Similarity-Based Reasoning and Retrieval with Order-Sorted Feature Logic

Discussion Paper

Gian Carlo Milanese, Gabriella Pasi

IKR3 Lab, University of Milano-Bicocca, Viale Sarca 336, 20126 Milan, Italy

Abstract

Logic programming languages based on first-order terms (FOTs) have been extended with a similarity relation on functor symbols in order to perform approximate reasoning, which may allow flexible querying and retrieval from a knowledge base. More flexibility is also provided by the terms of Order-Sorted Feature (OSF) logic, which relax a few syntactic restrictions of FOTs. Moreover, OSF term unification takes into account a subsumption ordering on sort symbols, which makes computations more efficient. This document presents the current efforts by the authors in defining similarity-based reasoning with OSF logic: the goal is to achieve a more flexible, but still efficient, processing of queries.

Keywords

Similarity-based retrieval, Approximate reasoning, Flexible querying, OSF logic, Logic programming

1. Introduction

There is considerable interest in extending logic programming languages for performing approximate reasoning. One approach proposed in the literature consists in considering a similarity relation (a fuzzy equivalence relation) between functors in order to represent approximation at a syntactic level, with the goal of performing approximate reasoning on the basis of *analogy* or *similarity* [1, 2]. Sessa [3] introduced both a variation of the standard unification algorithm for first-order terms (FOTs) called *weak unification* and a modification of SLD resolution that allows to overcome failures of the exact matching between functors if they are related by a non-zero similarity value. From a retrieval perspective, answers to a query posed to a knowledge base will consist not only of exact matches, but will also include instances that satisfy the constraints of the query with some approximation (an example will be provided in Section 2).

Sessa's weak unification algorithm has been extended to also support a proximity relation on functor symbols [4]. Proximity relations generalize similarity relations by relaxing the requirement that the relation be transitive, which makes them more suitable to model knowledge in some situations [5, 4]. Both Sessa's algorithm and this extension have been implemented in the fuzzy logic programming language Bousi∼Prolog [6, 4].

Recently, the similarity-based approach to approximate reasoning has been further developed by Aït-Kaci and Pasi [7]. Their approach not only tolerates different (but similar) functor

IIR 2021 – 11th Italian Information Retrieval Workshop, September 13–15, 2021, Bari, Italy

g.milanese1@campus.unimib.it (G. C. Milanese); gabriella.pasi@unimib.it (G. Pasi)

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

symbols, but also allows the unification of FOTs with a different number and possibly a different order of arguments. A possible incorporation of this approach in Bousi∼Prolog has been presented [8].

The work by Aït-Kaci and Pasi was preliminary towards defining similarity-based reasoning and retrieval with *Order-Sorted Feature* (OSF) logic, a knowledge representation and reasoning language that originates in Aït-Kaci's PhD thesis [9]. OSF logic not only provides more flexibility in the representation of knowledge when compared to ordinary FOTs (for instance, by replacing FOT argument positions by feature symbols, which also improves interpretability), but the OSF term unification algorithm can also take into account a *subsumption* (is-a) ordering between sort symbols (interpreted as concepts or types) to perform computations more efficiently [10, 11].

This document presents the current efforts by the authors to further develop the research initiated by Aït-Kaci and Pasi. The goal is to define similarity-based reasoning with OSF logic in order to achieve more flexibility in the processing of queries, while maintaining the efficiency of this language. The next section briefly reviews similarity-based reasoning with FOTs, while Section 3 presents the current challenges in developing this approach with OSF logic.

2. Similarity-Based Reasoning and Retrieval with FOTs

Similarity-based reasoning with FOTs is presented here with an example written in Bousi∼Prolog [6]. Consider the knowledge base of Fig. 1: the rule expresses that Celeste likes horror movies, and the facts state that Psycho is a horror movie, while Memento is a thriller. The only difference with a Prolog program consists in the similarity equation specifying that horror movies and thrillers are similar with approximation degree 0.7.

```
% Rules
likes("Celeste", X) :- horror(X)
% Facts
horror("Psycho")
thriller("Memento")
% Similarity Equations
horror ~ thriller = 0.7
```

Figure 1: Bousi∼Prolog knowledge base

Consider the query `likes("Celeste", X)` which aims to retrieve entities (in this case movies) liked by Celeste. In order to answer this query, a Prolog interpreter would first infer the fact `horror(X)` based on the unification with the first rule, and would then try to unify this latter term with `horror("Psycho")` and `thriller("Memento")`. In the first case the unification succeeds since the functor `horror` can be matched with itself, and Prolog would thus answer `X = "Psycho"`; the second unification would instead fail, since the two functors `horror` and `thriller` are different and thus cannot be matched.

Sessa's weak unification relaxes the equality constraint and allows matching different, but similar, functors [3]. Indeed, to the same query Bousi∼Prolog would also answer `X = "Memento"` with approximation degree 0.7, since the functors `horror` and `thriller` are specified to be similar with this degree. Informally, the weak unification and SLD resolution of [3] allow to perform reasoning by analogy or similarity: if Celeste likes horror movies and horror movies are similar to thrillers, then to some degree Celeste may also like thriller movies, so that "Memento" is a relevant answer to the posed query.

3. Similarity-Based Reasoning and Retrieval with OSF Logic

OSF logic [12] is based on sort symbols and feature symbols: sort symbols denote concepts or classes of objects, while feature symbols are interpreted as properties or attributes of sorts. Knowledge can be represented using OSF terms, which are record structures built using sort symbols, feature symbols, and variables. An example of an OSF term is the following:

$$X : \texttt{movie}(\texttt{genre} \rightarrow \texttt{thriller}, \texttt{title} \rightarrow Y : \texttt{string}) \tag{1}$$

where `movie` and `thriller` are sort symbols, `genre` and `title` are feature symbols, and X and Y are variables denoting objects of the domain (e.g., X must denote a movie, while Y must denote a string). OSF terms are a generalization of FOTs: for instance, the FOT `movie("Psycho", "Hitchcock")` can be directly translated into OSF syntax as $\texttt{movie}(1 \rightarrow \texttt{"Psycho"}, 2 \rightarrow \texttt{"Hitchcock"})$, or extended using feature symbols into $\texttt{movie}(\texttt{name} \rightarrow \texttt{"Psycho"}, \texttt{director} \rightarrow \texttt{"Hitchcock"})$ for added interpretability.

Sort symbols are ordered by a subsumption (is-a) relation $\preceq$, which formally is a partial order. This ordering is taken into account during the unification of OSF terms: for example, if $\texttt{horror} \preceq \texttt{movie}$ (i.e., the sort `movie` subsumes the sort `horror`), then the sort `horror` can be matched with the sort `movie`. This would not be possible with the ordinary unification of FOTs, where instead an instance of the sort `horror` would be inferred to also be of sort `movie` through SLD resolution, which in general may take many more steps. During OSF term unification two sorts can also be matched when there exists a most general sort that is subsumed by both sorts (called *most general common subsort*): for instance, if $\texttt{dog} \preceq \texttt{canid}$ and $\texttt{dog} \preceq \texttt{pet}$, then `canid` and `pet` may be matched.

Because the partial order $\preceq$ can be represented as a directed acyclic graph, thanks to graph encoding techniques deciding subsumption or computing the most general common subsort can be performed in constant time [13]. For this reason and because a single order-sorted unification step may possibly replace many SLD resolution steps, derivations with OSF logic can be much shorter than those of unsorted languages [10]. The advantages of many-sorted and order-sorted logic with respect to unsorted logics are also discussed in [11].

One of the main challenges in developing a *weak similarity-based version* of order-sorted unification for OSF logic thus consists in understanding how to take into account both the sort subsumption relation and the sort similarity relation while matching sort symbols. Most importantly, the matching must be as *efficient* as possible.

Besides deciding subsumption and computing most general common subsorts, in a weak OSF term unification procedure it should be possible to efficiently perform the following operations between two sort symbols s and t:

1. deciding whether the two sorts are similar with degree $\alpha \in (0, 1]$, written $s \sim_\alpha t$;
2. deciding whether s is subsumed by a sort u that is similar to t, written $s \preceq u \sim_\alpha t$;
3. deciding whether s is similar to a sort u that is subsumed by t, written $s \sim_\alpha u \preceq t$;
4. more generally, deciding whether there exists a chain of sorts

$$s \sim_{\alpha_0} s_0 \preceq t_1 \sim_{\alpha_1} s_1 \preceq t_2 \cdots s_{n-1} \preceq t_n \sim_{\alpha_n} t$$

with alternating subsumption and similarity links and associated degree $\alpha = \min_{0 \leq i \leq n} \alpha_i$ (taking the minimum corresponds to choosing the weakest similarity link, but alternatives are also possible). A chain of this kind will be referred to as a *similarity-subsumption chain* from now on.

For example, if we know that $\texttt{slasher} \preceq \texttt{horror} \sim_{0.7} \texttt{thriller}$, then it should be possible to match `slasher` and `thriller` while unifying, for instance, the term (1) above with the term $X : \texttt{movie}(\texttt{genre} \to \texttt{slasher})$. In this case, the result of the unification will be associated with the approximation degree 0.7.

In order to perform these operations efficiently, the following strategy is possible:

1. A threshold $\tau \in (0, 1]$ for the minimum similarity degree is chosen so as to avoid matching sort symbols that are considered too dissimilar, e.g., $\tau = 0.6$;
2. For each sort symbol s, the similarity class $[s]_\tau = \{t \mid s \sim_\alpha t, \alpha \geq \tau\}$ is computed: this is the set of all sorts t that are similar to s with degree at least τ;
3. A new partial order $[\preceq]_\tau$ can be defined between the similarity classes of sort symbols;
4. This new partial order can be represented as a directed graph, so that encoding techniques (like the one presented in [13]) can be used in order to compute in constant time whether $([s]_\tau, [t_\tau]) \in [\preceq]_\tau$, which means that it is possible to know in constant time whether a similarity-subsumption chain exists between s and t with associated degree at least τ.

The issue with this approach is that it does not provide the actual degree α associated with such a chain, but only specifies that this chain exists and that its associated degree is greater than τ. A solution to efficiently compute the values associated with these similarity-subsumption chains is currently being investigated.

4. Conclusion

This document has shortly presented an ongoing project finalised at the extension of OSF logic with the capability of performing similarity-based reasoning, which would allow flexibility in the processing of queries posed to a knowledge base, as exemplified in Section 2 with FOTs. In order to achieve this, it is necessary to define how to efficiently match two sort symbols not only when they are similar or they subsume each other (or have a most general common subsort), but also when there exists a similarity-subsumption chain that connects them, as argued in Section 3. Once this is settled and similarity-based reasoning with OSF logic is defined, the next steps may involve the implementation of a language based on this approach (e.g., a fuzzy extension of LOGIN [10]), or generalizing the approach to also support proximity relations as done in [4] for Bousi∼Prolog, which may be more appropriate in some modeling situations [5].

References

[1] F. Formato, G. Gerla, M. I. Sessa, Similarity-based unification, Fundamenta Informaticae 41 (2000) 393–414. doi:10.3233/FI-2000-41402.

[2] M. I. Sessa, Translations and similarity-based logic programming, Soft Computing 5 (2001) 160–170. doi:10.1007/PL00009891.

[3] M. I. Sessa, Approximate reasoning by similarity-based SLD resolution, Theor. Comput. Sci. 275 (2002) 389–426. doi:10.1016/S0304-3975(01)00188-8.

[4] P. Julián-Iranzo, C. Rubio-Manzano, Proximity-based unification theory, Fuzzy Sets Syst. 262 (2015) 21–43. doi:10.1016/j.fss.2014.07.006.

[5] S. Shenoi, A. Melton, Proximity relations in the fuzzy relational database model, Fuzzy Sets and Systems 100 (1999) 51 – 62. doi:10.1016/S0165-0114(99)80006-2.

[6] P. Julián-Iranzo, C. Rubio-Manzano, J. Gallardo-Casero, Bousi∼Prolog: a Prolog extension language for flexible query answering, Electronic Notes in Theoretical Computer Science 248 (2009) 131–147. doi:10.1016/j.entcs.2009.07.064.

[7] H. Aït-Kaci, G. Pasi, Fuzzy lattice operations on first-order terms over signatures with similar constructors: A constraint-based approach, Fuzzy Sets Syst. 391 (2020) 1–46. doi:10.1016/j.fss.2019.03.019.

[8] M. E. Cornejo, J. Medina Moreno, C. Rubio-Manzano, Towards a full fuzzy unification in the Bousi∼Prolog system, in: 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2018, pp. 1–7. doi:10.1109/FUZZ-IEEE.2018.8491514.

[9] H. Aït-Kaci, A Lattice-Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures, Ph.D. thesis, University of Pennsylvania, Philadelphia, 1984.

[10] H. Aït-Kaci, R. Nasr, Login: a logic programming language with built-in inheritance, The Journal of Logic Programming 3 (1986) 185–215. doi:10.1016/0743-1066(86)90013-0.

[11] A. G. Cohn, Taxonomic reasoning with many-sorted logics, Artificial intelligence review 3 (1989) 89–128. doi:10.1007/BF00128778.

[12] H. Aït-Kaci, A. Podelski, Towards a meaning of LIFE, J. Log. Program. 16 (1993) 195–234. doi:10.1016/0743-1066(93)90043-G.

[13] 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.