DRAFT—PLEASE DO NOT DISTRIBUTE

Fuzzy Lattices of Order-Sorted Feature Graphs

Hassan Aït-Kaci
HAK Language Technologies
hak@acm.org

Gabriella Pasi
Universitá de Milano-Bicocca
pasi@disco.unimib.it

March 1, 2020

Abstract

In this article, we provide a detailed formal study of the fuzzy interpretation of the two most important operations used in Knowledge Representation and Automated Reasoning; namely, unification and generalization of data and knowledge structures expressed as labeled graphs. The latter are composed of nodes and arrows between them. These components bear information: a node is labeled with a *sort* and an arrow is labeled with a *feature*. Formally, a sort denotes a *set* and a feature denotes a *function*. Semantically consistent such graphs have to respect feature composition (*i.e.*, all feature paths between two nodes must denote the same function). These graphs can then be seen as set-theoretic *commutative diagrams* of functions between sets. An "*is a*" partial order on sorts denoting set inclusion extends formally in a natural way to one on rooted labeled graphs,[a] whereby the features of a sort are inherited by any of its subsorts and respect sorts and path equalities. For this reason, these are called *Order-Sorted Feature* ($\mathcal{OSF}$) graphs. Furthermore, a lattice on sorts extends to a lattice on $\mathcal{OSF}$ graphs with two operations: unification is deriving the most general lower bound, and generalization is deriving the most specific upper bound, of two $\mathcal{OSF}$ graphs. These lattice operations on $\mathcal{OSF}$ graph structures — and variations thereof; such as, *e.g.*, the weaker notion of First-Order Term ($\mathcal{FOT}$) — have been used with great success for the past few decades in symbolic AI when made operational as constraint-solving systems. Recently, the authors developed an extension to Fuzzy Algebra of the traditional lattice of $\mathcal{FOT}$s when function symbols and argument positions may be comparable with a fuzzy equivalence relation (*a.k.a.*, *similarity*). This present paper elaborates on that study to provide yet a further extension to fuzzy lattice operations on $\mathcal{OSF}$ graph structures when the ordering on sorts may be fuzzy. Then, similarity between sorts is derived as the symmetric closure of their fuzzy sort ordering. This results in a generic formal and operational lattice calculus of fuzzy commutative diagrams for approximate deduction and abstraction.

Keywords: Approximate Information Processing; Lattice Algebra; Order-Sorted Feature Graphs; Fuzzy Knowledge Representation; Fuzzy Automated Reasoning; Fuzzy Unification; Fuzzy Generalization; Fuzzy Pattern-directed Reasoning; Fuzzy Pattern Induction; Fuzzy Machine Learning.

[a] A *rooted* labeled graph is any graph with a distinct node called its *root*, from which all other nodes in the graph are reachable.

DRAFT—PLEASE DO NOT DISTRIBUTE

1 Introduction

A First-Order Term ($\mathcal{FOT}$) 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-ordered sorts and with features, can be shown to extend those on more restricted kinds of rooted graphs such as $\mathcal{FOT}$s, labeled with functors, positions, and variables.

In this paper, we extend the results we recently reported in [AKP20] fuzzifying lattice operations on $\mathcal{FOT}$s modulo constructor similarity to rooted $\mathcal{OSF}$ graphs modulo sort similarity. In Section 2, we show how to fuzzify $\mathcal{OSF}$ graph subsumption and its lattice operations: in Section 2.2 we explicate fuzzy $\mathcal{OSF}$ graph unification, and in Section 2.3 fuzzy $\mathcal{OSF}$ graph generalization. In Section 3, we discuss issues related to the implementation of fuzzy $\mathcal{OSF}$ lattice operations using sort encoding.

2 Fuzzifying $\mathcal{OSF}$-Term Subsumption

As was done for First-Order Terms ($\mathcal{FOT}$s) in [AKP20], we here turn to fuzzifying lattice operations over $\mathcal{OSF}$ terms. We shall proceed as we did for $\mathcal{FOT}$s, up to partially ordered sort and unconstrained symbol feature arities. So let us start by making some important observations regarding some advantages in our approach to fuzzifying $\mathcal{FOT}$ unification and generalization.

1. The term structure itself (its syntax) is not fuzzified. For unification, only a conjunctive set E of equations (pairs of first-order terms — including substitutions) is given a similarity degree α. This is denoted as the fuzzy-weighted set E_α. For generalization, only a pair of tag substitutions in a judgment is given such a similarity degree α.

2. In unification rules the similarity degree of a conjunctive set of equations, and in generalization rule and axioms of a pair of tag mappings, can never increase from prior to posterior forms.

3. There is a similarity relation $\sim$ on functors f and g (as a half-matrix of similarity degrees in $[0.0, 0.1]$).[2]

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 $\alpha \in (0.0, 1.0]$ 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.

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

[2] Only one direction is needed; the other is equal by symmetry — see [AKP20].

Copyright © 2020 by the Authors All Rights Reserved

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 normalizing a $\mathcal{FOT}$ equation set or proving a $\mathcal{FOT}$ generalization judgment.

When considering $\mathcal{OSF}$ terms, we can proceed similarly to enforce constraint consistency when subsumption is realized by endomorphic tag mappings, which are sets of variable/variable equations — *i.e.*, $X \doteq Y$ — respecting sorts and feature application. These rules and axioms operate taking into account the following observations.

1. The $\mathcal{OSF}$ term itself is not fuzzified. For $\mathcal{OSF}$ unification, 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. For $\mathcal{OSF}$ generalization, only a pair of tag substitutions in a judgment is given such a similarity degree α.

2. In unification rules the similarity degree of a conjunctive set of atomic $\mathcal{OSF}$ constraints, and in generalization rule and axioms of a pair of $\mathcal{OSF}$ tag mappings, can never increase from prior to posterior forms.

3. There is a similarity relating pairs of sorts s and s' as a half-matrix of similarity degrees in $[0.0, 0.1]$.

So this looks pretty much the same as for $\mathcal{FOT}$s, except for one important detail: in the case of $\mathcal{OSF}$ terms, a similarity relation on a signature of partially ordered featured sorts must be also consistent with the ordering $\preceq$ on sorts. This means that, for all sorts s, s', t, t' in $\mathcal{S}$, the following *fuzzy sort-lattice consistency conditions* must hold for **lub**s and **glb**s when they exist:

$$\left.\begin{array}{l} \textbf{if } s \sim_\alpha s' \textbf{ and } t \sim_\beta t' \textbf{ then } (s \curlywedge t) \sim_{\alpha\wedge\beta} (s' \curlywedge t'), \\ \textbf{if } s \sim_\alpha s' \textbf{ and } t \sim_\beta t' \textbf{ then } (s \curlyvee t) \sim_{\alpha\wedge\beta} (s' \curlyvee t'); \end{array}\right\} \quad (1)$$

Note that the similarity degree in both foregoing fuzzy sort-lattice consistency conditions on the lattice operations on sorts, uses fuzzy conjunction ($\wedge$) of approximation degrees. While this may be expected for $\curlywedge$, it could appear odd for $\curlyvee$. However, it is correct to use $\wedge$ in both cases as we do because the homomorphic constraints expressed by Condition (1) apply to the *logical conjunction* "**and**" in the statement combining their premisses. In fact, the following (incorrect) constraint:

$$\textbf{if } s \sim_\alpha s' \textbf{ and } t \sim_\beta t' \textbf{ then } (s \curlyvee t) \sim_{\alpha\vee\beta} (s' \curlyvee t')$$

could make the fuzzy degree of an expression resulting from the fuzzy conjunction of two fuzzy expressions be greater than the degree of each — which is, again, incoherent since conjoining more fuzzy information can only decrease the resulting overall fuzzy degree.

In particular, a consequence of the fuzzy sort-lattice consistency conditions (1) is the following *fuzzy sort-order consistency condition* for any sorts $s, t, s', t' \in \mathcal{S}$ such that $s \preceq t$ and $s' \preceq t'$:

$$\textbf{if } s \sim_\alpha s' \textbf{ and } t \sim_\beta t' \textbf{ then } s \sim_{\alpha\wedge\beta} s'. \quad (2)$$

Copyright © 2020 by the Authors All Rights Reserved

This is illustrated generically in Figure 1 with abstract sorts and similarity weights, and with possible specific sorts and similarity weights defining an instance case pictured as Figure 2.

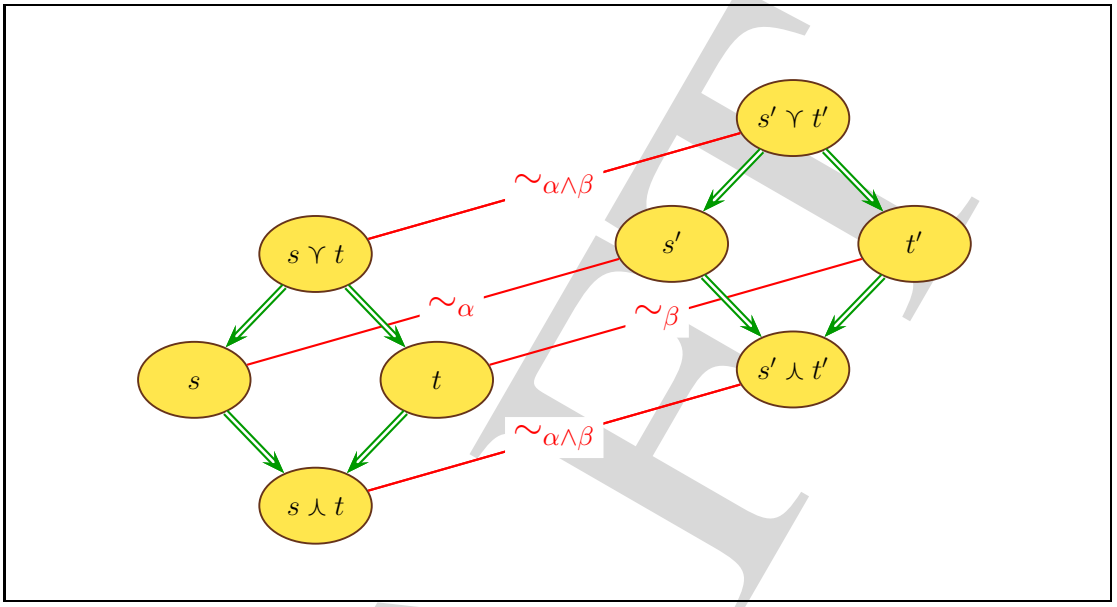

Figure 1: Order-inconsistent sort similarity

For example, given a similarity on sorts such that:

$$\texttt{employee} \sim_{.8} \texttt{assistant}$$
$$\texttt{student} \sim_{.9} \texttt{apprentice}$$

and an ordering on sorts such that:

$$\texttt{helper} \stackrel{\text{def}}{=} \texttt{assistant} \curlyvee \texttt{apprentice}$$
$$\texttt{intern} \stackrel{\text{def}}{=} \texttt{assistant} \curlywedge \texttt{apprentice}$$

and

$$\texttt{staff} \stackrel{\text{def}}{=} \texttt{employee} \curlyvee \texttt{student}$$
$$\texttt{working-student} \stackrel{\text{def}}{=} \texttt{employee} \curlywedge \texttt{student}$$

then, necessarily for a consistent set of sorts, it must be that:

$$\texttt{staff} \sim_{.8} \texttt{helper}.$$

But then, since **student** is a subsort of **staff** and since **apprentice** is a subsort of **helper**, order-consistency entails that the $\sim_{.8}$ similarity is inherited by all their respective subsorts. In particular, this implies **student** $\sim_{.8\wedge.9}$ **apprentice**; namely, **student** $\sim_{.8}$ **apprentice**. Similarly, order-consistency mandates that:

$$\texttt{working-student} \sim_{.8} \texttt{intern}.$$

Copyright © 2020 by the Authors All Rights Reserved

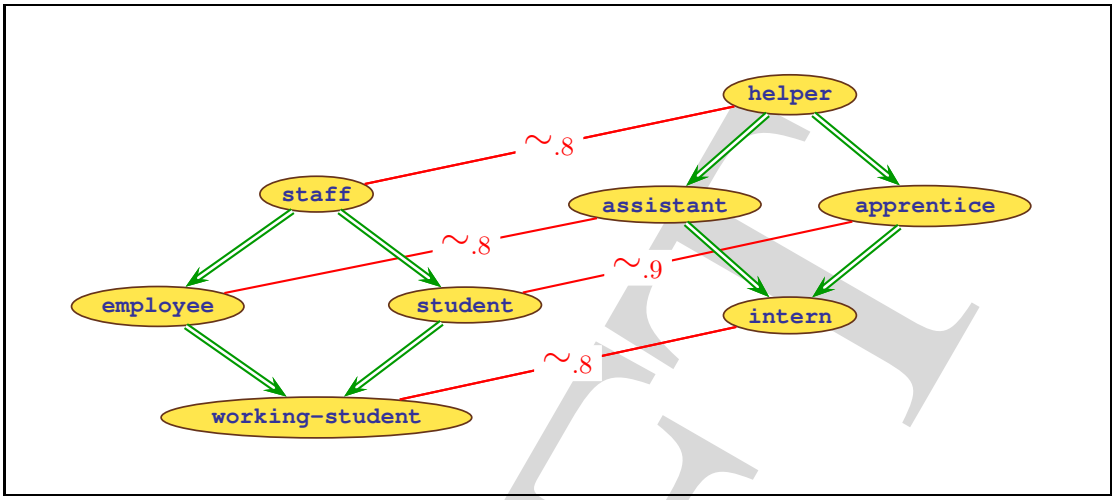

Figure 2: Order-inconsistent sort similarity example

Therefore, in order to ensure that a sort similarity $\sim$ is always consistent with the subsort ordering for all degree $\alpha \in \textbf{DEGREES}^{\sim}$, this necessitates that after declaring a few pairs of sorts to be similar at a given approximation degree, all their respective subsorts must also undergo an order-consistency closure. This closure consists in propagating similarities of all pairs of sorts to their subsorts as mandated by Condition (2). This is made formally precise next, while related considerations regarding how to compute and implement the fuzzy transitive closure of pairs of sorts that are declared similar in a fuzzy taxonomy are discussed in Section 3.

2.1 Fuzzy *vs.* subsort approximation

Before we proceed into further technicalities concerning fuzzy $\mathcal{OSF}$ term-lattice operations as fuzzy-constraint solving, let us discuss important implications of what a fuzzy ordering on sorts means. Then, using what we understand this to mean formally, let us make some specific remarks that will be helpful in understanding and justifying the correctness of the constraint-solving rules and axioms we shall propose.

A fuzzy ordering $\preceq$ on the set of sorts $\mathcal{S}$ means by definition that it is a fuzzy relation on $\mathcal{S}$ that is reflexive, anti-symmetric, and transitive. This implicitly defines the following fuzzy relations on $\mathcal{S}$:

- a similarity $\sim$ on $\mathcal{S}$ defined, for any $\alpha \in [0.0, 0.1]$, as:

$$\sim_\alpha \stackrel{\text{def}}{=} \preceq_\alpha \wedge \succeq_\alpha \tag{3}$$

  where $\succeq_\alpha$ is the fuzzy relation on $\mathcal{S}$ defined as: $\succeq_\alpha \stackrel{\text{def}}{=} \preceq_\alpha^{-1}$;

- a fuzzy partial order $\preceq$ on $\mathcal{S}$: a fuzzy set $\Pi^{\sim} \stackrel{\text{def}}{=} \{\, \Pi_\alpha^{\sim} \mid \alpha \in \textbf{DEGREES}^{\sim} \,\}$ of partitions of $\mathcal{S}$ with partial orders $\preceq_\alpha$ on each partition $\Pi_\alpha^{\sim}$ of $\mathcal{S}$ generated by $\sim$ (*i.e.*, Zadeh's "partition tree"), and defined at approximation degree $\alpha \in [0.0, 01]$ as:

$$[s]_\alpha^{\sim} \preceq_\alpha [t]_\alpha^{\sim} \text{ iff } \exists\, s' \in \mathcal{S},\ \exists\, t' \in \mathcal{S} \text{ s.t. } s \sim_\alpha s' \text{ and } s' \preceq_\alpha t' \text{ and } t' \sim_\alpha t. \tag{4}$$

Copyright © 2020 by the Authors All Rights Reserved

This last condition may look harder to read than what it actually means, and can be understood more easily as the following color-highlighted diagram:

$$[s]^{\sim}_{\alpha} \preceq_{\alpha} [t]^{\sim}_{\alpha} \quad \stackrel{\text{def}}{\textit{iff}} \quad \exists\, s' \in \mathcal{S}, \exists\, t' \in \mathcal{S} \;\; \text{s.t. :} \begin{cases} t' & \sim_{\alpha} & t \\ & \preceq_{\alpha} & \\ s & \sim_{\alpha} & s' \end{cases}$$

where a subsort is below its supersort and similar sorts are on the same level.

The two following lemmas are also a direct consequence of the above properties.

LEMMA 1 (SORT-SUBSUMPTION FUZZY SYMMETRY) *For any sorts s and t in $\mathcal{S}$, and any approximation degrees α and β in $[0.0, 1.0]$, if $s \preceq_{\alpha} t$ and $t \preceq_{\beta} s$, then $s \sim_{\alpha \wedge \beta} t$.*

LEMMA 2 (SORT-SUBSUMPTION FUZZY TRANSITIVITY) *For any sorts s, t, and u in $\mathcal{S}$, and any approximation degrees α and β in $[0.0, 1.0]$, if $s \preceq_{\alpha} t$ and $t \preceq_{\beta} u$, then $s \sim_{\alpha \wedge \beta} u$.*

An important consequence is that, when considering a similarity on a sort signature $\mathcal{S}$ that is partially ordered by a defined sort subsumption, this must necessarily obey some consistency conditions for the similarity and the sort ordering. In particular, as the approximation degree α decreases from 1.0 to 0.0, the set of sorts $[s]^{\sim}_{\alpha}$ (denoting the similarity class of a sort s at approximation degree α) may only increase in size. Indeed, as α decreases, similarity classes of sorts may only coalesce, forming coarser and coarser similarity partitions.[3] It is not difficult to establish that the following properties are always true for any order-consistent similarity $\sim$ on a set of partially ordered sorts $\mathcal{S}, \preceq$.

PROPOSITION 1 (FUZZY SORT SUBSUMPTION CONTRAVARIANCE) *For all sort s in $\mathcal{S}$, and all approximation degrees α and β in $[0.0, 0.1]$:*

$$\alpha \leq \beta \;\; \textit{iff} \;\; [s]^{\sim}_{\beta} \subseteq [s]^{\sim}_{\alpha} \,. \tag{5}$$

In words, the contravariance in Condition (5) of Proposition 1 states that the smaller the approximation degree, the larger the similarity class.

As a consequence, Corollary 1 states that all sorts are indistinguishable at approximation degree 0.0, since then all sort classes coalesce into a single similarity class equal to the whole set of sorts — which is what Condition (6) expresses.

COROLLARY 1 (FULLY SIMILAR SORTS) *For any sort $s \in \mathcal{S}$:*

$$[s]^{\sim}_{0.0} \;=\; \mathcal{S} \,. \tag{6}$$

Order-consistency between the fuzzy partial order on sorts $\preceq$ and inclusion $\subseteq$ on sort-similarity classes is established in the next proposition as a monotonic order isomorphism.

PROPOSITION 2 (SORT SUBSUMPTION MONOTONICITY) *For all sorts s and t in $\mathcal{S}$ and all approximation degree α in $[0.0, 0.1]$:*

$$s \preceq_{\alpha} t \;\; \textit{iff} \;\; [s]^{\sim}_{\alpha} \subseteq [t]^{\sim}_{\alpha} \,. \tag{7}$$

[3] This is the set of partitions $\Pi^{\sim}_{\alpha}, \alpha \in$ **DEGREES**$^{\sim}$ that Zadeh calls the similarity's "*partition tree*" [Zad71]. See also [DP80], Chapter II, Section 3 on Fuzzy Relations (Page 77).

Copyright © 2020 by the Authors All Rights Reserved

On the other hand, as indicated by Condition (7) in Proposition 2, the approximation degree is covariant with the subsort ordering. When in addition the partially ordered sort signature is also a lattice $\mathcal{S}, \preceq, \curlywedge, \curlyvee$, this is equivalent to the validity of the following two propositions.

PROPOSITION 3 (FUZZY SORT-CONGRUENCE APPROXIMATION) *For all sort s in $\mathcal{S}$, and all approximation degrees α and β in $[0.0, 0.1]$:*

$$\begin{aligned} [s]^{\sim}_{\alpha\vee\beta} &= [s]^{\sim}_{\alpha} \cap [s]^{\sim}_{\beta}\,, \\ [s]^{\sim}_{\alpha\wedge\beta} &= [s]^{\sim}_{\alpha} \cup [s]^{\sim}_{\beta}\,. \end{aligned} \tag{8}$$

Example 1 Fuzzy sort-congruence approximation — Let us take $s \stackrel{\text{def}}{=} \texttt{person}$, $\alpha = .6$, and $\beta = .4$, with $\vee \stackrel{\text{def}}{=} \max$ and $\wedge \stackrel{\text{def}}{=} \min$. Then, fuzzy sort subsumption contravariance (Proposition 1) is clearly satisfied since:

$$\begin{aligned} [\texttt{person}]^{\sim}_{.6} \cap [\texttt{person}]^{\sim}_{.4} &= [\texttt{person}]^{\sim}_{.6}\,, \\ [\texttt{person}]^{\sim}_{.6} \cup [\texttt{person}]^{\sim}_{.4} &= [\texttt{person}]^{\sim}_{.4}\,. \end{aligned}$$

PROPOSITION 4 (FUZZY SORT-CONGRUENCE LATTICE) *For all sorts s and t in $\mathcal{S}$ and all approximation degree α in $[0.0, 0.1]$:*

$$\begin{aligned} [s \curlyvee t]^{\sim}_{\alpha} &= [s]^{\sim}_{\alpha} \cup [t]^{\sim}_{\alpha}\,, \\ [s \curlywedge t]^{\sim}_{\alpha} &= [s]^{\sim}_{\alpha} \cap [t]^{\sim}_{\alpha}\,. \end{aligned} \tag{9}$$

Example 2 Fuzzy sort-congruence lattice — Referring to the sorts in Figure 2, let us take $s \stackrel{\text{def}}{=} \texttt{employee}$, $t \stackrel{\text{def}}{=} \texttt{student}$ in Proposition 4. It is then obvious that, as stated by Proposition 4, for any $\alpha \in [0.0, 0.1]$:

$$\begin{aligned} [\texttt{staff}]^{\sim}_{\alpha} &= [\texttt{employee}]^{\sim}_{\alpha} \cup [\texttt{student}]^{\sim}_{\alpha}\,, \\ [\texttt{working-student}]^{\sim}_{\alpha} &= [\texttt{employee}]^{\sim}_{\alpha} \cap [\texttt{student}]^{\sim}_{\alpha}\,. \end{aligned}$$

Figure 3 and Figure 4 illustrate graphically how fuzzy subset approximation varies for three possible sorts **a**, **b**, and **c** as the approximation degree decreases from fully crisp (*i.e.*, 1.0 — when *no distinct sorts in $\mathcal{S}$ are similar*), to fully fuzzy (*i.e.*, 0.0 — when *all non-empty sorts in $\mathcal{S}$ are similar*). Figure 3, shows a typical possible example of fuzzy subsorting. Each row varies from fully crisp (degree 1.0) at the top, to fully fuzzy (degree 0.0) at the bottom as indicated in the leftmost column, while each of the other columns on the right indicates the fuzzy denotation of (from left to right) the sort $\top$ (which denotes the whole set of sorts $\mathcal{S}$) and sorts **a**, **b**, and **c**. As the approximation degree varies from crisp 1.0 to lower values, first to β, then to fuzzier $\alpha \leq \beta$, and ultimately to 0.0, each column shows a conceivable consistent variation of the denotation any sort $s \in \mathcal{S}$ (*e.g.*, in our example, $s = \texttt{a}$, $s = \texttt{b}$, and $s = \texttt{c}$):

$$0.0 \leq \alpha \leq \beta \leq 1.1 \implies \begin{cases} \top_{1.0} \subseteq \top_{\beta} \subseteq \top_{\alpha} \subseteq \top_{0.0}\,, \\ s_{1.0} \subseteq s_{\beta} \subseteq s_{\alpha} \subseteq s_{0.0}\,, \\ \bot_{1.0} \subseteq \bot_{\beta} \subseteq \bot_{\alpha} \subseteq \bot_{0.0}\,. \end{cases}$$

Copyright © 2020 by the Authors All Rights Reserved

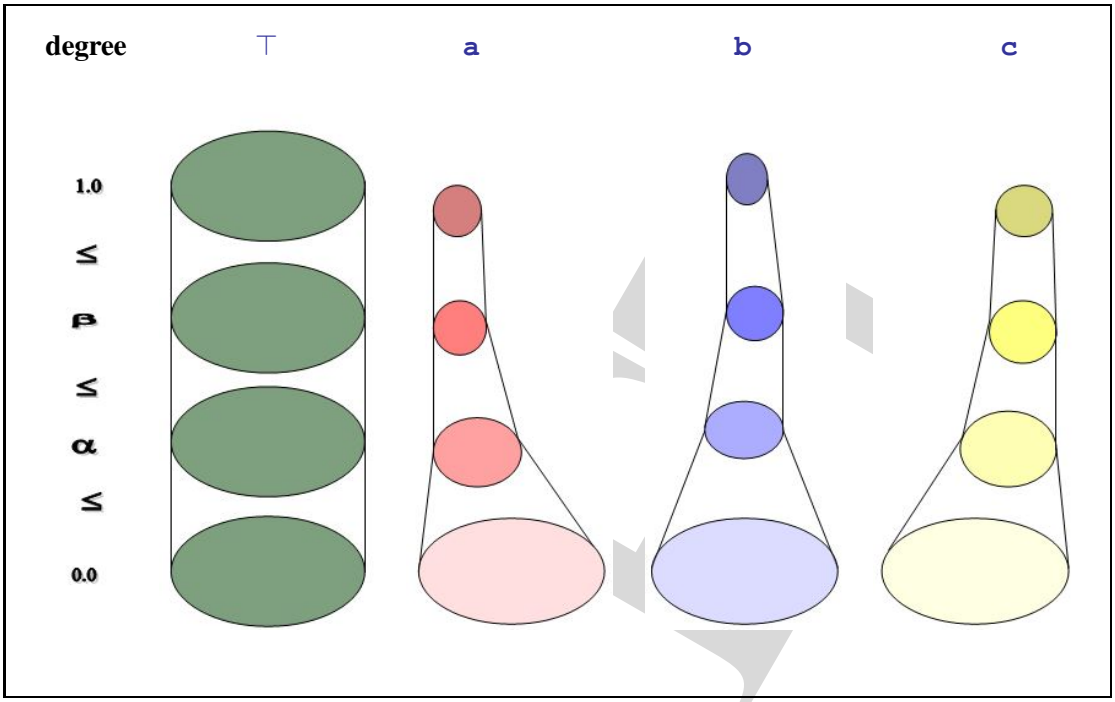

Figure 3: Fuzzy subset approximations

Since it is always true that $[\![\top_\alpha]\!] = \mathcal{S}$ at any approximation level $\alpha \in [0.0, 0.1]$, the column of fuzzy top denotations on the left in Figure 3 is always the full set $\mathcal{S}$. So are all the sort denotations in the bottom row when all sorts are similar, since $[\![s_{0.0}]\!] = \mathcal{S}$ for any sort $s \in \mathcal{S}$. This means that whenever $\alpha \leq \beta$ for some $\alpha \in [0.0, 1.0]$ and $\beta \in [0.0, 1.0]$, then necessarily $s_\beta \preceq s_\alpha$ (*i.e.*, equivalently, $[\![s_\beta]\!] \subseteq [\![s_\alpha]\!]$ by definition), for any sort s in $\mathcal{S}$.

This combined consistent approximation effect of conjugating both fuzzy approximation and subsort approximation on sort denotations, together with coalescing sort-similarity classes, is illustrated in Figure 4, showing the sort-similarity lattice orderings of these classes for each fuzzy level going from crisp at the top to fully fuzzy at the bottom. A good intuitive way to construe this effect is as that of a zooming lens or a magnifying-glass: approximation level 0.0 is farthest and most myopic (*i.e.*, "*seeing the forest for the trees*"), while level 1.0 is closest and sharpest (*i.e.*, "*seeing the trees for the forest*").

2.2 Fuzzy $\mathcal{OSF}$-term unification

We are, finally, ready to provide the constraint normalization rules for fuzzy $\mathcal{OSF}$-term unification and generalization. We first define formally what it means for two ψ-terms to be similar, modulo a subsort ordering and an order-consistent similarity (fuzzy equivalence) relation on the sorts.

Copyright © 2020 by the Authors All Rights Reserved

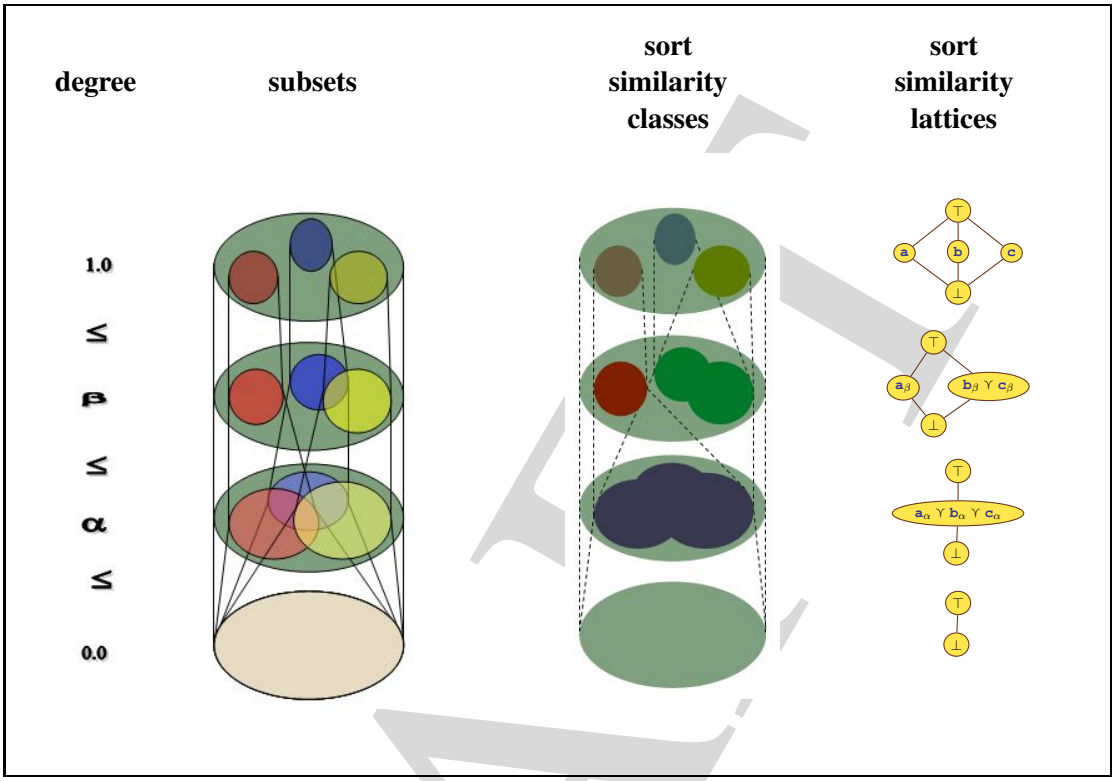

Figure 4: Fuzzy subset approximation lattice

§ 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 1 *For $\alpha \in [0.0, 0.1]$, 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 $\alpha \stackrel{def}{=} \beta \wedge \bigwedge_{i=0}^{n} \beta_i$ where:*

$$s \sim_\beta s' \tag{10}$$

for some $\beta \in (0.0, 1.0]$, and:

$$\psi_i \sim_{\beta_i} \psi'_{i'}[X/X'] \tag{11}$$

where $\beta_i \in (0.0, 1.0]$, for any $i \in \{1, \ldots, n\}$ such that $f_i \in \{f'_1, \ldots, f'_{n'}\}$.

The fuzzy $\mathcal{OSF}$ unification rules are shown in Figure 5.

THEOREM 1 (SIMILARITY OF ψ-TERMS) *The fuzzy binary relation $\sim_\alpha$ defined by Definition 1 is a similarity on the set of ψ-terms Ψ.*

Copyright © 2020 by the Authors All Rights Reserved

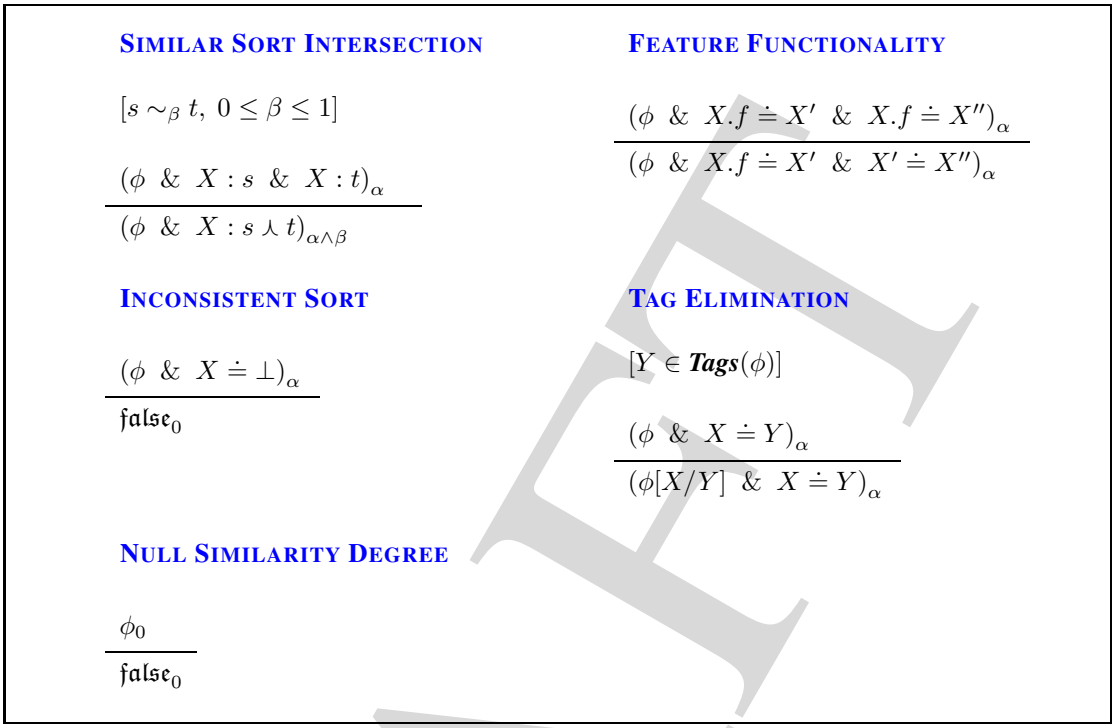

Figure 5: Constraint normalization rules for fuzzy $\mathcal{OSF}$ unification

Because of Theorem 1, we shall say that ψ and ψ' are α-*similar* iff $\psi \sim_\alpha \psi'$.

THEOREM 2 (CORRECTNESS OF FUZZY $\mathcal{OSF}$ UNIFICATION) *Given a fuzzy $\mathcal{OSF}$ constraint ϕ_α with $\alpha \in [0.0, 0.1]$, the process of non-deterministically applying to it any applicable rule shown in Figure 5 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 $\phi \sim_{\alpha'} \phi'$.*

2.3 Fuzzy $\mathcal{OSF}$-term generalization

Axiom **FUZZY EQUAL TAGS** in Figure 6 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, Rule **FUZZY UNEQUAL TAGS** in Figure 6 uses a "fuzzy unapply" operation '$\uparrow_\alpha$' that takes a pair of ψ-terms with unequal root tags and an approximation degree α and returns a pair of (possibly identical) ψ-terms and a possibly lesser approximation degree. It is defined as follows:

$$\begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \uparrow_\alpha \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix} \stackrel{\text{def}}{=} \begin{cases} \begin{pmatrix}X:\ldots\\ X:\ldots\end{pmatrix}_{\alpha\wedge\alpha_1\wedge\alpha_2} & \text{if } \exists\, X \text{ s.t. } \mathbf{ROOT}(\psi_i) = \gamma_i(X) \text{ for } i = 1,2; \\ \begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix}_\alpha & \text{otherwise.} \end{cases} \tag{12}$$

Copyright © 2020 by the Authors All Rights Reserved

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

$$\left[\begin{array}{l} X \neq Y;\ s \sim_{\beta} t;\ \alpha_0 \stackrel{\text{def}}{=} \alpha \wedge \beta;\\ m, n, p \geq 0 \text{ and } \{h_1, \ldots, h_p\} \stackrel{\text{def}}{=} \{f_1, \ldots, f_m\} \cap \{g_1, \ldots, g_n\}\\ \text{s.t. } h_k \stackrel{\text{def}}{=} f_k = g_k \text{ for all } k = 1, \ldots, p;\\ \gamma_1^0 \stackrel{\text{def}}{=} \gamma_1 \circ \{X/Z\} \text{ and } \gamma_2^0 \stackrel{\text{def}}{=} \gamma_2 \circ \{Y/Z\}, \text{ where } Z \text{ is a new tag name} \end{array}\right]$$

$$\frac{\begin{pmatrix}\gamma_1^0\\ \gamma_2^0\end{pmatrix}_{\alpha_0} \vdash \begin{pmatrix}\psi_1\\ \xi_1\end{pmatrix} \uparrow_{\alpha_0} \begin{pmatrix}\gamma_1^0\\ \gamma_2^0\end{pmatrix} \chi_1 \begin{pmatrix}\gamma_1^1\\ \gamma_2^1\end{pmatrix}_{\alpha_1} \cdots \begin{pmatrix}\gamma_1^{p-1}\\ \gamma_2^{p-1}\end{pmatrix}_{\alpha_{p-1}} \vdash \begin{pmatrix}\psi_p\\ \xi_p\end{pmatrix} \uparrow_{\alpha_{p-1}} \begin{pmatrix}\gamma_1^{p-1}\\ \gamma_2^{p-1}\end{pmatrix} \chi_p \begin{pmatrix}\gamma_1^p\\ \gamma_2^p\end{pmatrix}_{\alpha_p}}{\begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}_{\alpha} \vdash \begin{pmatrix}X : s\,(f_i \rightarrow \psi_i)_{i=1}^m\\ Y : t\,(g_j \rightarrow \xi_j)_{j=1}^n\end{pmatrix} Z : s \vee t\,(h_k \rightarrow \chi_k)_{k=1}^p \begin{pmatrix}\gamma_1^p\\ \gamma_2^p\end{pmatrix}_{\alpha_p}}$$

Figure 6: Fuzzy $\mathcal{OSF}$ generalization axiom and rule

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 tag (X) to the roots of the pair of ψ-terms to be generalized (ψ_1, ψ_2) each at, respectively, approximation α_1 and α_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.

This rule basically 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 corresponding sets of features of ψ_1 and ψ_2 (*i.e.*, the corresponding features they have in common), and Z is a new tag name.

As was the case for $\mathcal{FOT}$s, note that fuzzy $\mathcal{OSF}$ unapplication defined by Equation (12) returns a pair of terms and a (possibly lesser) approximation degree, unlike crisp unapplication defined in [AKP20] that returns only a pair of terms. Because of this, when we write a fuzzy $\mathcal{OSF}$ generalization judgment such as:

$$\begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}_{\alpha} \vdash \begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \uparrow_{\alpha} \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix} \psi \begin{pmatrix}\gamma_1'\\ \gamma_2'\end{pmatrix}_{\beta} \tag{13}$$

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 $\begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \uparrow_{\alpha} \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}$.

Copyright © 2020 by the Authors All Rights Reserved

Formally, the notation of the fuzzy $\mathcal{OSF}$ generalization judgment (13) is equivalent to:

$$\begin{pmatrix}\psi'_1\\ \psi'_2\end{pmatrix}_{\beta'} \stackrel{\text{def}}{=} \begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \uparrow_\alpha \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix} \quad \text{and} \quad \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}_{\beta'} \vdash \begin{pmatrix}\psi'_1\\ \psi'_2\end{pmatrix} \psi \begin{pmatrix}\gamma'_1\\ \gamma'_2\end{pmatrix}_{\beta} \tag{14}$$

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 (12), lowering the *prior* approximation degree of the judgment. This is therefore applicable to pairs of subterms having some features in common. It consists in 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.

THEOREM 3 (CORRECTNESS OF FUZZY $\mathcal{OSF}$ GENERALIZATION) *The process of using any applicable constrained Horn clause shown in Figure 6 as long as one applies starting with the fuzzy $\mathcal{OSF}$ judgment to establish:*

$$\begin{pmatrix}\emptyset\\ \emptyset\end{pmatrix}_{1.0} \vdash \begin{pmatrix}\psi_1\\ \psi_2\end{pmatrix} \psi \begin{pmatrix}\gamma_1\\ \gamma_2\end{pmatrix}_{\alpha}$$

to prove this judgment's validity always terminates with $\psi = \psi_1 \wedge_\alpha \psi_2$, together with $\gamma_1 : \textbf{\textit{Tags}}(\psi) \to \textbf{\textit{Tags}}(\psi_1)$ and $\gamma_2 : \textbf{\textit{Tags}}(\psi) \to \textbf{\textit{Tags}}(\psi_2)$ such that $\psi_1 = \gamma_1(\psi)$ and $\psi_2 = \gamma_2(\psi)$.

3 Implementation

An $\mathcal{OSF}$ constraint ϕ in solved form is always satisfiable in a canonical interpretation structure; *viz.*, the $\mathcal{OSF}$ graph algebra Ψ [AKP93]. 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 [AKdC93]. 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 [AKBLN89]. This results in tremendous speed performance when compared to encoding a class taxonomy's partial order using First-Order Logic monadic implication, even when resorting to proof "*memoing*" as done, for example, in [KLW95], since this requires dynamically memorizing arbitrary proofs, thereby facing a hefty overhead price both in space and time. Indeed, resorting to bitvector-encoded ordered sorts rather than monadic implications is the key providing immediate deductive response thanks to static transitive closure on the "`is-a`" ordering and static consistent typing propagation of features to subtypes (see [AKA15] and [AAK16]).

However, isn't this valuable implementation trick lost with fuzzy, rather than binary, truth values? Some is, clearly, though not totally as we discuss next; and the gain fortunately outweighs the loss.

§ CLOSURE OF 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 ("$s_1 \preceq s_2$") when these are consistent (*e.g.*, when there is no cycle) or cycles are detected and

Copyright © 2020 by the Authors All Rights Reserved

reported [AKA15]. 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 [AKBLN89].

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 $[0.0,1.0]$-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-bit vector; *i.e.*, a vector of real values in the closed interval $[0.0,1.0]$ representing the fuzzy set $\{\,\alpha/s \mid \alpha \in [0.0,1.0] \text{ for all } s \in \mathcal{S}\,\}$. The bitwise Boolean operations on bit-vectors are now fuzzified into $\wedge$, $\vee$, and $\alpha \to (1.0-\alpha)$ on fuzzy set elements' fuzzy weights.[4] 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:

$$X \wedge Y \stackrel{\text{def}}{=} \{\,(\alpha \wedge \beta)/s \mid \alpha/s \in X \text{ and } \beta/s \in Y, \text{ for all } s \in \mathcal{S}\,\} \tag{15}$$

$$X \vee Y \stackrel{\text{def}}{=} \{\,(\alpha \vee \beta)/s \mid \alpha/s \in X \text{ and } \beta/s \in Y, \text{ for all } s \in \mathcal{S}\,\} \tag{16}$$

$$\overline{X} \stackrel{\text{def}}{=} \{\,(1.0-\alpha)/s \mid \alpha/s \in X, \text{ for all } s \in \mathcal{S}\,\} \tag{17}$$

for all X and Y fuzzy sets over a reference set of sorts $\mathcal{S}$. This is also the case with a similarity degree α and a fuzzy set X over $\mathcal{S}$:

$$\alpha \wedge X \stackrel{\text{def}}{=} \{\,(\alpha \wedge \beta)/s \mid \beta/s \in X, \text{ for all } s \in \mathcal{S}\,\} \tag{18}$$

and:

$$\alpha \vee X \stackrel{\text{def}}{=} \{\,(\alpha \vee \beta)/s \mid \beta/s \in X, \text{ for all } s \in \mathcal{S}\,\}. \tag{19}$$

Note that we seldom need to represent explicitly a 0.0-similarity degree fuzzy element (*i.e.*, of the form $0.0/s$) and neither do we need to store it explicitly in a fuzzy set representation. In particular, in all the foregoing definitions given as Equation (15)–Equation (19), 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 $0.0/s \in X$.

It will always be assumed that a top sort ("$\top$") and a bottom sort ("$\bot$") are implicitly declared such that:

$$s \preceq_{1.0} \top \tag{20}$$

and,

$$s \neq \top \Rightarrow \top \preceq_{0.0} s \tag{21}$$

[4] We use the notation "$x \to e$" to denote a nameless function associating the expression e to the argument x; *i.e.*, what the adepts of Functional Programming write as $\lambda x.e$ (here, $\lambda\alpha.(1.0-\alpha)$) and call a functional abstraction or λ-expression.

Copyright © 2020 by the Authors All Rights Reserved

as well as:

$$\bot \preceq_{1.0} s \tag{22}$$

and,

$$s \neq \bot \Rightarrow s \preceq_{0.0} \bot \tag{23}$$

for all sorts $s \in \mathcal{S}$, in order to express respectively that there is no fuzziness in the sort ordering of $\top$ as the greatest (and all-encompassing) sort, and $\bot$ as the least (and all-excluding sort).

In [AKBLN89] and [AKA15], the encoding of crisp-ordered sorts as bit vectors is given in pseudo-code as a reflexive-transitive 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 reviewed in [AKA15] 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.

```
1  procedure CloseFuzzyTaxonomy ()
2  |  Set⟨Sort⟩ layer ← ⊥.parents;
3  |  while layer ≠ ∅ do
4  |  |  foreach Sort s ∈ layer do
5  |  |  |  s.fuzzyset ← {1.0/s} ∨ ⋁_{α/u ∈ 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

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 by (or is a subsort of) sort s_2 with similarity degree $\alpha \in (0.0, 1.0]$; namely, $(s_1, \alpha) \in s_2$.`children`. The class `Sort` has another field called "`parents`" of type `Set⟨Sort⟩` containing, for any

Copyright © 2020 by the Authors All Rights Reserved

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 `children` sets while closing a fuzzy taxonomy.

In addition, the class `Sort` has:

- 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`$\langle$`Sort,double`$\rangle$ 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 $\alpha_{ij} \in (0.0, 1.0]$ 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**).

§ Optimizing Closure and Lattice Operations

There is an immediate issue that we should keep 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 [AKBLN89], and used in [AKA15] and in [AK14]. After closing it, a sort's bit-vector represents the set of its lower bounds. Indeed, this enabled optimizing set-lattice operations on ordered set-denoting sorts (very fast operations 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 [AKBLN89]. With a fuzzy partial-order, however, a sort is no longer identified with a bit vector but with a $(0.0, 1.0]$-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.*, $N \stackrel{\text{def}}{=} |\bot$`.parents`$|$. Each value `base[i]`, `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

Copyright © 2020 by the Authors All Rights Reserved

an array of $N \leq$ `taxonomy.size()` of $[0.0, 1.0]$-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:[5]

$$\texttt{s.crispvalue}[i] \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } \texttt{s.fuzzybase}[i] > 0; \\ 0 & \text{otherwise.} \end{cases}$$

This information is therefore straightforward for 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 [AKBLN89] and [AKA15].

4 Conclusion

We demonstrated in detail how the fact that an Order-Sorted Feature graph structure is but a special case of a Prolog's first-order terms allows a fuzzification of its lattice operations in a similar way as demonstrated for $\mathcal{FOT}$s in [AKP20]. The same crisp and fuzzy lattice-theoretic operations for these more general rooted graphs can be shown to extend those on the more restricted $\mathcal{FOT}$s for constraint-based knowldege representation and automated reasoning — both in expressivity and efficiency. We have summarized the essential of what this entails formally and operationally as systems solving fuzzy constraints.

References

[AAK16] Samir Amir and Hassan Aït-Kaci. An efficient and large-scale reasoning method for the semantic web. *Journal of Intelligent Information Systems*, 47(3):1–22, December 2016. [Available online].

[AK14] 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].

[AKA15] 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].

[AKBLN89] 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].

[AKdC93] 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].

[5] This is representable, for example, as a Java class such as `hlt.osf.util.BitCode`, which extends the standard Java class `java.util.BitSet`.

Copyright © 2020 by the Authors All Rights Reserved

[AKP93] Hassan Aït-Kaci and Andreas Podelski. Towards a meaning of $\mathcal{LIFE}$. *Journal of Logic Programming*, 16(3-4):195–234, 1993. [Available online].

[AKP20] Hassan Aït-Kaci and Gabriella Pasi. Fuzzy lattice operations on first-order terms over signatures with similar constructors: A constraint-based approach. *Fuzzy Sets & Systems*, 2020. [Available online].

[DP80] 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].

[KLW95] 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].

[Zad71] Lotfi A. Zadeh. Similarity relations and fuzzy orderings. *Information Sciences*, 3:177–200, 1971. [Available online].

DRAFT

Copyright © 2020 by the Authors *All Rights Reserved*