Similarity-based OSF Logic for Flexible SPARQL Query Answering

Gian Carlo Milanese[1,*], Gabriella Pasi[1]

[1]University of Milano-Bicocca, DISCo, Milan, Italy

Abstract

We present an updated version of `fosf`, a reasoner for fuzzy order-sorted feature (OSF) logic, with support for flexible SPARQL query answering over RDF knowledge graphs. Fuzzy OSF logic is a knowledge representation and reasoning language where feature symbols denote functions and sort symbols denote fuzzy sets. Sorts are organized in a fuzzy subsumption relation that extends to OSF terms, record-like structures representing classes of entities. The unification algorithm for these structures provides a calculus of fuzzy type subsumption. The latest release of `fosf` introduces support for similarity-based reasoning by extending the OSF language with a similarity relation among sorts, alongside a translation engine for compiling OSF terms into standard SPARQL. Queries expressed as OSF terms undergo fuzzy ontology-aware rewriting prior to translation. This process incorporates an OSF theory of sort definitions, feature domain and range declarations, and fuzzy subsumption and similarity relations through fuzzy OSF constraint normalization. This rewriting enables flexible query answering by returning approximate solutions, allowing queries that would otherwise be inconsistent with the underlying ontology to yield relevant results.

Keywords

RDF knowledge graphs, SPARQL, Similarity-based reasoning, Order-sorted feature logic, Query rewriting

1. Introduction

We present an updated release of `fosf`,[1] a reasoner for fuzzy order-sorted feature (OSF) logic, that introduces similarity-based OSF reasoning alongside an ontology-aware translation engine for querying RDF knowledge graphs (KGs) via SPARQL. Queries expressed as OSF terms undergo fuzzy constraint normalization prior to translation into standard SPARQL. This process integrates ontological information from a sort taxonomy, a sort similarity relation, sort definitions, and feature domain and range declarations. The resulting query rewriting mechanism enables the retrieval of approximate solutions for queries that would otherwise fail under strict ontological constraints.

Background OSF logic is a knowledge representation and reasoning language introduced by Aït-Kaci [1, 2]. It is based on *sorts*, which represent classes of objects, and *features*, which represent functional attributes. Sorts are organized in a subsumption relation that extends to OSF terms, record-like structures representing classes of entities and their constraints. Reasoning with OSF logic is based on the *normalization* (or *unification*) of OSF terms, a procedure that aims to combine the constraints expressed by one or more terms while taking the sort hierarchy into account. OSF logic is at the basis of CEDAR [3], a Semantic Web reasoner that normalizes and optimizes queries expressed as OSF terms before translating them into SPARQL, possibly leading to more efficient retrieval.

Fuzzy OSF logic [4] extends OSF logic by interpreting sorts and OSF terms as *fuzzy sets*, and by generalizing the sort subsumption to a *fuzzy subsumption relation among sorts*, which is then extended to OSF terms. The core reasoning algorithms of fuzzy OSF logic are implemented in the `fosf` Python library [5], which supports fuzzy taxonomic reasoning, subsumption degree computation, and fuzzy normalization and unification of OSF terms.

ISWC 2026 Companion Volume, October 25–29, 2026, Bari, Italy

*Corresponding author.

giancarlo.milanese@unimib.it (G. C. Milanese); gabriella.pasi@unimib.it (G. Pasi)

0000-0002-1757-0152 (G. C. Milanese); 0000-0002-6080-8170 (G. Pasi)

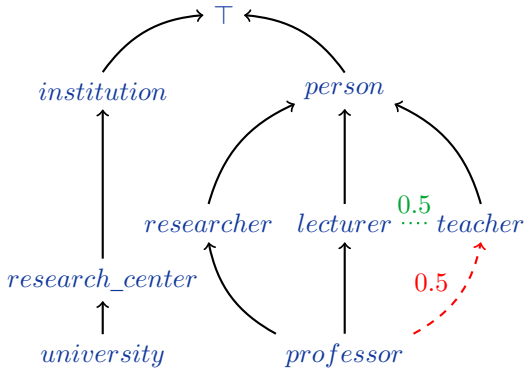


© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

[1]Source code and examples: https://github.com/fuzzyosf/fosf. Interactive demo: https://fosf-demo.pages.dev/.

```
from fosf.parsers import parse_taxonomy
sim_tax_str = """
# Subsumption declarations
institution, person < top .
research_center < institution .
university < research_center .
researcher, lecturer, teacher < person .
professor < researcher, lecturer .
# Similarity declarations
teacher ~ lecturer = 0.5 ."""
sim_tax = parse_taxonomy(sim_tax_str)
```

(a) Parsing subsumption and similarity relations.



(b) Graphical representation.

Figure 1: Example of how a sort subsumption $\preceq$ (black edges) and a sort similarity $\sim$ (dotted green edges) are parsed in `fosf`, and their graphical representation (where $\bot$ is omitted). After parsing, the two relations are combined into a single fuzzy subsumption $\preceq$ (black edges and dashed red edges).

Contributions. The new release of `fosf` introduces the following features: (i) support for a similarity relation between sorts, which is integrated into the main reasoning algorithms of fuzzy OSF logic, enabling *similarity-based reasoning with OSF logic* [6]; (ii) support for feature domain and range declarations during OSF term normalization; (iii) querying RDF KGs through the translation of normalized OSF queries into standard SPARQL. The normalization exploits fuzzy sort subsumption and sort similarity relations to relax incompatible constraints and retrieve approximate solutions.

Related work. Implementations of (crisp) OSF logic include the constraint logic programming language LIFE [7] and the CEDAR Semantic Web reasoner [3]. Closely related formalisms include description logics (DLs) and fuzzy DLs, which are supported by several reasoners (e.g., [8, 9, 10, 11, 12]). The similarities and differences between OSF logic and DLs have been studied extensively [13, 14, 15, 16]. Similarity-based OSF logic is inspired by fuzzy unification in fuzzy logic programming, where unification is guided by similarity or proximity relations (e.g., [17, 18, 19, 20, 21]); implementations include Bousi~Prolog [22] and FASILL [23]. In the Semantic Web community, various approaches handle imprecise or flexible querying over RDF data, including fuzzy SPARQL extensions (e.g., [24, 25]) and query relaxation techniques (e.g., [26]). In this context, `fosf` applies similarity-based normalization at the OSF logic level before compiling the resulting query terms into standard SPARQL.

2. Similarity-based OSF Taxonomic Reasoning

At the core of OSF logic is a sort taxonomy, a finite set of sort symbols $\mathcal{S}$ that is ordered in a lattice $(\mathcal{S}, \preceq, \wedge)$ with top element $\top$ and bottom element $\bot$. Similarity-based OSF logic [6] extends the sort taxonomy with a *similarity relation* on sort symbols, that is, a fuzzy binary relation $\sim\colon \mathcal{S} \times \mathcal{S} \to [0, 1]$ which satisfies reflexivity, symmetry and transitivity.[2] As described in [6], the information provided by the sort subsumption $\preceq$ and the similarity $\sim$ can be combined into a single fuzzy subsumption relation $\preceq : \mathcal{S} \times \mathcal{S} \to [0, 1]$, according to the following informal inference, inspired by the similarity-based approaches to logic programming (e.g., [18]):

if the sort s_0 is subsumed by the sort s_1	(i.e., $s_0 \preceq s_1$)
and s_1 is similar to the sort s_2 with degree α	(i.e., $\sim(s_1, s_2) = \alpha$)
then s_0 is subsumed by s_2 with degree α	(i.e., $\preceq(s_0, s_2) = \alpha$).

For example, if $professor$s are $lecturer$s, and $lecturer$s are similar to $teacher$s with degree 0.5, then we can conclude that $professor$s are also $teacher$s with subsumption degree 0.5. As shown in [6], the

[2] For fuzzy set theory definitions using the same notation as this paper, we refer to the appendix of [6].

```
from fosf.parsers import parse_term
from fosf.reasoning import normalize_term
# Normalizing an OSF term according to subsumption + similarity only
t_str = "X0:person(teach_at -> X1:institution, research_at -> X1:research_center)"
t = parse_term(t_str)
print(normalize_term(t, sim_tax))
# Output: X0 : person(teach_at -> X1 : research_center, research_at -> X1)
```

Figure 2: Normalizing an OSF term in `fosf`.

```
from fosf.parsers import parse_theory
theory_str = sim_tax_str + """
professor           := Yp:professor(research_at -> Y1:university) .
domain(teach_at)    := teacher .
domain(research_at) := researcher ."""
theory = parse_theory(theory_str, ensure_closed=True)
# Normalizing an OSF term also according to the theory
nt, degree = normalize_term(t, sim_tax, theory, return_degree=True)
print(degree, nt)
# Output: 0.5 X0 : professor(teach_at -> X1 : university, research_at -> X1)
```

Figure 3: Normalizing an OSF term modulo an OSF theory in `fosf`.

resulting fuzzy subsumption relation is a fuzzy lattice $(\mathcal{S}, \preceq, \curlywedge)$, a fuzzy binary relation that satisfies reflexivity, transitivity and strong antisymmetry, and such that a GLB $s_0 \curlywedge s_1$ exists for every $s_0, s_1 \in \mathcal{S}$.[3]

As shown in Fig. 1, `fosf` supports parsing a sort subsumption relation $\preceq$ and a sort similarity relation $\sim$ and combining them in a single fuzzy subsumption relation $\precsim$. Cycle detection is performed to ensure that the resulting fuzzy subsumption $\precsim$ is antisymmetric, possibly merging each sort cycle into a single disjunctive set of sorts, according to [6]. When the resulting fuzzy relation $\precsim$ is not a (fuzzy) lattice, but simply a partial order, `fosf` works on a completion of $\precsim$ that does not need to be explicitly computed.[4] By employing the graph encoding technique of [28] and optimized weighted graph traversal of [29], it is possible to efficiently compute the GLB $s_0 \curlywedge s_1$ and the subsumption degree $\precsim(s_0, s_1)$ of two sorts s_0 and s_1 [5]. For instance, $teacher \curlywedge researcher = professor$ and $\precsim(professor, teacher) = 0.5$.

3. Similarity-based OSF Term Normalization

Sort symbols may be used with variables (also called tags) $X_0, X_1 \ldots \in \mathcal{V}$ and features symbols $f_0, \ldots, f_n \in \mathcal{F}$ to build *OSF terms*. These are record-like representations of (fuzzy) sets of objects, such as $t = X_0 : person(teach_at \to X_1 : institution, research_at \to X_1 : research_center)$, which denotes the class of people that teach at some institution X_1 and carry out research at the same X_1, which is also a research center. In general, OSF terms may include redundant, or even contradictory, information (e.g., the sort of X_1 is specified twice in t). OSF terms that are well-behaved to this regard are called *normal OSF terms* or ψ*-terms* [2], and they are also denoted ψ, ψ_i, and so on. The `fosf` reasoner supports normalizing OSF terms according to a sort subsumption relation $\preceq$ and similarity $\sim$: as shown in Fig. 2, the term t can be normalized to $person(teach_at \to X_1 : research_center, research_at \to X_1)$, where the sort of X_1 is simplified to $research_center = institution \curlywedge research_center$.

The `fosf` reasoner also supports the normalization of OSF terms modulo an *OSF theory* of *sort definitions* and *feature domain and range declarations*. Sort definitions associate each sort s with a normal term $\Theta(s) = \psi$ such that $\Theta(\bot) = \bot$, $\Theta(\top) = \top$, and the root sort of $\Theta(s)$ is s for all $s \in \mathcal{S}$ [30]. Intuitively, the *sort definition* $\Theta(s)$ specifies structural constraints that each OSF term of type s

[3] A remark about notation: $\preceq$ denotes the crisp subsumption lattice on $\mathcal{S}$, and $\curlywedge$ is the GLB operation on $(\mathcal{S}, \preceq)$; $\precsim$ is the fuzzy lattice resulting from combining $\preceq$ with $\sim$, and $\curlywedge$ is the GLB operation on $(\mathcal{S}, \precsim)$.

[4] For more details, see [27, 6]. In this paper we assume that $\precsim$ is always a fuzzy lattice for simplicity.

```
SELECT ?X0 WHERE {
  ?X0 rdf:type :person .
  ?X0 :teach_at ?X1 .
  ?X0 :research_at ?X1 .
  ?X1 rdf:type :institution .
  ?X1 rdf:type :research_center . }
```

(a) SPARQL query before fuzzy OSF normalization.

```
SELECT ?X0 WHERE {
  ?X0 rdf:type :professor .
  ?X0 :teach_at ?X1 .
  ?X0 :research_at ?X1 .
  ?X1 rdf:type :university . }
```

(b) SPARQL query after fuzzy OSF normalization.

Figure 4: SPARQL queries corresponding to the OSF term t, before and after normalization.

should satisfy. Feature *domain* and *range* declarations have shape $domain(f) = s_0$ and $range(f) = s_1$, and they ensure that, whenever $t' = X : \top(f \to Y : \top)$ is an OSF term, then X must be of sort s_0 and Y must be of sort s_1. Overall, the (fuzzy) sort subsumption and similarity relations, the sort definitions, and the feature domain and range declarations constitute the *ontological component* according to which an OSF term can be normalized. Fig. 3 shows an example of parsing an OSF theory in `fosf` and normalizing the OSF term t according to the theory. In the example, the domain declarations for the features $teach_at$ and $research_at$ enforce the tag X_0 to be of sort $teacher \curlywedge researcher = professor$, and the sort definition for $professor$ thus enforces X_1 to be a $university$. The constraints imposed by an OSF theory may be satisfied up to a degree $\alpha \in [0, 1]$. For instance, in the term `nt` of Fig. 3, X_0 is of sort $professor$ rather than $teacher$ (as required by the domain of $teach_at$) so that `nt` satisfies the OSF theory with degree $\preceq(professor, teacher) = 0.5$. Note that, without the fuzzy subsumption $\preceq(professor, teacher)$ induced by $professor \preceq lecturer$ and $\sim(lecturer, teacher)$, the GLB of $teacher$ and $researcher$ would instead be $\bot$, and the normalization of t would fail.

4. Similarity-based OSF for SPARQL Query Answering

The `fosf` reasoner supports translating a (normalized) OSF term into SPARQL for instance retrieval over an RDF KG. This translation is inspired by the query rewriting pipeline of CEDAR [3], in which an OSF term query is preprocessed to ensure consistency and to potentially reduce the retrieval search space before generating a SPARQL query. The `fosf` library extends this framework by integrating sort definitions and the sort similarity relation $\sim$ into the normalization, enabling similarity-based query rewriting. Without the sort similarity, queries violating ontological constraints would be declared inconsistent and return no results. Instead, `fosf` can rewrite OSF terms into queries that approximately satisfy the OSF theory to some degree, thus enabling the retrieval of approximate solutions.

Consider again the query term $t = ?X_0 : person(teach_at \to X_1 : institution, research_at \to X_1 : research_center)$, where X_0 is marked with a question mark to denote instance retrieval. Under *standard OSF normalization*, t is inconsistent because $teacher \curlywedge researcher = \bot$, as discussed in Section 3. Translating t *directly to SPARQL without normalizing* ignores the ontology entirely, yielding the query in Fig. 4a, which fails to enforce domain/range semantics and sort definitions. In contrast, by taking $\sim$ into account `fosf` normalizes t to $X_0 : professor(teach_at \to X_1 : university, research_at \to X_1)$, deriving an approximate solution that satisfies the OSF theory with a degree of 0.5 (Fig. 3). Fig. 4b shows the resulting SPARQL translation. Providing approximate solutions for otherwise inconsistent queries can be desirable in many scenarios. For example, users often do not possess full knowledge of extensive ontologies, making an inconsistent result potentially surprising. Moreover, the inconsistency of t arises from the closed-world assumption of OSF logic, where $teacher \curlywedge researcher = \bot$ forces disjointness of the two sorts. Relaxing this requirement can be valuable in contexts where this assumption does not hold, such as settings where domain knowledge can evolve.

From a practical standpoint, `fosf` handles OSF query parsing, normalization, and translation to SPARQL. Base IRIs and prefixes can be specified prior to an OSF theory or query using Turtle-like syntax (e.g., `@base <IRI> .` `@prefix : <IRI> .`). The syntax for OSF theories (including subsumption and similarity declarations) and OSF terms supports base sorts like $person$ (resolved via the base IRI), as

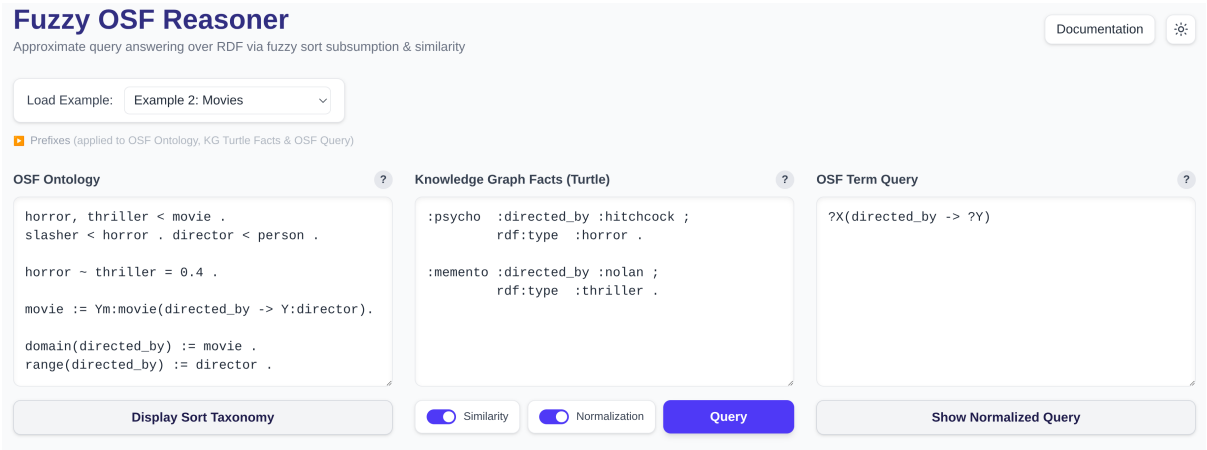


Figure 5: The web interface of `fosf`, displaying the OSF ontology composed of sort subsumption and similarity declarations, sort definitions, and feature domain/range constraints (left), the target RDF KG facts (middle), and the OSF query term (right). Action buttons enable users to inspect the taxonomy, display the translated query, and execute the generated SPARQL, with options to toggle OSF query normalization and the similarity relation.

well as absolute IRIs (e.g., `<http://example.org/person>`) and prefixed names (e.g., `ex:person`). The query generated by `fosf` can be executed over in-memory RDF graphs or remote SPARQL endpoints using the `rdflib` and `SPARQLWrapper` libraries,[5] or sent to any standard SPARQL query engine.

5. Demo Interface

We provide an interactive Web interface designed to demonstrate the query normalization and execution pipeline of `fosf`.[6] As shown in Fig. 5, users can specify custom OSF theories, target RDF KGs, and OSF queries directly in the browser, or select from a list of pre-defined examples. The interface can visually render the sort subsumption hierarchy, the sort similarity relation, and the resulting fuzzy subsumption relation. Users can inspect the generated SPARQL queries and execute them directly against the RDF KG to view retrieved instances alongside the approximation degree of the normalized query. Two global toggles, for OSF query normalization and sort similarity, allow users to compare OSF terms, generated SPARQL queries, and retrieval results with or without normalization and similarity-based reasoning.

6. Conclusion and Future Work

We have presented an updated release of `fosf`, extending fuzzy OSF logic with similarity-based reasoning and ontology-aware SPARQL query generation. By integrating a sort similarity relation into constraint normalization, `fosf` enables flexible query rewriting, deriving approximate solutions for queries that would otherwise be declared inconsistent under strict ontological constraints.

While the current release of `fosf` provides a functional prototype, several limitations present avenues for future development. Regarding performance, we are replacing the pure-Python reasoning core with a C++ engine to improve execution speed and memory efficiency. To address the closed-world assumption of OSF logic, where two sorts s_0 and s_1 are treated as disjoint whenever $s_0 \curlywedge s_1 = \bot$, we plan to allow declarations of non-disjoint sorts within sort taxonomies, without the need to explicitly name their GLB. Furthermore, we intend to improve the expressiveness of `fosf` by extending support for disjunctive sorts and terms in both theories and queries, as well as enabling sort instances in OSF query expressions. Finally, in addition to manually defined similarity relations, we plan to support automated similarity estimation using techniques such as vector embeddings.

[5] Available at https://github.com/RDFLib/rdflib and https://github.com/RDFLib/sparqlwrapper.
[6] Available at https://fosf-demo.pages.dev/.

Acknowledgments

The authors express their gratitude and a fond thought to Hassan Aït-Kaci, who with Gabriella Pasi set out to define a fuzzy version of OSF logic. Our development of fuzzy OSF logic originates from their work on the definition of similarity-based unification for OSF terms, extending the approach of [21]. We also wish to thank Marco Braga and Stefano Molari for their help in setting up the online demonstration system.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT, Gemini in order to: Grammar and spelling check. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] H. Aït-Kaci, A Lattice Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures, Ph.D. thesis, University of Pennsylvania, 1984. URL: https://www.proquest.com/openview/3816ec450cb910c1586b7627c6353da6.

[2] H. Aït-Kaci, A. Podelski, Towards a meaning of life, The Journal of Logic Programming 16 (1993) 195–234. doi:10.1016/0743-1066(93)90043-G.

[3] S. Amir, H. Aït-Kaci, An efficient and large-scale reasoning method for the semantic web, J. Intell. Inf. Syst. 48 (2017) 653–674. doi:10.1007/s10844-016-0435-2.

[4] G. C. Milanese, G. Pasi, Fuzzy order-sorted feature logic, Fuzzy Sets and Systems 477 (2024) 108800. doi:10.1016/j.fss.2023.108800.

[5] G. C. Milanese, G. Pasi, Implementing fuzzy osf logic unification and normalization (short paper), in: A. Biere, C. Lutz, S. Negri (Eds.), Automated Reasoning, Springer Nature Switzerland, Cham, 2026, pp. 368–377. URL: https://fosf.pages.dev/preprints/26_ijcar.pdf. doi:10.1007/978-3-032-32592-1_22.

[6] G. C. Milanese, G. Pasi, Similarity-Based Reasoning with Order-Sorted Feature Logic, IEEE Transactions on Fuzzy Systems 32 (2024) 2797–2810. doi:10.1109/TFUZZ.2024.3362897.

[7] H. Aït-Kaci, An introduction to LIFE-programming with logic, inheritance, functions, and equations, in: Proceedings of the 1993 International Symposium on Logic Programming, ILPS '93, MIT Press, Cambridge, MA, USA, 1993, pp. 52–68. URL: https://dl.acm.org/doi/10.5555/187812.187831.

[8] D. Tsarkov, I. Horrocks, FaCT++ Description Logic Reasoner: System Description, in: U. Furbach, N. Shankar (Eds.), Automated Reasoning, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 292–297. doi:10.1007/11814771_26.

[9] R. Shearer, B. Motik, I. Horrocks, Hermit: A Highly-Efficient OWL Reasoner, in: OWL: Experiences and Directions, 2008. URL: https://api.semanticscholar.org/CorpusID:7951194.

[10] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, Journal of Web Semantics 5 (2007) 51–53. doi:10.1016/j.websem.2007.03.004.

[11] G. Stoilos, N. Simou, G. Stamou, S. Kollias, Uncertainty and the Semantic Web, IEEE Intelligent Systems 21 (2006) 84–87. doi:10.1109/MIS.2006.105.

[12] F. Bobillo, U. Straccia, Optimising fuzzy description logic reasoners with general concept inclusion absorption, Fuzzy Sets and Systems 292 (2016) 98–129. doi:10.1016/j.fss.2014.10.029, Special Issue in Honor of Francesc Esteva on the Occasion of his 70th Birthday.

[13] B. Nebel, G. Smolka, Representation and reasoning with attributive descriptions, in: K. H. Bläsius, U. Hedtstück, C.-R. Rollinger (Eds.), Sorts and Types in Artificial Intelligence, Springer Berlin Heidelberg, Berlin, Heidelberg, 1990, pp. 111–139. doi:10.1007/3-540-52337-6_21.

[14] B. Nebel, G. Smolka, Attributive description formalisms... and the rest of the world, Springer Berlin Heidelberg, Berlin, Heidelberg, 1991, pp. 439–452. doi:10.1007/3-540-54594-8_74.

[15] H. Aït-Kaci, Data models as constraint systems: a key to the semantic web, Constraint Processing Letters 1 (2007). URL: https://constraint-programming.com/letters/Papers/v1/hak.pdf.
[16] H.-U. Krieger, U. Schäfer, DL Meet FL: A Bidirectional Mapping between Ontologies and Linguistic Knowledge, in: C.-R. Huang, D. Jurafsky (Eds.), Coling 2010: Posters, Coling 2010 Organizing Committee, Beijing, China, 2010, pp. 588–596. URL: https://aclanthology.org/C10-2067/.
[17] G. Gerla, M. I. Sessa, Similarity in Logic Programming, Springer US, Boston, MA, 1999, pp. 19–31. doi:10.1007/978-1-4615-5261-1_2.
[18] M. I. Sessa, Approximate reasoning by similarity-based SLD resolution, Theor. Comput. Sci. 275 (2002) 389–426. URL: http://www.sciencedirect.com/science/article/pii/S0304397501001888. doi:https://doi.org/10.1016/S0304-3975(01)00188-8.
[19] P. Julián-Iranzo, F. Sáenz-Pérez, Proximity-Based Unification: An Efficient Implementation Method, IEEE Transactions on Fuzzy Systems 29 (2021) 1238–1251. doi:10.1109/TFUZZ.2020.2973129.
[20] T. Kutsia, C. Pau, Solving Proximity Constraints, in: M. Gabbrielli (Ed.), Logic-Based Program Synthesis and Transformation, Springer International Publishing, Cham, 2020, pp. 107–122. doi:10.1007/978-3-030-45260-5_7.
[21] H. Aït-Kaci, G. Pasi, Fuzzy lattice operations on first-order terms over signatures with similar constructors: A constraint-based approach, Fuzzy Sets and Systems 391 (2020) 1–46. doi:10.1016/j.fss.2019.03.019.
[22] P. Julián-Iranzo, F. Sáenz-Pérez, Bousi∼Prolog: Design and implementation of a proximity-based fuzzy logic programming language, Expert Systems with Applications 213 (2023) 118858. doi:10.1016/j.eswa.2022.118858.
[23] P. Julián-Iranzo, G. Moreno, J. A. Riaza, The Fuzzy Logic Programming language FASILL: Design and implementation, International Journal of Approximate Reasoning 125 (2020) 139–168. doi:10.1016/j.ijar.2020.06.002.
[24] J. Cheng, Z. M. Ma, L. Yan, f-sparql: A flexible extension of sparql, in: P. G. Bringas, A. Hameurlain, G. Quirchmayr (Eds.), Database and Expert Systems Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 487–494. doi:10.1007/978-3-642-15364-8_41.
[25] J. M. Almendros-Jiménez, A. Becerra-Terón, G. Moreno, J. A. Riaza, Tuning fuzzy sparql queries, International Journal of Approximate Reasoning 170 (2024) 109209. doi:https://doi.org/10.1016/j.ijar.2024.109209.
[26] G. Fakih, P. Serrano-Alvarado, A survey on sparql query relaxation under the lens of rdf reification, Semantic Web 15 (2024) 2507–2554. doi:10.3233/SW-243621.
[27] H. Aït-Kaci, S. Amir, Classifying and Querying Very Large Taxonomies, Technical Report, LIRIS, Département d'Informatique, Université Claude Bernard Lyon 1, Villeurbanne, France, 2013. URL: https://projet.liris.cnrs.fr/cedar/papers/ctr2.pdf, CEDAR Technical Report Number 2, CEDAR Project.
[28] H. Aït-Kaci, R. Boyer, P. Lincoln, R. Nasr, Efficient implementation of lattice operations, ACM Trans. Program. Lang. Syst. 11 (1989) 115–146. doi:10.1145/59287.59293.
[29] G. C. Milanese, G. Pasi, Conjunctive Reasoning on Fuzzy Taxonomies with Order-Sorted Feature Logic, in: 2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2021, pp. 1–7. doi:10.1109/FUZZ45933.2021.9494474.
[30] H. Aït-Kaci, A. Podelski, S. C. Goldstein, Order-sorted feature theory unification, The Journal of Logic Programming 30 (1997) 99–124. doi:10.1016/S0743-1066(96)00053-2.