function. In other words, a multi-valued functional attribute may be aggregated into a set. Indeed, combining Rule **SORT INTERSECTION** with Rule **FEATURE FUNCTIONALITY** of Figure 3.6 enforces that a variable's sort, and hence value, may only be computed by intersection of consistent sorts. On the other hand, a relational feature denotes a set-valued function, and normalization must thus provide a means for aggregating mutually distinct values of a sort.

This semantics can be accommodated with the following value aggregation rule, which generalizes Rule **SORT INTERSECTION**. The notation for the atomic constraint "$X : s$" is generalized to carry an optional value $e \in \mathcal{E}$ (*i.e.*, e is an extensional sort): "$X = e : s$" means "X has value e of sort s," where $X \in \mathcal{V}$, $e \in \mathcal{E}$, $s \in \mathcal{S}$. The shorthand "$X = e$" means "$X = e : \top$." When the sort $s \in \mathcal{S}$ is a commutative monoid $\langle \star, \mathbf{1}_\star \rangle$, the shorthand "$X : s$" means "$X = \mathbf{1}_\star : s$."

The conditions (3.4) are then extended with:

$$\mathcal{A}, \alpha \ \models \ X = e : s \ \text{ iff } \ e^{\mathcal{A}} \in s^{\mathcal{A}} \ \text{ and } \ \alpha(X) = e^{\mathcal{A}}. \tag{B.5}$$

Thus, element values of a sort that denotes a commutative monoid $M = \langle \star, \mathbf{1}_\star \rangle$ may be composed using this monoid's operation. In particular, such a monoid operation may be that of a *set* constructor; *i.e.*, one that is associative, commutative, and idempotent. This is what Rule **VALUE AGGREGATION** of Figure B.7 accommodates.

VALUE AGGREGATION:

$$\frac{\phi \;\&\; X = e : s \;\&\; X = e' : s'}{\phi \;\&\; X = e \star e' : s \wedge s'} \; [s, s' \text{ both subsorts of commutative monoid } \langle \star, \mathbf{1}_\star \rangle]$$

Figure B.7: Aggregation

Note that Rule **VALUE AGGREGATION** is more general than need be for just accommodating aggregating a set (*i.e.*, a commutative idempotent free monoid) or a multiset (commutative but non-idempotent free monoid). Indeed, it also accommodates any other commutative monoids using aggregation operations such as min, max, sum, product, *etc.*, ... Thus, one may use this rule by using **AGGREGATE**$(f, s, m, \star, \mathbf{1}_\star)$ to declare the fact that when feature f is applied on (domain) sort s, it takes values in (range) sort m denoting a specific commutative monoid $\langle \star, \mathbf{1}_\star \rangle$ (*i.e.*, $s \in \mathbf{dom}(f)$ and $\mathbf{ran}_s(f) = m$). In other words,

$$X : s \;\&\; X.f \doteq Y \;\&\; Y = \mathbf{1}_\star : m. \tag{B.6}$$

Then, the rules **PARTIAL FEATURE DOMAIN/RANGE NARROWING** used in conjunction with Rule **VALUE AGGREGATION** of Figure B.7 will use such a declaration to proceed with the correct aggregation for so-declared features.

For example, declaring: **AGGREGATE**(`offspring`, **person**, **person**, $\cup, \emptyset$), would ensure that whenever applied on sort **person** feature `offspring` aggregates values of sort **person** using set union ($\cup$). The default value is the empty set ($\emptyset$). If the feature `offspring` of a **person** object is an individual element p of sort **person**, it is assimilated to the singleton set $\{p\}$.

Note however that we require a *commutative* monoid to ensure confluence of this rule with the other $\mathcal{OSF}$-constraint normalization rules in a non-deterministic normalization setting. In other words, the order in which the rules are applied does not matter on the outcome of the aggregation only when the monoid operation is commutative. This rule can also be used on non-commutative collection structures such as lists (free monoid), although the order of application may then result in different structures.

Decidability results concerning the differences between attributive concepts using functional features *vs.* relation roles are reviewed in [112]. Aggregation has also been considered in the same setting in [34] with similar decidability results. This last work offers intriguing potential connections with the paradigm of declarative aggregation as described in [60] or [66] where a versatile computable algebraic theory of monoid comprehensions is defined in terms of monoid homomorphisms effective declarative aggregates where the aggregation can be any associative binary operation. As defined in [59], a Monoid Comprehension Calculus can be defined as a

conservative extension of the λ-calculus with aggregates to specify an object-relational model enjoying algebraic properties that greatly facilitate query optimization.

B.3 $\mathcal{FOT}$ Terms as $\mathcal{OSF}$ Terms

As mentioned at the beginning of Chapter 3, a $\mathcal{FOT}$ can be seen as a special case of $\mathcal{OSF}$ term. More precisely, it is an $\mathcal{OSF}$ term when functional constructor symbols are syntactically seen as mutually incomparable sorts, all of which are subsorts of one "*most general sort*" (denoted using the symbol '$\top$' and denoting the congruence class of any variable modulo renaming), and all of which are supersorts of one "*least general sort*" the sort of no term (denoted using the symbol '$\bot$' and denoting the empty set) which can be construed as failure of unification. Its features are argument positions. As functions, they are are subterm projections: for any function symbol f of arity n, $f(t_1, \ldots, t_n).i \stackrel{\text{def}}{=} t_i$, for $i \in \{1, \ldots, n\}$. However, the semantics of arity imposes that positions be partial features (*i.e.*, position i is defined for sort f/n only for $i \leq n$). We can formally recast Herbrand and rational terms as $\mathcal{OSF}$ terms as follows.

A first-order rational term in $\mathcal{T}_{\Sigma,\mathcal{V}}$ can be viewed as a particular ψ-term. For this, it suffices to take $\mathcal{S} = \Sigma \cup \{\top, \bot\}$ and $\mathcal{F} = \mathbb{N} - \{0\}$. Namely, function symbols in $\Sigma = \bigcup_{n>0} \Sigma_n$ denote singleton sorts (*i.e.*, they are mutually incomparable and $\forall f \in \Sigma, \bot \prec f \prec \top$), and positive integers as features (*i.e.*, argument projections). Thus, the term $f(t_1, \ldots, t_n)$ is the ψ-term $f(1 \rightarrow t_1, \ldots, n \rightarrow t_n)$. The features here are *argument positions* and are interpreted in the $\mathcal{OSF}$ formalism as *projection functions*. Additional axioms are needed to enforce arity constraints. Namely:

$$\textbf{arity}(\top) = \emptyset \tag{B.7}$$

$$\textbf{arity}(\bot) = \{\, i \in \mathbb{N} - \{0\} \mid i \leq \max\{n > 0 \,|\, \Sigma_n \neq \emptyset\} \,\} \tag{B.8}$$

$$\forall f \in \Sigma_n : \quad \textbf{arity}(f) = \{1, \ldots, n\} \tag{B.9}$$

$$\forall i \in \mathcal{F} : \quad \textbf{dom}(i) = \bigcup_{i \leq n} \Sigma_n \tag{B.10}$$

$$\forall i \in \mathcal{F}, \forall f \in \Sigma : \quad \textbf{ran}_f(i) = \begin{cases} \top & \text{if } f \in \textbf{dom}(i), \\ \bot & \text{otherwise.} \end{cases} \tag{B.11}$$

Condition (B.7) states that $\top$ has empty arity. This corresponds to the fact that logical variables may appear only as term leaves. Condition (B.8) states that $\bot$ has the maximal arity of all symbols. Condition (B.9) declares the arity for each function symbol. Condition (B.10) declares the domains for each argument position—namely, the set of symbols that have at least that many arguments. Condition (B.11) enforces the domains and ranges declared in the signature for function symbols according to their arity constraints.

For sorted algebras, the sort signature $\mathcal{S}$ may also contain non-minimal sorts above the singleton-denoting functors of Σ. Thus, multi- or order-sorted versions of free term algebra $\mathcal{T}_{\Sigma,\mathcal{V}}$

are readily expressible in the $\mathcal{OSF}$ formalism by making Condition (B.11) involve non-singleton sorts other than $\top$ as the range of projection features. With these signature constraints,rules **PARTIAL FEATURE DOMAIN/RANGE NARROWING** of Figure B.4 combined with the basic $\mathcal{OSF}$ normalization rules of Figure 3.6 will make unification of (rational) Herbrand terms behave as expected.[8]

It is therefore expected that on such restricted $\mathcal{OSF}$ terms, the $\mathcal{OSF}$ unification (Figure 3.6) and generalization (Figure A.4) operations correspond to $\mathcal{FOT}$ unification (Figure 2.3) and generalization (Figure 2.5), respectively.

However, for this to be the case, implicit extensionality must be explicitly taken into account. Let us take an example to illustrate this: seen as an $\mathcal{OSF}$ term, the $\mathcal{FOT}$ $f(a,a)$ is the $\mathcal{OSF}$ term $f(1 \to X_1 : a, 2 \to X_2 : a)$ rather than $f(1 \to X_1 : a, 2 \to X_1)$. These two $\mathcal{OSF}$ terms will have the same denotation only if the sort symbols f and a are considered extensional, and if $f \in \Sigma_2$ (*i.e.*, its arity is 2). Then, Rule **WEAK EXTENSIONALITY** of Figure B.5 will ensure identities of equal denotations.[9]

When all signature functors are declared as extensional and their positions are partial features, then the $\mathcal{OSF}$ unification and generalization of $\mathcal{FOT}$s coincides with $\mathcal{FOT}$ unification and generalization, respectively.

Example B.4 **$\mathcal{FOT}$ generalization using $\mathcal{OSF}$ rules** — Since a $\mathcal{FOT}$ is a special case of $\mathcal{OSF}$ term, we can verify that using the $\mathcal{OSF}$ generalization rules of Figure A.4 computes the correct $\mathcal{FOT}$ generalizing two given $\mathcal{FOT}$s. Let us illustrate this using the terms of Example 2.3 on Page 18.

We are given two $\mathcal{FOT}$s: $t_1 \stackrel{\text{def}}{=} f(a,a,a)$ and $t_2 \stackrel{\text{def}}{=} f(b,c,c)$. Written as $\mathcal{OSF}$ terms, they correspond to the ψ-terms:

$$\psi_1 \stackrel{\text{def}}{=} X_1 : f(1 \to Y_1 : a,\ 2 \to Z_1 : a,\ 3 \to U_1 : a)$$

$$\psi_2 \stackrel{\text{def}}{=} X_2 : f(1 \to Y_2 : b,\ 2 \to Z_2 : c,\ 3 \to U_2 : c)$$

which, once dissolved, correspond to the following $\mathcal{OSF}$ constraints:

$$\begin{aligned}\phi_1 \stackrel{\text{def}}{=}\ & \mathtt{X1} : \mathtt{f} \ \&\ \mathtt{X1}\,.\,1 \doteq \mathtt{Y1} \ \&\ \mathtt{Y1} : \mathtt{a} \\ & \&\ \mathtt{X1}\,.\,2 \doteq \mathtt{Z1} \ \&\ \mathtt{Z1} : \mathtt{a} \\ & \&\ \mathtt{X1}\,.\,3 \doteq \mathtt{U1} \ \&\ \mathtt{U1} : \mathtt{a}\end{aligned}$$

$$\begin{aligned}\phi_2 \stackrel{\text{def}}{=}\ & \mathtt{X2} : \mathtt{f} \ \&\ \mathtt{X2}\,.\,1 \doteq \mathtt{Y2} \ \&\ \mathtt{Y2} : \mathtt{b} \\ & \&\ \mathtt{X2}\,.\,2 \doteq \mathtt{Z2} \ \&\ \mathtt{Z2} : \mathtt{c} \\ & \&\ \mathtt{X2}\,.\,3 \doteq \mathtt{U2} \ \&\ \mathtt{U2} : \mathtt{c}.\end{aligned}$$

We then initiate the normalization process as indicated by Expression (A.27) on Page 123.

1. Start with:
   - $\gamma_1 = \{\mathtt{X} = \mathtt{X1}\}$
   - $\gamma_2 = \{\mathtt{X} = \mathtt{X2}\}$

[8] Since rational terms may be cyclic, for rational-term unification, one must dispense with $X \notin \mathbf{var}(t)$ test in Rule **VARIABLE ELIMINATION** [122].

[9] The strong extensionality rules of Figure B.6 enforce these identities for cyclic terms as well.

- $\left[\underline{\mathtt{X1}:\mathtt{f}}\ \&\ \mathtt{X1}\,.\,1\doteq\mathtt{Y1}\ \&\ \mathtt{Y1}:\mathtt{a}\ \&\ \mathtt{X1}\,.\,2\doteq\mathtt{Z1}\ \&\ \mathtt{Z1}:\mathtt{a}\ \&\ \mathtt{X1}\,.\,3\doteq\mathtt{U1}\ \&\ \mathtt{U1}:\mathtt{a}\right.$
  $\|$
  $\left.\underline{\mathtt{X2}:\mathtt{f}}\ \&\ \mathtt{X2}\,.\,1\doteq\mathtt{Y2}\ \&\ \mathtt{Y2}:\mathtt{b}\ \&\ \mathtt{X2}\,.\,2\doteq\mathtt{Z2}\ \&\ \mathtt{Z2}:\mathtt{c}\ \&\ \mathtt{X2}\,.\,3\doteq\mathtt{U2}\ \&\ \mathtt{U2}:\mathtt{c}\right]$

2. apply Rule **FEATURE INDUCTION**:
   - $\gamma_1 = \{\mathtt{X}=\mathtt{X1}\}$
   - $\gamma_2 = \{\mathtt{X}=\mathtt{X2}\}$
   - $\mathtt{X}:\mathtt{f}$
     $\left[\underline{\mathtt{X1}\,.\,1\doteq\mathtt{Y1}}\ \&\ \mathtt{Y1}:\mathtt{a}\ \&\ \mathtt{X1}\,.\,2\doteq\mathtt{Z1}\ \&\ \mathtt{Z1}:\mathtt{a}\ \&\ \mathtt{X1}\,.\,3\doteq\mathtt{U1}\ \&\ \mathtt{U1}:\mathtt{a}\right.$
     $\|$
     $\left.\underline{\mathtt{X2}\,.\,1\doteq\mathtt{Y2}}\ \&\ \mathtt{Y2}:\mathtt{b}\ \&\ \mathtt{X2}\,.\,2\doteq\mathtt{Z2}\ \&\ \mathtt{Z2}:\mathtt{c}\ \&\ \mathtt{X2}\,.\,3\doteq\mathtt{U2}\ \&\ \mathtt{U2}:\mathtt{c}\right]$

3. apply Rule **FEATURE INDUCTION**:
   - $\gamma_1 = \{\mathtt{X}=\mathtt{X1},\mathtt{Y}=\mathtt{Y1}\}$
   - $\gamma_2 = \{\mathtt{X}=\mathtt{X2},\mathtt{Y}=\mathtt{Y2}\}$
   - $\mathtt{X}:\mathtt{f}\ \&\ \mathtt{X}\,.\,1\doteq\mathtt{Y}$
     $\left[\underline{\mathtt{Y1}:\mathtt{a}}\ \&\ \mathtt{X1}\,.\,2\doteq\mathtt{Z1}\ \&\ \mathtt{Z1}:\mathtt{a}\ \&\ \mathtt{X1}\,.\,3\doteq\mathtt{U1}\ \&\ \mathtt{U1}:\mathtt{a}\right.$
     $\|$
     $\left.\underline{\mathtt{Y2}:\mathtt{b}}\ \&\ \mathtt{X2}\,.\,2\doteq\mathtt{Z2}\ \&\ \mathtt{Z2}:\mathtt{c}\ \&\ \mathtt{X2}\,.\,3\doteq\mathtt{U2}\ \&\ \mathtt{U2}:\mathtt{c}\right]$

4. apply Rule **SORT INDUCTION**:
   - $\gamma_1 = \{\mathtt{X}=\mathtt{X1},\mathtt{Y}=\mathtt{Y1}\}$
   - $\gamma_2 = \{\mathtt{X}=\mathtt{X2},\mathtt{Y}=\mathtt{Y2}\}$
   - $\mathtt{X}:\mathtt{f}\ \&\ \mathtt{X}\,.\,1\doteq\mathtt{Y}\ \&\ \mathtt{Y}:\mathtt{a}\vee\mathtt{b}$
     $\left[\underline{\mathtt{X1}\,.\,2\doteq\mathtt{Z1}}\ \&\ \mathtt{Z1}:\mathtt{a}\ \&\ \mathtt{X1}\,.\,3\doteq\mathtt{U1}\ \&\ \mathtt{U1}:\mathtt{a}\right.$
     $\|$
     $\left.\underline{\mathtt{X2}\,.\,2\doteq\mathtt{Z2}}\ \&\ \mathtt{Z2}:\mathtt{c}\ \&\ \mathtt{X2}\,.\,3\doteq\mathtt{U2}\ \&\ \mathtt{U2}:\mathtt{c}\right]$

5. apply Rule **FEATURE INDUCTION**:
   - $\gamma_1 = \{\mathtt{X}=\mathtt{X1},\mathtt{Y}=\mathtt{Y1},\mathtt{Z}=\mathtt{Z1}\}$
   - $\gamma_2 = \{\mathtt{X}=\mathtt{X2},\mathtt{Y}=\mathtt{Y2},\mathtt{Z}=\mathtt{Z2}\}$
   - $\mathtt{X}:\mathtt{f}\ \&\ \mathtt{X}\,.\,1\doteq\mathtt{Y}\ \&\ \mathtt{Y}:\mathtt{a}\vee\mathtt{b}\ \&\ \mathtt{X1}\,.\,2\doteq\mathtt{Z}$
     $\left[\underline{\mathtt{Z1}:\mathtt{a}}\ \&\ \mathtt{X1}\,.\,3\doteq\mathtt{U1}\ \&\ \mathtt{U1}:\mathtt{a}\right.$
     $\|$
     $\left.\underline{\mathtt{Z2}:\mathtt{c}}\ \&\ \mathtt{X2}\,.\,3\doteq\mathtt{U2}\ \&\ \mathtt{U2}:\mathtt{c}\right]$

6. apply Rule **SORT INDUCTION**:
   - $\gamma_1 = \{\mathtt{X}=\mathtt{X1},\mathtt{Y}=\mathtt{Y1},\mathtt{Z}=\mathtt{Z1}\}$

- $\gamma_2 = \{\mathtt{X} = \mathtt{X2}, \mathtt{Y} = \mathtt{Y2}, \mathtt{Z} = \mathtt{Z2}\}$
- $\mathtt{X} : \mathtt{f} \;\&\; \mathtt{X} \,.\, 1 \doteq \mathtt{Y} \;\&\; \mathtt{Y} : \mathtt{a} \vee \mathtt{b} \;\&\; \mathtt{X1} \,.\, 2 \doteq \mathtt{Z} \;\&\; \mathtt{Z} : \mathtt{a} \vee \mathtt{c}$
  $[\, \underline{\mathtt{X1} \,.\, 3 \doteq \mathtt{U1}} \;\&\; \mathtt{U1} : \mathtt{a}$
  $\|$
  $\underline{\mathtt{X2} \,.\, 3 \doteq \mathtt{U2}} \;\&\; \mathtt{U2} : \mathtt{c}\,]$

7. apply Rule **FEATURE INDUCTION**:

   - $\gamma_1 = \{\mathtt{X} = \mathtt{X1}, \mathtt{Y} = \mathtt{Y1}, \mathtt{Z} = \mathtt{Z1}, \mathtt{U} = \mathtt{U1}\}$
   - $\gamma_2 = \{\mathtt{X} = \mathtt{X2}, \mathtt{Y} = \mathtt{Y2}, \mathtt{Z} = \mathtt{Z2}, \mathtt{U} = \mathtt{U2}\}$
   - $\mathtt{X} : \mathtt{f} \;\&\; \mathtt{X} \,.\, 1 \doteq \mathtt{Y} \;\&\; \mathtt{Y} : \mathtt{a} \vee \mathtt{b} \;\&\; \mathtt{X} \,.\, 2 \doteq \mathtt{Z} \;\&\; \mathtt{Z} : \mathtt{a} \vee \mathtt{c} \;\&\; \mathtt{X} \,.\, 3 \doteq \mathtt{U}$
     $[\, \underline{\mathtt{U1} : \mathtt{a}}$
     $\|$
     $\underline{\mathtt{U2} : \mathtt{c}}\,]$

8. apply Rule **SORT INDUCTION**:

   - $\gamma_1 = \{\mathtt{X} = \mathtt{X1}, \mathtt{Y} = \mathtt{Y1}, \mathtt{Z} = \mathtt{Z1}, \mathtt{U} = \mathtt{U1}\}$
   - $\gamma_2 = \{\mathtt{X} = \mathtt{X2}, \mathtt{Y} = \mathtt{Y2}, \mathtt{Z} = \mathtt{Z2}, \mathtt{U} = \mathtt{U2}\}$
   - $\mathtt{X} : \mathtt{f} \;\&\; \mathtt{X} \,.\, 1 \doteq \mathtt{Y} \;\&\; \mathtt{Y} : \mathtt{a} \vee \mathtt{b} \;\&\; \mathtt{X} \,.\, 2 \doteq \mathtt{Z} \;\&\; \mathtt{Z} : \mathtt{a} \vee \mathtt{c} \;\&\; \mathtt{X} \,.\, 3 \doteq \mathtt{U} \;\&\; \mathtt{U} : \mathtt{a} \vee \mathtt{c}$

9. apply Rule **WEAK EXTENSIONALITY** (on value $\mathtt{a} \vee \mathtt{c}$ eliminating tag $\mathtt{U}$ for tag $\mathtt{Z}$).

   - $\gamma_1 = \{\mathtt{X} = \mathtt{X1}, \mathtt{Y} = \mathtt{Y1}, \mathtt{Z} = \mathtt{Z1}, \mathtt{Z} = \mathtt{U1}\}$
   - $\gamma_2 = \{\mathtt{X} = \mathtt{X2}, \mathtt{Y} = \mathtt{Y2}, \mathtt{Z} = \mathtt{Z2}, \mathtt{Z} = \mathtt{U2}\}$
   - $\mathtt{X} : \mathtt{f} \;\&\; \mathtt{X} \,.\, 1 \doteq \mathtt{Y} \;\&\; \mathtt{Y} : \mathtt{a} \vee \mathtt{b} \;\&\; \mathtt{X} \,.\, 2 \doteq \mathtt{Z} \;\&\; \mathtt{X} \,.\, 3 \doteq \mathtt{Z} \;\&\; \mathtt{Z} : \mathtt{a} \vee \mathtt{c}$.

   This $\mathcal{OSF}$ normal form corresponds to the $\mathcal{FOT}$ $f(Y, Z, Z)$ as computed in Example 2.3 when the $\vee$ operation on two different functors returns $\top$ (*i.e.*, in this case: $\mathtt{a} \vee \mathtt{b} = \mathtt{a} \vee \mathtt{c} = \top$).

Example B.5 $\mathcal{FOT}$ **generalization using** $\mathcal{OSF}$ **rules** — Example B.4 could as well have started out by putting the initial terms in normal extensional form. Namely,

$$\begin{aligned}
\phi_1 \stackrel{\text{def}}{=}\; & \mathtt{X1} : \mathtt{f} \;\&\; \mathtt{X1} \,.\, 1 \doteq \mathtt{Y1} \\
& \;\&\; \mathtt{X1} \,.\, 2 \doteq \mathtt{Y1} \\
& \;\&\; \mathtt{X1} \,.\, 3 \doteq \mathtt{Y1} \;\&\; \mathtt{Y1} : \mathtt{a} \\
\phi_2 \stackrel{\text{def}}{=}\; & \mathtt{X2} : \mathtt{f} \;\&\; \mathtt{X2} \,.\, 1 \doteq \mathtt{Y2} \;\&\; \mathtt{Y2} : \mathtt{b} \\
& \;\&\; \mathtt{X2} \,.\, 2 \doteq \mathtt{Z2} \\
& \;\&\; \mathtt{X2} \,.\, 3 \doteq \mathtt{Z2} \;\&\; \mathtt{Z2} : \mathtt{c}.
\end{aligned}$$

As shown by the normalization below, this terminates with the same results (up to tag renaming) as the normalization above, although with a different rule sequence.

1. Start with:

- $\gamma_1 = \{\mathtt{X} = \mathtt{X1}\}$
- $\gamma_2 = \{\mathtt{X} = \mathtt{X2}\}$
- $[\underline{\mathtt{X1} : \mathtt{f}} \;\&\; \mathtt{X1} \,.\, 1 \doteq \mathtt{Y1} \;\&\; \mathtt{X1} \,.\, 2 \doteq \mathtt{Y1} \;\&\; \mathtt{X1} \,.\, 3 \doteq \mathtt{Y1} \;\&\; \mathtt{Y1} : \mathtt{a}$
  $\|$
  $\underline{\mathtt{X2} : \mathtt{f}} \;\&\; \mathtt{X2} \,.\, 1 \doteq \mathtt{Y2} \;\&\; \mathtt{Y2} : \mathtt{b} \;\&\; \mathtt{X2} \,.\, 2 \doteq \mathtt{Z2} \;\&\; \mathtt{X2} \,.\, 3 \doteq \mathtt{Z2} \;\&\; \mathtt{Z2} : \mathtt{c}]$

2. apply Rule **SORT INDUCTION**:

- $\gamma_1 = \{\mathtt{X} = \mathtt{X1}\}$
- $\gamma_2 = \{\mathtt{X} = \mathtt{X2}\}$
- $\mathtt{X} : \mathtt{f}$
  $[\underline{\mathtt{X1} \,.\, 1 \doteq \mathtt{Y1}} \;\&\; \mathtt{X1} \,.\, 2 \doteq \mathtt{Y1} \;\&\; \mathtt{X1} \,.\, 3 \doteq \mathtt{Y1} \;\&\; \mathtt{Y1} : \mathtt{a}$
  $\|$
  $\underline{\mathtt{X2} \,.\, 1 \doteq \mathtt{Y2}} \;\&\; \mathtt{Y2} : \mathtt{b} \;\&\; \mathtt{X2} \,.\, 2 \doteq \mathtt{Z2} \;\&\; \mathtt{X2} \,.\, 3 \doteq \mathtt{Z2} \;\&\; \mathtt{Z2} : \mathtt{c}]$

3. apply Rule **FEATURE INDUCTION**:

- $\gamma_1 = \{\mathtt{X} = \mathtt{X1}, \mathtt{Y} = \mathtt{Y1}\}$
- $\gamma_2 = \{\mathtt{X} = \mathtt{X2}, \mathtt{Y} = \mathtt{Y2}\}$
- $\mathtt{X} : \mathtt{f} \;\&\; \mathtt{X} \,.\, 1 \doteq \mathtt{Y}$
  $[\underline{\mathtt{X1} \,.\, 2 \doteq \mathtt{Y1}} \;\&\; \mathtt{X1} \,.\, 3 \doteq \mathtt{Y1} \;\&\; \mathtt{Y1} : \mathtt{a}$
  $\|$
  $\mathtt{Y2} : \mathtt{b} \;\&\; \underline{\mathtt{X2} \,.\, 2 \doteq \mathtt{Z2}} \;\&\; \mathtt{X2} \,.\, 3 \doteq \mathtt{Z2} \;\&\; \mathtt{Z2} : \mathtt{c}]$

4. apply Rule **FEATURE INDUCTION**:

- $\gamma_1 = \{\mathtt{X} = \mathtt{X1}, \mathtt{Y} = \mathtt{Y1}, \mathtt{Z} = \mathtt{Y1}\}$
- $\gamma_2 = \{\mathtt{X} = \mathtt{X2}, \mathtt{Y} = \mathtt{Y2}, \mathtt{Z} = \mathtt{Z2}\}$
- $\mathtt{X} : \mathtt{f} \;\&\; \mathtt{X} \,.\, 1 \doteq \mathtt{Y} \;\&\; \mathtt{X} \,.\, 2 \doteq \mathtt{Z}$
  $[\underline{\mathtt{X1} \,.\, 3 \doteq \mathtt{Y1}} \;\&\; \mathtt{Y1} : \mathtt{a}$
  $\|$
  $\mathtt{Y2} : \mathtt{b} \;\&\; \underline{\mathtt{X2} \,.\, 3 \doteq \mathtt{Z2}} \;\&\; \mathtt{Z2} : \mathtt{c}]$

5. apply Rule **COREFERENCE INDUCTION**:

- $\gamma_1 = \{\mathtt{X} = \mathtt{X1}, \mathtt{Y} = \mathtt{Y1}, \mathtt{Z} = \mathtt{Y1}\}$
- $\gamma_2 = \{\mathtt{X} = \mathtt{X2}, \mathtt{Y} = \mathtt{Y2}, \mathtt{Z} = \mathtt{Z2}\}$
- $\mathtt{X} : \mathtt{f} \;\&\; \mathtt{X} \,.\, 1 \doteq \mathtt{Y} \;\&\; \mathtt{X} \,.\, 2 \doteq \mathtt{Z} \;\&\; \mathtt{X} \,.\, 3 \doteq \mathtt{Z}$
  $[\underline{\mathtt{Y1} : \mathtt{a}}$
  $\|$
  $\underline{\mathtt{Y2} : \mathtt{b}} \;\&\; \mathtt{Z2} : \mathtt{c}]$

6. apply Rule **SORT INDUCTION**:

- $\gamma_1 = \{\mathtt{X} = \mathtt{X1}, \mathtt{Y} = \mathtt{Y1}, \mathtt{Z} = \mathtt{Y1}\}$
- $\gamma_2 = \{\mathtt{X} = \mathtt{X2}, \mathtt{Y} = \mathtt{Y2}, \mathtt{Z} = \mathtt{Z2}\}$
- $\mathtt{X} : \mathtt{f} \;\&\; \mathtt{X}\,.\,1 \doteq \mathtt{Y} \;\&\; \mathtt{X}\,.\,2 \doteq \mathtt{Z} \;\&\; \mathtt{X}\,.\,3 \doteq \mathtt{Z} \;\&\; \mathtt{Y} : \mathtt{a} \vee \mathtt{b}$
  $\left[\underline{\mathtt{Y1} : \mathtt{a}} \;\|\; \underline{\mathtt{Z2} : \mathtt{c}}\right]$

7. apply Rule **SORT INDUCTION**:

- $\gamma_1 = \{\mathtt{X} = \mathtt{X1}, \mathtt{Y} = \mathtt{Y1}, \mathtt{Z} = \mathtt{Y1}\}$
- $\gamma_2 = \{\mathtt{X} = \mathtt{X2}, \mathtt{Y} = \mathtt{Y2}, \mathtt{Z} = \mathtt{Z2}\}$
- $\mathtt{X} : \mathtt{f} \;\&\; \mathtt{X}\,.\,1 \doteq \mathtt{Y} \;\&\; \mathtt{X}\,.\,2 \doteq \mathtt{Z} \;\&\; \mathtt{X}\,.\,3 \doteq \mathtt{Z} \;\&\; \mathtt{Y} : \mathtt{a} \vee \mathtt{b} \;\&\; \mathtt{Z} : \mathtt{a} \vee \mathtt{c}$

Bibliography

[1] Ágnes Achs and Attila Kiss. Fuzzy extension of Datalog. *Acta Cybernetica*, 12(2):153–166, 1995. [Available online].

[2] Alfred Aho, John Hopcroft, and Jeffrey Ullmann. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.

[3] Hassan Aït-Kaci. *A Lattice-Theoretic Approach to Computation Based on a Calculus of Partially-Ordered Type Structures*. PhD thesis, Computer and Information Science, University of Pennsylvania, Philadelphia, PA (USA), September 1984. Abstract: [Available online].

[4] Hassan Aït-Kaci. An algorithm for finding a minimal recursive path ordering. *Revue d'Automatique, d'Informatique, et de Recherche Opérationnelle—Informatique théorique*, 19(4):359–382, 1985. [Available online].

[5] Hassan Aït-Kaci. An algebraic semantics approach to the effective resolution of type equations. *Theoretical Computer Science*, 45:293–351, 1986. [Available online].

[6] Hassan Aït-Kaci. *Warren's Abstract Machine—A Tutorial Reconstruction*. MIT Press, 1991. [Available online].

[7] Hassan Aït-Kaci. An introduction to $\mathcal{LIFE}$—Programming with Logic, Inheritance, Functions, and Equations. In Dale Miller, editor, *Proceedings of the International Symposium on Logic Programming*, pages 52–68. MIT Press, October 1993. [Available online].

[8] Hassan Aït-Kaci. Data models as constraint systems—A key to the Semantic Web. *Constraint Processsing Letters*, 1(1):33–88, November 2007. [Available online].

[9] Hassan Aït-Kaci. $\mathcal{HOOT}$ a language for expressing and querying $\mathcal{H}$ierarchical $\mathcal{O}$ntologies, $\mathcal{O}$bjects, and $\mathcal{T}$ypes—a specification. Technical Report Number 16, $\mathcal{CEDAR}$ Project, LIRIS, Département d'Informatique, Université Claude Bernard Lyon 1, Villeurbanne, France, December 2014. [Available online].

[10] Hassan Aït-Kaci. $\mathcal{FFF}$: a $\mathcal{F}$uzzy $\mathcal{F}$acility for $\mathcal{F}$irst-order terms. Java Software, August 2018. [Available online].

[11] Hassan Aït-Kaci and Samir Amir. Classifying and querying very large taxonomies with bit-vector encoding. *Journal of Intelligent Information Systems*, 45(2):1–25, October 2015. [Available online].

[12] Hassan Aït-Kaci, Robert Boyer, Patrick Lincoln, and Roger Nasr. Efficient implementation of lattice operations. *ACM Transactions on Programming Languages and Systems*, 11(1):115–146, January 1989. [Available online].

[13] Hassan Aït-Kaci and Roberto di Cosmo. Compiling order-sorted feature term unification. Technical Note 7, Digital Paris Research Laboratory, Rueil-Malmaison, France, December 1993. [Available online].

[14] Hassan Aït-Kaci, Bruno Dumant, Richard Meyer, Andreas Podelski, and Peter Van Roy. The Wild $\mathcal{LIFE}$ handbook. [Available online], 1994.

[15] Hassan Aït-Kaci and Roger Nasr. Logic and inheritance. In *Proceeedings of the* 13^{th} *ACM Sigplan Conference on Principles of Programming Languages (POPL 1986)*, pages 219–228, St. Petersburg Beach, FL (USA), January 1986. Association for the Computing Machinery, ACM. [Available online].

[16] Hassan Aït-Kaci and Gabriella Pasi. Lattice operations on terms over similar signatures. In Fabio Fioravanti and John Gallagher, editors, *Proceedings of the* 27^{th} *International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR 2017)*, Namur, Belgium, October 20-12 2017. [Available online].

[17] Hassan Aït-Kaci and Gabriella Pasi. Lattice operations on terms over similar signatures—a constraint-based approach. Presentation slides for the 27^{th} *International Symposium on Logic-Based Program Synthesis and Transformation* (LOPSTR 2017), October 10–12, 2017. [Available online].

[18] Hassan Aït-Kaci and Andreas Podelski. Towards a meaning of $\mathcal{LIFE}$. *Journal of Logic Programming*, 16(3-4):195–234, 1993. [Available online].

[19] Hassan Aït-Kaci, Andreas Podelski, and Seth C. Goldstein. Order-sorted feature theory unification. *Journal of Logic Programming*, 30(2):99–124, 1997. [Available online].

[20] Hassan Aït-Kaci, Andreas Podelski, and Gert Smolka. A feature-based constraint system for logic programming with entailment. *Theoretical Computer Science*, 122(1–2):263–283, January 1994. [Available online].

[21] Hassan Aït-Kaci and Yutaka Sasaki. An axiomatic approach to feature term generalization. In Luc de Raedt and Peter Flach, editors, *Proceedings of the* 12^{th} *European Conference on Machine Learning (ECML'01)*, pages 1–12, Berlin Heidelberg, September 2001. LNCS 2167, Springer-Verlag. [Available online].

[22] Srinivas M. Aji. *Graphical Models and Iterative Decoding*. PhD thesis, California Institute of Technology, Pasadena, CA, USA, May 2000. [Available online].

[23] Srinivas M. Aji and Robert J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, March 2000. [Available online].

[24] María Alpuente, Santiago Escobar, Javier Espert, and José Meseguer. A modular order-sorted equational generalization algorithm. *Information and Computation*, 235:98–136, 2014. [Available online].

[25] María Alpuente, Santiago Escobar, José Meseguer, and Pedro Ojeda. Order-sorted generalization. *Electronic Notes in Theoretical Computer Science*, 246:27–38, 2009. [Available online].

[26] Teresa Alsinet, Llús Godo, and Sandra Sandri. Two formalisms of extended possibilistic logic programming with context-dependent fuzzy unification: a comparative description. *Electronic Notes in Theoretical Computer Science*, 66(5):21 pages, 2002. [Available online].

[27] Krzysztof R. Apt. Logic programming. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 492–574. Elsevier, 1990.

[28] Francesca Arcelli and Ferrante Formato. A fuzzy logic programming language. In Luca Console, Evelina Lamma, Paola Mello, and Michela Milano, editors, *Programmazione Logica e Prolog*, pages 319–332. UTET Università, 1997. [Available online].

[29] Francesca Arcelli and Ferrante Formato. Likelog: A logic programming language for flexible data retrieval. In Hisham Al Haddad, editor, *Proceedings of the 1999 ACM Symposium on Applied Computing*, pages 260–267, San Antonio, TX (USA), February 28–March 2, 1999. Association for Computing Machinery, ACM. [Available online].

[30] Francesca Arcelli, Ferrante Formato, and Giangiacomo Gerla. Extending unification through similarity relations. *Bulletin pour les Sous Ensembles Flous et leurs Applications (BUSEFAL)*, pages 3–12, 1997. [Available online].

[31] Francesca Arcelli-Fontana. Likelog for flexible query answering. *Soft Computing*, 7(2):107–114, December 2002.

[32] Francesca Arcelli-Fontana and Ferrante Formato. A similarity-based resolution rule. *International Journal of Intelligent Systems*, 17:853–872, 2002. [Available online].

[33] Peter R.J. Asveld. Algebraic aspects of families of fuzzy languages. *Theoretical Computer Science*, 293(2):417–445, February 2003. [Available online].

[34] Franz Baader and Ulrike Sattler. Description logics with aggregates and concrete domains. In *Proceedings of the International Workshop on Description Logics*, Gif sur Yvette, France, 1997. [Available online].

[35] Rolf Backofen. Regular path expressions in feature logic. In Claude Kirchner, editor, *Proceedings of the 5th International Conference on Rewriting Techniques and Applications (RTA'93)*, pages 121–135, Montreal, QC (Canada), June 16–8, 1993. Springer. LNCS 690, [Available online].

[36] James F. Baldwin, Jonathan Lawry, and Trevor P. Martin. Efficient algorithms for semantic unification. In *Proceedings of the 6^{th} International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU '96)*, pages 527–532, Granada, Spain, July 1–5, 1996. [Available online].

[37] James F. Baldwin and B. W. Pilsworth. Semantic unification with fuzzy concepts in FRIL. *International Journal of Intelligent Systems*, 7(1):61–69, January 1992. [Available online].

[38] Mustapha Baziz, Mohand Boughanem, Gabriella Pasi, and Henri Prade. A fuzzy set approach to concept-based information retrieval. In E. Montseny and P. Sobrevilla, editors, *Proceedings of the Joint 4^{th} Conference of the European Society for Fuzzy Logic and Technology and the* 11èmes Rencontres Francophones sur la Logique Floue et ses Applications*, Barcelona (Spain)*, pages 1287–1292. EUSFLAT/LFA, September 7–9, 2005. [Available online].

[39] Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, September 2, 1954. Invited address at the annual summer meeting of the AMS, Laramie, WY (USA) [Available online].

[40] Garrett Birkhoff. *Lattice Theory*, volume 25 of *Colloquium Publications*. American Mathematical Society, Providence, RI (USA), 1940, revised 1948, 1967, 1979, and 1979. [Available online].

[41] Garrett Birkhoff and Jonh Non Neumann. The logic of quantum mechanics. *Annals of Mathematics*, 37(4):823–843, October 1936. [Available online].

[42] Gloria Bordogna, Dario Lucarella, and Gabriella Pasi. A fuzzy object oriented data model. In *Proceedings of the 3rd IEEE Conference on Fuzzy Systems*. IEEE World Congress on Computational Intelligence, July 1994. [Available online].

[43] Ronald J. Brachman. *A Structural Paradigm for Representing Knowledge*. PhD thesis, Artificial Intelligence, Harvard University, Cambridge, MA (USA), 1977. Available as BBN Technical Report 3605 from Bolt Beranek and Newman Inc. (1978).

[44] Tru H. Cao and Peter N. Creasy. Fuzzy types: a framework for handling uncertainty about types of objects. *International Journal of Approximate Reasoning*, 25(3):217–253, November 2000. [Available online].

[45] Tru Hoang Cao and Jonathan Michael Rossiter. FRIL++ for machine learning. 2002. [Available online].

[46] Yongzhi Cao and Yoshinori Ezawac. Nondeterministic fuzzy automata. *Information Sciences*, 191:86–97, 2012. [Available online].

[47] Bob Carpenter. Typed feature structures: A generalization of first-order terms. In Vijay Saraswat and Kazunori Ueda, editors, *Proceedings of the 1991 International Symposium on Logic Programming*, pages 187–201, Cambridge, MA, 1991. MIT Press.

[48] Stefano Ceri, Georg Gottlob, and Letizia Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE Transactions on Knowledge and Data Engineering*, 1(1):146–166, March 1989. [Available online].

[49] Aarthi Chandramohan and M. V. C. Rao. Novel, useful, and effective definitions for fuzzy linguistic hedges. *Discrete Dynamics in Nature and Society*, 2006(Article ID 46546):1–13, 2006. [Available online].

[50] William F. Clocksin and Christopher S. Mellish. *Programming in Prolog*. Springer-Verlag, 2nd edition, 1984.

[51] Alain Colmerauer. Prolog and infinite trees. In Keith L. Clark and Sten Åke Tärnlund, editors, *Logic Programming*, pages 231–251. Academic Press, 1982.

[52] Maria Eugenia Cornejo, Jesús Medina Moreno, and Clemente Rubio-Manzano. Towards a full fuzzy unification in the bousi prolog system. In Fernando Gomide, editor, *Proceedings of the International Conference on Fuzzy Systems (FUZZ-IEEE 2018)*, July 2018. [Available online].

[53] Andreas de Vries. Algebraic hierarchy of logics unifying fuzzy logic and quantum logic. *ArXiv e-prints*, 0707(2161), July 2007. [Available online].

[54] Erik D. Demaine, Shay Mozes, Benjamin Rossman, and Oren Weimann. An optimal decomposition algorithm for tree edit distance. *ACM Transactions on Algorithms*, 6(1):article 2, December 2009. [Available online].

[55] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science—Volume B: Formal Methods and Semantics*, chapter 6, pages 243–320. North-Holland, Amsterdam, The Netherlands, 1990. [Available online].

[56] Didier Dubois and Henri Prade. *Fuzzy Sets and Systems: Theory and Applications*, volume 144 of *Mathematics in Science and Engineering, Edited by William F. Ames, Georgia Institute of Technology*. Academic Press, 180. [Available online].

[57] Santiago Escobar, José Meseguer, and Prasanna Thati. Narrowing and rewriting logic: from foundations to applications. *Electronic Notes in Theoretical Computer Science*, 177:5–33, 2007. [Available online].

[58] M.J. Fay. First order unification in an equational theory. In W.H. Joyner, editor, *Proceedings of the 4^{th} Workshop on Automated Deduction (Austin, TX)*, pages 161–167, London, UK, 1979. Academinc Press.

[59] Leonidas Fegaras. Query unnesting in object-oriented databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pages 49–60, Seattle, WA (USA), June 2–4 1998. [Available online].

[60] Leonidas Fegaras and David Maier. Optimizing object queries using an effective calculus. *ACM Transactions on Database Systems*, 25(4):457–516, December 2000. [Available online].

[61] David Gilbert and Michael Schroeder. FURY: Fuzzy unification and resolution based on edit distance. In Nikolaos G. Bourbakis, editor, *Proceedings of the 1^{st} IEEE International Symposium on Bioinformatics and Biomedical Engineering (BIBE 2000)*, pages 330–336, Arlington, VA (USA), November 8–10, 2000. IEEE Computer Society. [Available online].

[62] Joseph Goguen. What is unification? In Hassan Aït-Kaci and Maurice Nivat, editors, *Resolution of Equations in Algebraic Structures—Algebraic Techniques*, chapter 6, pages 217–261. Academic Press, 1989. [Available online].

[63] Joseph A. Goguen. L-fuzzy sets. *Journal of Mathematical Analysis and Applications*, 18:145–174, 1967. [Available online].

[64] Joseph A. Goguen, James W. Thatcher, Eric G. Wagner, and Jesse B. Wright. Initial algebra semantics and continuous algebras. *Journal of the Association for Computing Machinery*, 24(1):68–95, January 1977. [Available online].

[65] Lovre Grisogono. Classical logic *vs.* quantum logic. Lecture slides, University of Zagreb, Croatia, July 8–9 2013. [Available online].

[66] Torsten Grust. A versatile representation for queries. In P.M.D. Gray, L. Kerschberg, P.J.H. King, and A. Poulovassilis, editors, *The Functional Approach to Data Management: Modeling, Analyzing and Integrating Heterogeneous Data*. Springer, September 2003. [Available online].

[67] Hyowon Gweon, Joshua B. Tenenbaum, and Laura E. Schulz. Infants consider both the sample and the sampling process in inductive generalization. In Susan E. Carey, editor, *Proceedings of the National Academy of Sciences*, pages 9066–9071, Harvard University, Cambridge, MA (USA), May 18, 2010. PNAS vol. 107, no. 20, National Academy of Sciences of the United States of America. [Available online].

[68] Jacques Herbrand. *Recherches sur la théorie de la démonstration*. PhD thesis, Faculté des sciences de l'université de Paris, Paris (France), 1930. [Available online]; English translation in [69].

[69] Jacques Herbrand. *Logical Writings*. Harvard University Press, Cambridge, MA, 1971. Edited by Warren D. Goldfarb.

[70] Markus Höhfeld and Gert Smolka. Definite relations over constraint languages. LILOG Report 53, IWBS, IBM Deutschland, Stuttgart, Germany, October 1988. [Available online].

[71] Jelena Ignjatović and Miroslav Ćirić. Myhill-nerode theory for fuzzy languages and automata. Lecture Slides, AUTOMATHA Workshop on Algebraic Theory of Automata and Logic, Szeged, Hungary[1], September 30–October 1 2006. [Available online].

[72] Pascual Julián Iranzo. A procedure for the construction of a similarity relation. In Luis Magdalena, Manuel Ojeda-Aciego, and José-Luis Verdegay, editors, *Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU '08)*, pages 489–496, Málaga, Spain, June 22–27, 2008. [Available online].

[73] Pascual Julián Iranzo and Clemente Rubio Manzano. A similarity-based WAM for Bousi∼Prolog. In *Proceedings of the 10th International Work-Conference on Artificial Neural Networks (IWANN '09)—Part I: Bio-Inspired Systems: Computational and Ambient Intelligence*, pages 245–252, Salamanca, Spain, June 10–12, 2009. Springer. [Available online].

[74] Pascual Julián Iranzo and Clemente Rubio Manzano. An efficient fuzzy unification method and its implementation into the Bousi∼Prolog system. In *Proceedings of 19th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2010)*, pages 658–665, Barcelona, Spain, July 18–23, 2010. IEEE. [Available online].

[75] Pascual Julián Iranzo and Clemente Rubio Manzano. A sound and complete semantics for a similarity-based logic programming language. *Fuzzy Sets and Systems*, 317:1–26, 2017.

[76] Pascual Julián Iranzo, Clemente Rubio Manzano, and Juan Gallardo-Casero. Bousi∼Prolog: a Prolog extension language for flexible query answering. *Electronic Notes in Theoretical Computer Science*, 248:131–147, August 5, 2009. [Available online].

[77] Pascual Julián Iranzo, Ginés Moreno, Jaime Penabad, and Carlos Vázquez. A fuzzy logic programming environment for managing similarity and truth degrees. In Santiago Escobar, editor, *Proceedings XIV Jornadas sobre Programación y Lenguajes (PROLE 2014)*, pages 71–86. Electronic Proceedings in Theoretical Computer Science, EPTCS 173, January 2015. [Available online].

[78] Joxan Jaffar. Efficient unification over infinite terms. *New Generation Computing*, 2(3):207 – 219, September 1984. [Available online].

[1] http://www.inf.u-szeged.hu/~csl06/ws.php

[79] Joxan Jaffar and Jean-Louis Lassez. Constraint logic programming. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages*, Munich, W. Germany, January 1987.

[80] Petteri Jokinen, Jorma Tarhio, and Esko Ukkonen. A comparison of approximate string matching algorithms. *Software–Practice and Experience*, 1(1988):1–4, January 1. [Available online].

[81] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object oriented and frame based languages. *Journal of the ACM*, 42(4):741–843, 1995. [Available online].

[82] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, UK, 1970. Reprinted in *Automatic Reasoning*, 2, Springer-Verlag, pp. 342–276 (1983).

[83] Robert A. Kowalski. *Logic for Problem Solving*, volume 7 of *Artificial Intelligence Series*. North Holland, New York, NY, 1979.

[84] Temur Kutsia. Pattern unification with sequence variables and flexible arity symbols. *Electronic Notes in Theoretical Computer Science*, 66(5):52–69, 2002. [Available online].

[85] Temur Kutsia, Jordi Levy, and Mateu Villaret. Anti-unification for unranked terms and hedges. *Journal of Automated Reasoning*, 52(2):155–190, 2014. [Available online].

[86] Simon Lacoste-Julien, Konstantina Palla, Alex Davies, Gjergji Kasneci, Thore Graepel, and Zoubin Ghahramani. SiGMa: Simple greedy matching for aligning large knowledge bases. In Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy, editors, *Proceedings of the 19th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2013—Chicago, IL, USA)*, pages 572–580, New York, NY (USA), August 11–14, 2013. Association for Computing Machinery, ACM. [Available online]; see also [Available online].

[87] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics–Doklady, Cybernetics and Control Theory*, 10(8):707–710, February 1966. [Available online].

[88] Ravi K. M. *Some Investigations in Fuzzy Automata*. PhD thesis, Jaypee Institute of Information Technology, Noida (India), February 2012. [Available online].

[89] Bernd Mahr and Johann A. Makowsky. Characterizing specification languages which admit initial semantics. *Theoretical Computer Science*, 31(1–2):49–59, 1984. [Available online].

[90] Ebrahim E. Mamdani. Application of fuzzy logic to approximate reasoning using linguistic synthesis. *IEEE Transactions on Computers*, C-26(12):1182–1191, December 1977. [Available online].

[91] Ebrahim E. Mamdani and Seto Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, January 1975. [Available online].

[92] Alberto Martelli and Ugo Montanari. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4(2):258–282, April 1982. [Available online].

[93] Takashi Mitsuishi and Grzegorz Bancerek. Lattice of fuzzy sets. *Formalized Mathematics*, 11(4):393–398, 2003. [Available online].

[94] Jiří Močkoř. Fuzzy and non-deterministic automata. Research Report 8, University of Ostrava, Institute for Research and Applications of Fuzzy Modeling, Ostrava (Czech Republic), November 6, 1997. [Available online].

[95] Jiří Močkoř. Fuzzy and non-deterministic automata. *Soft Computing*, 3(4):221–226, December 1999. See also [94].

[96] Santiago Onta nón and Enric Plaza. Similarity measures over refinement graphs. *Machine Learning*, 87(1):57–92, 2012. [Available online].

[97] Robert A. Orchard. *FuzzyCLIPS Users Guide*. Integrated Reasoning Group, Institute for Information Technology, National Research Council, Ottawa, ON (Canada), version 6.10d edition, October 2004. [Available online].

[98] Juiyao Pan, Guilherme N. DeSouza, and Avinash C. Kak. A large-scale expert system shell using fuzzy logic for uncertainty reasoning. *IEEE Transactions on Fuzzy Systems*, 6(4):563–581, November 1998. [Available online].

[99] Benjamin Pierce. *Basic Category Theory for the Computer Scientists*. MIT Press, 1991.

[100] Gordon D. Plotkin. Lattice theoretic properties of subsumption. Technical Memo MIP-R-77, Department of Machine Intelligence and Perception, University of Edinburgh, Edinburgh, Scotland (UK), June 1970.

[101] Gordon D. Plotkin. A note on inductive generalization. In Bernard Metzer and Donald Michie, editors, *Machine Intelligence 5*, chapter 8, pages 154–163. Edinburgh University Press, Edinburgh, Scotland (UK), 1970. [Available online].

[102] Jarosław Pykacz. Fuzzy quantum logic I. *International Journal of Theoretical Physics*, 32(10):1691–1708, October 1993.

[103] Jarosław Pykacz. Quantum structures and fuzzy set theory. In Kurt Engesser, Dov M. Gabbay, and Daniel Lehmann, editors, *Handbook of Quantum Logic and Quantum Structures—Quantum Structures*, pages 55–74. Elsevier, 2007. [Available online].

[104] John C. Reynolds. Transformational systems and the algebraic nature of atomic formulas. In Bernard Metzer and Donald Michie, editors, *Machine Intelligence 5*, chapter 7, pages 135–151. Edinburgh University Press, Edinburgh, Scotland (UK), 1970. [Available online].

[105] John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41, January 1965. [Available online].

[106] Jonathan M. Rossiter, T.H. Cao, T. P. Martin, and James F. Baldwin. A FRIL++ compiler for soft computing object-oriented logic programming. In *Proceedings of the 6th International Conference on Soft Computing*, pages 340–345, Iizuka, Fukuoka, Japan, October 1–4, 2000. [Available online].

[107] Jonathan Michael Rossiter and Tru Hoang Cao. FRIL++ and its applications. In Zongmin Ma, editor, *Advances in Fuzzy Object-Oriented Databases: Modeling and Applications*, chapter IV, pages 113–151. Idea Group, Inc., 2005.

[108] Yutaka Sasaki. Applying type oriented ILP to IE rule generation. In *Proceedings of the Workshop on Machine Learning for Information Extraction*, pages 43–47, Orlando, FL (USA), 1999. AAAI-99. [Available online].

[109] Yutaka Sasaki. Induction logic based on ψ-terms. In *Proceedings of the 10th International Conference on Algorithmic Learning Theory*, pages 169–181, Tokyo, Japan, 1999. ALT-99, Springer-Verlag LNAI 1720.

[110] Yutaka Sasaki. *Hierarchically Sorted Inductive Logic Programming and its Application to Information Extraction*. PhD thesis, Graduate School of Systems and Information Engineering, University of Tsukuba, Japan, September 2000.

[111] Yutaka Sasaki and Masahiko Haruno. RHB$^+$: A type oriented ILP system learning from positive data. In *Proceedings of IJCAI-97*, pages 894–899, Nagoya, Japan, 1997. [Available online].

[112] Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.

[113] Ingo Schmitt, Andreas Nürnberger, and Sebastian Lehrack. On the relation between fuzzy and quantum logic. In Rudolf Seising, editor, *Views on Fuzzy Sets and Systems from Different Perspectives—Philosophy and Logic, Criticisms and Applications*, Studies in Fuzziness and Soft Computing, chapter 20, pages 417–438. Springer, March 2009. [Available online].

[114] Michael Schroeder. Re: your work on FURY. Private Communication (email), October 20, 2017. [Available online].

[115] Michael Schroeder and Ralf Schweimeier. Arguments and misunderstandings: Fuzzy unification for negotiating agents. *Electronic Notes in Theoretical Computer Science*, 70(5):1–19, October 2002. [Available online].

[116] Ralf Schweimeier and Michael Schroeder. Fuzzy unification and argumentation for well-founded semantics. In Peter Van Emde Boas and Július Štuller Jaroslav Pokorný, Mária Bieliková, editors, *Proceedings of the 30th Conference on Current Trends in Theory and Practice of Computer Science*, pages 102–121, Měřín, Czech Republic, 24–30 January 2004. LNCS 2932, Springer, Lecture Notes in Computer Science. [Available online].

[117] Maria I. Sessa. Approximate reasoning by similarity-based SLD resolution. *Theoretical Computer Science*, 275:389–426, 2002. [Available online].

[118] Umberto Straccia. A fuzzy description logic. In Jack Mostow and Chuck Rich, editors, *Proceedings of the 15thNational Conference on Artificial Intelligence*, Madison, WI (USA), 1998. American Association for Artificial Intelligence. [Available online].

[119] Umberto Straccia. Reasoning within fuzzy description logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001. [Available online].

[120] S.P. Tiwari, Anupam K. Singh, and Shambhu Sharan. Fuzzy subsystems of fuzzy automata based on lattice-ordered monoid. *Annals of Fuzzy Mathematics and Informatics*, 7(3):437–445, March 2013. [Available online].

[121] S.P. Tiwari, Vijay K. Yadav, and Anupam K. Singh. On algebraic study of fuzzy automata. *International Journal of Machine Learning and Cybernetics*, 6(3):479–485, June 2015.

[122] Maarten H. van Emden and John W. Lloyd. A logical reconstruction of Prolog II. *Journal of Logic Programming*, 2:143–149, 1984. [Available online].

[123] Claudio Vaucheret, Sergio Guadarrama, and Susana Mu noz Hernández. Fuzzy Prolog: a simple general implementation using CLP($\mathbb{R}$). In Matthias Baaz and Andrei Voronkov, editors, *Proceedings of the 9th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2002)*, pages 450–464, Tbilisi, Georgia, October 14–18, 2002. LNCS 2514, Springer. [Available online].

[124] Enrique Vidal, Andrés Marzal, and Pablo Aibar. Fast computation of normalized edit distances. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(9):899–902, Septembre 1995. [Available online].

[125] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, January 1974. [Available online].

[126] Stephen Warshall. A theorem on Boolean matrices. *Journal of the ACM*, 9(1):11–12, January 1962.

[127] Kevin Wayne. Union-find. Tutorial lecture slides based on book "Algorithm Design" by Jon Kleinberg and Éva Tardos (Addison-Wesley, 2015). [Available online].

[128] Raymond T. Yeh. Toward an algebraic theory of fuzzy relational systems. Technical Report TR-25, Department of Computer Sciences and Electronics Research Center, the University of Texas at Austin, Austin, TX (USA), July 1973. [Available online].

[129] John Yen. Generalizing term subsumption languages to fuzzy logic. In John Mylopoulos and Raymond Reiter, editors, *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 472–477, Sydney, Australia, August 24–30, 1991. IJCAI, Morgan Kaufmann. [Available online].

[130] Lotfi A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965. [Available online].

[131] Lotfi A. Zadeh. Similarity relations and fuzzy orderings. *Information Sciences*, 3:177–200, 1971. [Available online].

[132] Lotfi A. Zadeh. A fuzzy-set-theoretic interpretation of linguistic hedges. *Journal of Cybernetics*, 2(3):4–34, 1972. [Available online].

[133] Lotfi A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(1):28–44, January 1973. [Available online].

[134] Hans-Jürgen Zimmermann. *Fuzzy Set Theory—and Its Applications*. Springer Science+Business Media, New York, NY (USA), fourth edition, 2001. [Available online].