Fuzzy Prolog Search

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

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

October 29, 2019

Abstract

This formalizes a "best-so-far next" strategy for fuzzy Logic Programming ($\mathcal{FLP}$) that generalizes the default Left-Right/Depth-First strategy of Standard Prolog. Fuzzification is introduced through unification operations on standard First-Order Term ($\mathcal{FOT}$s) modulo a similarity (*i.e.*, a fuzzy equivalence relation) on the signature of functors that occur in these $\mathcal{FOT}$s. The criterion for what "always best" means in this setting is to garantee the highest confidence level for any possible query answer. This confidence level is identified to the maximal fuzzy degree over all computable answers. However the number of the latter being unbounded in the general case, this is not achievable. We propose to use instead a "best-so-far" strategy defined as the highest confidence level over all the possible fuzzy unifications modulo a functor similarity on the signature between a goal to prove and the clause heads of a fuzzy Prolog program consisting of an ordered sequence of Horn clauses. Since garanteeing always finding first the highest possible approximate answer to a query is not possible, this compromise still provides the best answer it knows about taking into account the available information accumulated at this point.

Keywords: Prolog; First-Order Term; Functor-Signature Similarity; Unification with Similar Functors; Fuzzy Logic Programming; Fuzzy Pattern-directed Reasoning; Approximate Information Processing.

1 Introduction

1.1 Motivational example

Let us consider the Prolog program's rule base:

```
happiness :- freedom, nice_weather.

nice_weather :- sunny.
nice_weather :- partly_cloudy, breezy.

freedom :- being_on_holiday.
```

together with the fact base:

```
partly_cloudy.
windy.
staying_at_home.
```

and the query:

```
?- happiness.
```

The transformation of this query by Prolog's default search strategy is:

```
?- happiness.
?- freedom, nice_weather.
?- being_on_holiday, nice_weather.
```

and since no fact in the fact base unifies with `being_on_holiday`, this can proceed no further; so Prolog answers:

```
No
```

Since it relies on a strict true/false binary logic, this negative answer, as well as any other positive answer given by Prolog, is always given with a 100% confidence level.

Let us now consider the *same* Prolog program where terms are built with the *same* functor symbols, this time subject to the similarity closure of the following fuzzy pairs of functors (see [5] for details on how to compute such a similarity closure):

$$\texttt{staying\_at\_home} \sim_{0.8} \texttt{being\_on\_holiday}$$
$$\texttt{sunny} \sim_{0.7} \texttt{partly\_cloudy}$$
$$\texttt{windy} \sim_{0.9} \texttt{breezy}$$

Now, the transformation of the *same* query '`?- happiness.`' in the context of the *same* Prolog program by a fuzzy Prolog using the fuzzy unification defined in [5] (see Appendix Section A.2) with Prolog's default *Left-Right/Depth-First* (LR/DF) search strategy is:[1]

```
[1.0] ?- happiness.
[1.0] ?- freedom, nice_weather.
[1.0] ?- being_on_holiday, nice_weather.
```

but now our functor similarity says `staying_at_home` $\sim_{0.8}$ `being_on_holiday`, so the query becomes fuzzy:

```
[0.8] ?- nice_weather.
[0.8] ?- sunny.
```

and so, using the first unifiable clause for `nice_weather`, since `sunny` $\sim_{0.7}$ `partly_cloudy`, this fuzzy Prolog will answer:

```
[0.7] Possibly
```

to the query `happiness` with a confidence level of 0.7.

This answer is less categorical than the one given by Prolog to the crisp interpretation of the same query for the same program. However, this answer is not one that can be obtained with

[1] We extend our trace of this fuzzy Prolog's current query transformation at each resolution step with its possibility level: "`[`α`]?` *query*." (where $0 < \alpha < 1$), and its final answer as one of:

- `Yes` — for crisp true;
- `[`α`] Possibly` — for a fuzzy possibility with confidence level α (where $0 < \alpha < 1$);
- `No` — for crisp false.

The non-negative answers (`Yes` and `Possibly`) may be accompanied with a query-variable substitution if any.

highest confidence. Indeed, if we proceed with a different strategy consisting, not of selecting (1) the leftmost goal in the query and (2) and earliest fact or rule head in the program that are *fuzzy-unifiable whatever the confidence level may be*, but instead (1) the leftmost goal in the query and (2) earliest fact or rule head in the program that are *fuzzy-unifiable with highest confidence*, a smarter fuzzy Prolog will have selected not the first but the second rule defining **nice_weather**, since the confidence level is 1.0 (because **partly_cloudy** unifies with itself), yielding the new fuzzy query:

```
[0.8] ?- breezy.
```

and, since **windy** $\sim_{0.9}$ **breezy** and **windy** is a fact, this smarter fuzzy Prolog will answer:

```
[0.8] Possibly
```

which is a more confident first answer than the one obtained by a fuzzy Prolog using LR/DF search.

One may then wonder how to set up control so that the **[0.8]**-confidence answer be given in priority over the **[0.7]**-confidence answer. In this case, it would suffice to select the clause whose head literal fuzzy-unifies with the current goal with highest-confidence level. However, in general this is not sufficient since even if clause-selection is done in this manner, this cannot guarantee that the confidence level will not decrease further down this proof branch to a value that will be strictly less than a previous (necessarily non-failing) node of the and/or proof-tree (and in particular any previously visited successful leaf nodes).

In fact, since Horn-clause resolution is undecidable, this too is obviously undecidable — even when using breadth-first despite its generally higher book-keeping cost rather than simple depth-first since it is the only search strategy garanteed to be complete.[2] Still, since this is also obviously a better choice than the one at hand, we can keep going down this proof branch as long as it has not reached a proof-tree node whose confidence level is actually less than or equal to the latest proof-tree node with higher confidence level that was previously visited but not fully resolved. If this happens (*i.e.*, the current node's confidence level falls too low), the current proof-tree state is saved for a possible later resumption and control reinstates the previously frozen state and resumes the proof from there.

This is what the control strategy we formalize in this document does.

1.2 Relation to other work

There have been other fuzzifications of Logic Programming. For the reader curious to situate this paper in the general existing research context, we next brush a quick relation to other work before we develop our specific perspective. It is beyond this paper's scope to be exhaustive. We simply list those works that we found to be the most closely related to ours in chronological order of publication.[3] For a thorough discussion and other systems, the reader is referred to [5] and [4].

- 1995 — Fuzzy extension of Datalog [1];[4];

[2] Completeness of a proof strategy means that it is garanteed to enumerate all existing finitely derivable solutions. For example, replacing the **freedom** clause in the above program with:

```
freedom :- staying_at_home, being_on_holiday.
```

will make our query have an infinite proof tree.

[3] Ironically, "*most closely related*" being a fuzzy property, this constitutes an interesting meta-example of fuzzy Prolog approximate query seeking a maximal confidence level where the similarity level of two specific approaches to fuzzy Logic Programming is left to one's appreciation.

[4] http://people.inf.elte.hu/kiss/14abea/Achs_1995_ActaCybernetica.pdf

- 1997 — Likelog (another fuzzy datalog) [8];[5]
- 2002 — LP with context-dependent fuzzy unification [6];[6]
- 2002 — Fuzzy Prolog using CLP($\mathbb{R}$); not a fuzzy CLP($\mathbb{R}$) programming language, but using CLP($\mathbb{R}$) to implement fuzzy operations on union of intervals of real numbers [25];[7]
- 2010 — Bousi~Prolog a fuzzy Prolog using a weak version of fuzzy unification [15];[8]
- 2015 — FASILL a fuzzy logic programming language with implicit/explicit truth degree annotations, extending Bousi~Prolog by concurrently allowing many different user-definable fuzzy connectives in a same application and thus several user-definable interpretations of fuzzy unification by similarity; [17];[9]
- 2018 — Bousi~Prolog using Aït-Kaci/Pasi unification [10].[10]

Whereas this previous work specified several possibilities for dealing with approximate reasoning by fuzziness in Logic Programming, we did not find any formally addressing the specific issue of confidence-driven control and its implication on implementation. What we propose is a formal specification of a generic control protocol seeking to maximize the confidence level of a query's answer provided while extending Prolog's standard left-right/depth-first search strategy. The latter is obtained as the special case corresponding to when functor similarity is restricted to symbol identity only. Our proposed generic best-first search is guided by a heuristic valuation of confidence level determined by the accumulated approximation degree of the all the partial and definitive proofs carried out so far at any point of computation. The challenge is then that the stack-oriented architecture of standard Prolog is no longer viable. Partial states of computation must be kept sorted in order of fuzzy approximation degrees, each state being saved with all the information needed to resume from (such as partial solutions as variable substitutions possibly conflicting with those in later distinct but more complete proof branches).

1.3 Organization of this document

The rest of this document formalizes and operationalizes a "best-so-far" strategy for Fuzzy Logic Programming ($\mathcal{FLP}$) that generalizes the default LR/DF strategy of Standard Prolog. Section 2 presents a formal operational semantics of Fuzzy Prolog extending the standard resolution semantics of Prolog (recalled in Appendix Section A.1). This uses the fuzzy First-Order Term ($\mathcal{FOT}$) unification operations modulo a similarity (*i.e.*, a fuzzy equivalence relation) on the signature of functors used by the $\mathcal{FOT}$s developed in [5] and recalled in Appendix Section A.2.

2 Fuzzy Prolog Resolution

The criterion for what "best so far" means in this setting is the highest confidence level given at any point by the maximal fuzzy degree over all the possible fuzzy unifications between a goal to prove and all the clause heads of a program consisting of an ordered sequence of Horn clauses. When such a maximal degree corresponds to several possible Horn clauses, their order as specified

[5] http://www.programmazionelogica.it/[...]/uploads/1997/06/319_Fontana1.pdf
[6] http://repositori.udl.cat/bitstream/handle/10459.1/57984/001858.pdf
[7] https://cliplab.org/papers/fuzzy-lpar02_bitmap.pdf
[8] http://www.sciencedirect.com/science/article/pii/S1571066109002874
[9] https://arxiv.org/pdf/1501.02034.pdf
[10] https://[...]Towards_a_Full_Fuzzy_Unification_in_the_Bousi_Prolog_system

in the program prevails. Thus, in the crisp case (*i.e.*, the identity: the empty functor similarity), this strategy coincides with the default (LR/DF) search of Standard Prolog.

Following Standard Prolog's formal computational semantics briefly recalled in Appendix Section A.1, we shall now use the same notation to specify the way our fuzzification of Prolog proceeds by always choosing the goal/program-head resolving-pair that is fuzzy-unifiable with the highest confidence level at any point of computation. If there exist several goal/program-head resolving-pairs with an equal confidence level, computation follows Standard Prolog's LR/DF order. If there exists none (there is no pair of goal/head that fuzzy-unifies with a positive confidence level), computation proceeds by backtracking to the latest choice-point of highest confidence level. If there is no saved backtrack point, computation terminates with a "**No**" answer.

DEFINITION 1 A *fuzzy Prolog query* q_α is a finite ordered sequence of goals calibrated by a confidence level $0 < \alpha \leq 1$:

$$\texttt{[}\alpha\texttt{]}\ \texttt{?-}\ g_1,\ \ldots,\ g_n. \tag{1}$$

with $n \geq 0$, and each goal g_i is a $\mathcal{FOT}$, for $1 \geq n$.

As in the body of a Standard Prolog query, the comma character ',' in a fuzzy query denotes logical conjunction. Yet, this is now fuzzy conjunction as this will participate in the fuzzy query resolution transformation rule according to its confidence level.

Given a Prolog program $c_1.\ \ldots\ c_m.$, fuzzy Prolog transforms a fuzzy query:

$$\texttt{[}\alpha\texttt{]}\ \texttt{?-}\ g_1,\ \ldots,\ g_n.$$

into the new fuzzy query:

$$\texttt{[}\gamma\texttt{]}\ \texttt{?-}\ g_1\sigma^\star,\ \ldots,\ g_{i^\star-1}\sigma^\star,\ \mathbf{body}(c'_{j^\star})\sigma^\star,\ g_{i^\star+1}\sigma^\star,\ \ldots,\ g_n\sigma^\star.$$

where $\gamma \stackrel{\text{def}}{=} \alpha \wedge \beta^\star$ and $i^\star \in \{1,\ldots,n\}$ and $j^\star \in \{1,\ldots,m\}$ are the least indices, and $\sigma^\star$ the most general substitution such that:

$$\beta^\star \stackrel{\text{def}}{=} \bigvee_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}} \{\, \beta \mid \exists \sigma \text{ s.t. } g_i\sigma \sim_\beta \mathbf{head}(c'_j)\sigma \,\} \tag{2}$$

and where $\langle i^\star, j^\star, \sigma^\star \rangle$ is a triple consisting of the smallest goal index $i^\star$, the smallest program index $j^\star$, and most general substitution $\sigma^\star$, such that $g_{i^\star}\sigma^\star \sim_{\beta^\star} \mathbf{head}(c'_{j^\star})\sigma^\star$ for the value of $\beta^\star$ given by Equation (2).

If no such indices $i^\star$, $j^\star$ and substitution $\sigma^\star$ exist such that $\gamma > 0.0$, fuzzy Prolog transforms the query into the failing query $\mathfrak{false}$, and answers "**No**."

If $\gamma > 0.0$ but $\mathbf{body}(c'_{j^\star}) = \mathfrak{true}$, the new fuzzy query is:

$$\texttt{[}\gamma\texttt{]}\ \texttt{?-}\ g_1\sigma^\star,\ \ldots,\ g_{i^\star-1}\sigma^\star,\ g_{i^\star+1}\sigma^\star,\ \ldots,\ g_n\sigma^\star.$$

i.e., it erases the goal previously at index $i^\star$ in the query. If $\gamma > 0.0$ but the new query is empty (*i.e.*, $\mathfrak{true}$), fuzzy Prolog, answers "**Yes**" if $\gamma = 1.0$ or "**[γ] Possibly**" if $\gamma < 1.0$, and in either case a query-variable substitution, if any, is printed along the answer.

The formal properties that must hold for such a desired fuzzy Prolog are the following.

PROPERTY 1 (MONOTONICALLY DECREASING CONFIDENCE LEVEL) Fuzzy Prolog's search must provide alternative fuzzy answers to a query according to an order of monotonically decreasing confidence level.

This makes natural sense as one would prefer answers of higher confidence level first.

PROPERTY 2 (PROLOG SEARCH AS A SPECIAL CASE) Fuzzy Prolog's search must become Prolog's backtracking search for the identity similarity.

This is intuitively reassuring since it is the default case.

LEMMA 1 (PROLOG SEARCH AS A SPECIAL CASE) Fuzzy Prolog's search preferring first highest confidence levels, and LR/DF if there are several, corresponds to Prolog's backtracking search for the identity similarity.

However, while Property 2 holds in our fuzzy Prolog, Property 1 does not hold due to undecidability of our fuzzy Prolog, which is a corollary of Prolog's since it is a special case of our fuzzy Prolog.

COROLLARY 1 (UNDECIDABLE MONOTONICALLY DECREASING CONFIDENCE LEVEL) It is undecidable for Fuzzy Resolution to be complete following a search choosing in priority the earliest alternative of highest confidence level to enumerate alternative fuzzy answers to a query according to an order of monotonically decreasing confidence level.

The fuzzy resolution strategy we presented above regarding the order in which alternative answers are to be collected by fuzzy Prolog may be formally satisfying. However, many a reader must have wondered its utility taking into account the high number of fuzzy unifications that must be computed in order to keep the fuzzy query organized so as to provide answers in decreasing confidence levels. This is a problem, indeed. Be that as it may, we next show how this can be minimized, and even tolerated given the reasoning flexibility provided by fuzzy Prolog, with little change. We keep Prolog's data structures as they are and only slightly modify its management of control (since this is all that is needed) using static information. This can drastically limit the needed pairs of goal/program-head indices that should be tested in the dynamic search for the next fuzzy resolution given a statically defined functor similarity. We explain this in the next section.

3 Fuzzy Prolog's Control Management

In addition to having to manage multiple criss-cross fuzzy unifications, a more confident fuzzy Prolog's backtracking must return to pursuing the incomplete proof of highest confidence level

not only upon failing at the end of the current search branch but as soon as *it reaches a confidence level that is strictly less than that the maximal level recorded earlier.*[11] In other words a proof branch with a non-zero confidence level is not to be discarded when its confidence level reaches a lower level than a previous incomplete proof branch. This is because it is still live and can be revived should its confidence level ever prevails again. Because of this, control should not discard such an incomplete branch, *it should only suspend it*; *i.e.*, save it in a state to resume from where control had suspended it.[12] Indeed, it is possible for control to resume transforming this incomplete fuzzy query at this yet-unfinished computation point as soon as its confidence level becomes higher than or equal to the current one being proven. Note that the condition is $\geq$ to maintain preference for LR/DF order upon equal confidence levels. It makes then sense always to keep choice-points sorted by highest confidence level first. This entails saving and restoring partial computations depending on the highest prevailing confidence level, with equal confidence-level choices being ordered chronologically as in Standard Prolog. In addition, choice-points must be kept live as long as they lead to resumable computation. This complicates the pure stack-based architecture used for Standard Prolog.

However, there are also many observations that could, and should, be made in favor of this operational semantics of fuzzy Prolog to be worthy of pursuit, especially taking into account the interpretation expressive flexibility fuzzy Prolog provides over strict Prolog. Here are the most notable we propose to use to some advantage:

- a fuzzy Prolog using the foregoing control strategy does not fuzzify programs (*i.e.*, rules and facts); rather, it only fuzzifies *unification* by tolerating that some functors in the signature be similar, and in this way it can always quantify a visited node in the proof tree with a confidence level (and therefore its answers, since they are as proof-tree leaf nodes);
- the similarity over all the known functors is static;
- our fuzzy Prolog is an extension of "pure" Prolog — the rules and facts are static;
- therefore, static partial cross-unification of query goals and program head terms can be computed to avoid being done over and over at runtime;
- in addition, the number of such fuzzy-unifiable goal/head pairs is likely to be very small;
- even though the stack-architecture advantage offered by LR/DF is compromised (since backtracking from an unfinished branch is not definitive), our fuzzy Prolog control still has a flavor of always "*trying the next available option*" — only now it needs to keep state stacks somehow organized according to confidence and keep track of where each next highest-confidence level query needs to resume from, since it may be to one suspended much earlier than the one that would be on top of a standard depth-first stack of disjunctive proof branches. And when going back to such an earlier greater-confidence level, control-switch cannot just discard all its work since the last choice point unless it reached a definitive failure point (standard backtracking).

[11] Why not just the *latest* or *earliest* recorded incomplete branch? Because we should give preference to the *highest confidence* level among the earliest potential standard Prolog proof.

[12] Purists might call this a fuzzy-proof continuation in the manner of continuation-passing style proofs [12].

A Appendix

A.1 Prolog resolution

In this section we recall very briefly some well-known characteristics of Prolog along with notation conventions we follow in the rest of this paper.

DEFINITION 2 A *Prolog program* is a finite ordered sequence of Horn clauses of the form:

$$\begin{array}{l} c_1. \\ \vdots \\ c_m. \end{array} \tag{3}$$

where $m \geq 0$.

As a logical reading, a Prolog program denotes the ordered disjunction of its clauses (the order being that in which they appear in the program).

DEFINITION 3 A *Horn clause* c is of the form:

$$h \texttt{ :- } b_1, \ \ldots, \ b_k. \tag{4}$$

where $\textbf{head}(c) \stackrel{\text{def}}{=} h$ is a $\mathcal{FOT}$ and $\textbf{body}(c) \stackrel{\text{def}}{=} b_1, \ \ldots, \ b_k$, $k \geq 0$, is a possibly empty finite ordered sequence of $\mathcal{FOT}$s.

In a Horn clause, the symbol '`:-`' stands for '$\leftarrow$' (backward implication) and denotes logical "*if*," while the comma character ',' denotes logical "*and*" (conjunction). A Horn clause with an empty body stands for the form "h `:-` `true`." and is written simply "h." and is called a *fact*; a Horn clause with a non-empty body is called a *rule*.

DEFINITION 4 A *Prolog query* q is a finite ordered sequence of goals of the form:

$$\texttt{?-}\ g_1, \ \ldots, \ g_n. \tag{5}$$

where $n \geq 0$, and each goal g_i is a $\mathcal{FOT}$, for $1 \geq n$.

As in the body of a clause, the comma character ',' in a query denote logical conjunction. A query may be seen as the body of a headless Horn clause (*i.e.*, a Horn clause whose head is `true`).

Standard Prolog's operational semantics consists in establishing the proof of a query given a Prolog program using Robinson's *resolution* [23]. It is a query transformation process that is considered successful if it terminates with the empty query (written `true`, the idempotent—or identity—element for conjunction). It is considered unsuccessful if it terminates with the failing query (written `false`, the absorptive—or zero—element for conjunction). Determining whether this process terminates for any query and program is undecidable since one can easily encode a Turing Machine in Prolog.[13,14]

[13] https://www.metalevel.at/prolog/showcases/turing.pl
[14] Or see a simpler one on https://en.wikipedia.org/wiki/Prolog.

Prolog uses a search strategy we shall call LR/DF, which stands for "left-right/depth-first." It is "left-right" because it establishes query goals in the order they are written.[15] It is "depth-first" because it will select first the earliest Horn clause of a program having a head that unifies with the goal being proven. This is made operational using backtracking control (which is conveniently implementable with an architecture of stacks). This is expressed formally using the above terminology and notation as follows.

Given a program $c_1.\ \ldots\ c_m.$, Prolog transforms a query $?-\ g_1, \ldots, g_n.$ into the new query:

$$?-\ \mathbf{body}(c'_j)\sigma,\ g_2\sigma,\ \ldots,\ g_n\sigma.$$

where $1 \leq j \leq m$ and j is the least index such that there exists a substitution σ with $g_1\sigma = \mathbf{head}(c'_j)\sigma$ where c'_j is a consistent variable-renaming copy of c_j.

If no such index j and substitution σ exist, Prolog transforms the query into the failing query $\mathfrak{false}$, and answers "**No**."

If $\mathbf{body}(c'_j) = \mathfrak{true}$, the new query is $?-\ g_2\sigma,\ \ldots,\ g_n\sigma$.

If the new query is empty (*i.e.*, $\mathfrak{true}$), Prolog prints the resulting query-variable substitution if any and answers "**Yes**."

This works for Prolog, defining its LR/DF strategy.

In the next section, we consider a fuzzy Prolog whose syntax of $\mathcal{FOT}$s, rules, and facts are identical to Prolog's, but where the interpretation is resolution giving preference to a "best-first" choice of resolving pairs rather than a LR/DF. Such a fuzzy Prolog interprets Standard Prolog programs (*i.e.*, Horn clauses over $\mathcal{FOT}$s). It also transforms Prolog queries, only this time it does so with a fuzzy confidence level computed using fuzzy unification of $\mathcal{FOT}$ over functor symbols that may be related with a fuzzy equivalence relation as formally defined in [5].

A.2 Fuzzy $\mathcal{FOT}$ unification

This summarizes the fuzzy $\mathcal{FOT}$ unification operation setting presented in [3] and formally justified in detail in [5].

Sessa's weak unification A fuzzy unification operation on $\mathcal{FOT}$s, dubbed "*weak unification*," was proposed by Maria Sessa in [24] which consists in normalizing fuzzy equations between conventional $\mathcal{FOT}$s modulo a similarity relation $\sim$ over functor symbols [11]. This similarity relation is then homomorphically extended to one over all $\mathcal{FOT}$s. Following Maria Sessa's formal setting [24], 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 5 (SESSA'S $\mathcal{FOT}$ SIMILARITY) The fuzzy relation $\sim^{\mathcal{T}}$ on $\mathcal{T}_{\Sigma,\mathcal{V}}$ is defined inductively as follows:

1. $\forall\, X \in \mathcal{V},\ \ X \sim^{\mathcal{T}}_1 X$;
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 $f \in \Sigma_n$ 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 $s_i \sim^{\mathcal{T}}_{\alpha_i} t_i$ for $i = 1, \ldots, n$, then:

$$f(s_1, \ldots, s_n) \sim^{\mathcal{T}}_{\alpha \wedge \bigwedge_{i=1}^{n} \alpha_i} g(t_1, \ldots, t_n). \tag{6}$$

THEOREM 1 (FOT SIMILARITY [24]) The relation $\sim^{\mathcal{T}}$ defined by Definition 5 is a similarity relation on the set of $\mathcal{FOT}$s $\mathcal{T}_{\Sigma,\mathcal{V}}$.

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 1.

[15] Assuming of course that the writing direction goes from left to right, as in English.

DEFINITION 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 6, the more specific term t_1 is then called a fuzzy instance of term t_2 realized with substitution 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 **SUBST** $_\mathcal{T}$ as a corollary of Theorem 1. It is formally equivalent to the relation defined in [24].[16]

DEFINITION 7 (FUZZY "MORE GENERAL" ORDERING ON $\mathcal{FOT}$ SUBSTITUTIONS) If σ_1 and σ_2 are two substitutions in **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 **SUBST** $_\mathcal{T}$ can also be verified to be a similarity as a corollary of Theorem 1.[17]

DEFINITION 8 ($\mathcal{FOT}$ SUBSTITUTION SIMILARITY) Given an approximation degree α in $[0,1]$, two substitutions σ and θ in **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 6 of fuzzy $\mathcal{FOT}$ subsumption, it comes as a fact that:

LEMMA 2 For any two substitutions σ and θ in **SUBST** $_\mathcal{T}$ and approximation degree α in $[0,1]$, $\sigma \preceq_\alpha \theta$ iff $\sigma \sim_\alpha \theta\delta$ for some substitution δ.

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 3 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 $t_1 \preceq_{\alpha\wedge\beta} t_3$.

COROLLARY 2 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 $\sigma_1 \preceq_{\alpha\wedge\beta} \sigma_3$.

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 $t_1\sigma \sim_\alpha t_2\sigma$.

In Figure 1, we provide a set of declarative rewrite rules for fuzzy unfication equivalent to Sessa's case-based "weak unification algorithm" [24]. 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 [24], Page 413.) Later, we will relax this and allow functors of different arities to be similar.

The rules of Figure 1 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 1 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

[16] *Op. cit.*, Page 410, Definition 6.2

[17] A equivalent definition is given in [16].

WEAK TERM DECOMPOSITION

$$[f \sim_\beta g;\ n \geq 0]$$

$$\frac{(E \cup \{\, f(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_n) \,\})_\alpha}{(E \cup \{\, s_1 \doteq t_1, \ldots, s_n \doteq t_n \,\})_{\alpha \wedge \beta}}$$

VARIABLE ERASURE

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

VARIABLE ELIMINATION

$$[X \notin \mathbf{var}(t);\ X \text{ occurs in } E]$$

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

EQUATION ORIENTATION

$$[t \notin \mathcal{V}]$$

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

Figure 1: Normalization rules corresponding to Maria Sessa's "weak unification"

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 $s\sigma \sim_\alpha t\sigma$.

In [24],[18] 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 10.[19]

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 [18].

Still, Sessa's approach 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 $\approx\colon\ \Sigma \times \Sigma \to [0, 1]$ 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.

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

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

[19] Note that in [24], 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 Programming in [7].

- 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 $\beta \in [0, 1]$:

$$\alpha \leq \beta \implies \mu^{\alpha}_{fg} \subseteq \mu^{\beta}_{fg} \quad \text{(as sets of pairs)}; \tag{7}$$

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

$$\mu^{\alpha}_{ff} = \mathbb{1}_{\{1,\ldots,n\}}; \tag{8}$$

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

$$\mu^{\alpha}_{fg} \circ \mu^{\alpha}_{gf} = \mathbb{1}_{\{1,\ldots,n\}}; \tag{9}$$

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

$$\mu^{\alpha}_{fh} = \mu^{\alpha}_{gh} \circ \mu^{\alpha}_{fg}. \tag{10}$$

Note that Condition (10) 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,[20] and in the normalization rules.[21] 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 (9). 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 $\mathbf{arity}(f) \leq \mathbf{arity}(g)$, we shall sometimes simply write $f \approx^{p}_{\alpha} g$, with p standing for the injective argument realignment mapping μ_{fg}.

DEFINITION 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:

$$f(s_1, \ldots, s_m) \approx^{\mathcal{T}}_{\alpha \wedge \bigwedge_{i=1}^{m} \alpha_i} g(t_1, \ldots, t_n). \tag{11}$$

[20] *Cf.*, Definition 9 below.
[21] *Cf.*, Figure 2 below, Rule **FUZZY EQUATION ORIENTATION**.

THEOREM 2 (NON-ALIGNED $\mathcal{FOT}$ SIMILARITY [5]) The fuzzy relation $\approx^{\mathcal{T}}$ on the set $\mathcal{T}$ of $\mathcal{FOT}$s specified in Definition 9 is a similarity.

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 6–8 and Lemmas 2–3, as well as Corollary 2, 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 set of unification rules ([13] and [20]),[22] 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 is the only one that may possibly alter (decrease) the equation set's 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 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 10 (FUZZY UNIFICATION RULE CORRECTNESS) A fuzzy unification rule that transforms a pair E_α consisting of a set of equations E and a prior approximation degree α, into a pair E'_β consisting of a set of equations E' and 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 correctness, 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 is so at a possibly lesser posterior approximation degree.

THEOREM 3 (FUZZY NON-ALIGNED $\mathcal{FOT}$ UNIFICATION [5]) The fuzzy unification rules of Figure 1 where Rule "**WEAK TERM DECOMPOSITION**" is replaced by the rules of Figure 2 are correct.

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;

[22] This ruleset is recalled in [5].

FUZZY NON-ALIGNED-ARGUMENT TERM DECOMPOSITION

$$\left[0 \leq m \leq n;\ f \approx^{p}_{\beta} g\right]$$

$$\frac{(E \cup \{\, f(s_1, \ldots, s_m) \doteq g(t_1, \ldots, t_n) \,\})_{\alpha}}{(E \cup \{\, s_1 \doteq t_{p(1)}, \ldots, s_m \doteq t_{p(m)} \,\})_{\alpha \wedge \beta}}$$

FUZZY EQUATION ORIENTATION

$$[0 \leq m < n]$$

$$\frac{(E \cup \{\, g(t_1, \ldots, t_n) \doteq f(s_1, \ldots, s_m) \,\})_{\alpha}}{(E \cup \{\, f(s_1, \ldots, s_m) \doteq g(t_1, \ldots, t_n) \,\})_{\alpha}}$$

Figure 2: Fuzzy $\mathcal{FOT}$ unification's non-aligned decomposition and orientation rules

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 [26].[23]

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

A.3 Prolog's LEGO SOS Semantics

A LEGO-state machine is a syntax-oriented state-transformation automaton. A state of this machine is a quadruple of the form $\langle \mathtt{L}, \mathtt{E}, \mathtt{G}, \mathtt{O} \rangle$ that stands for Literal, Environment, Goals, Or-continuation. It specifies a formal operational semantics for Prolog in the same manner as the SECD machine is defined to stand for Store, Code, Environment, Dump to designate a state 4-tuple $\langle \mathtt{S}, \mathtt{E}, \mathtt{C}, \mathtt{D} \rangle$ for a state-transformation machine for the call-by-value λ-calculus [19, 9]. This 4-tuple state is the unit being transformed at each step of computation of deterministic Plotkin-style formal Structure-Oriented Semantics (SOS) [22] we define for Prolog with the syntactic domains and state-transformation normalization rules that follow.

Notation A syntactic domain is specified as a set of elements represented either as a given atomic domain or as a meta-expression of syntactic domains. Given two syntactic domain expressions E and E', we shall write $E \times E'$ the cartesian product of E and E'; *i.e.*, the set of all pairs $\langle e, e' \rangle$ where $e \in E$ and $e' \in E'$. We shall write $E + E'$ the disjoint union of E and E'; *i.e.*, the set $E \cup E'$ where $E \cap E' = \emptyset$. Given a syntactic domain expression E, the domain expression $\overline{E}$ denotes the complement of the set denoted by the domain expression E (*i.e.*, any syntactically well-formed domain expression that is syntactically incompatible with E). The domain expression $E^* \stackrel{\text{def}}{=} \Sigma_{n \geq 0} E^n$ denotes the set of possibly empty sequences of domain

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

expressions of E. That is, an element of domain E^* is either the empty tuple $\langle\rangle$, or a tuple $\langle e_1, e_2\rangle$ where $e_1 \in E$ and $e_2 \in E^*$. We also write $\langle e\rangle$ rather than $\langle e, \langle\rangle\rangle$ and $\langle e_1, e_2, e_3\rangle$ instead of $\langle e_1, \langle e_2, e_3\rangle\rangle$. The domain expression $E^+ \stackrel{\text{def}}{=} \Sigma_{n\geq 1} E^n$ denotes the set of non-empty sequences of domain expressions of E; in other words, $E^* = \{\langle\rangle\} + E^+$. Finally, we shall write $e : E$ to state that the syntactic term expression e has one of the form required for elements of the syntactic domain expression E.

Terms and Literals

Basic Syntactic Domains

V	Variables
Φ_n	Function symbols of arity $n (n \geq 0)$
Π_n	Predicate symbols of arity $n (n \geq 0)$

Derived Syntactic Domains

Atomic symbols	A	$\stackrel{\text{def}}{=}$	$V + \Phi_0 + \Pi_0$
Function Symbols	Φ	$\stackrel{\text{def}}{=}$	$\sum_{n\geq 0} \Phi_n$
Predicate Symbols	Π	$\stackrel{\text{def}}{=}$	$\sum_{n\geq 0} \Pi_n$
Functional Terms	T	$\stackrel{\text{def}}{=}$	$\{\perp\} + V + \sum_{n\geq 0} (\Phi_n \times T^n)$
Literals	L	$\stackrel{\text{def}}{=}$	$\sum_{n\geq 0} (\Pi_n \times T^n)$

First-Order Unification

Syntactic Domains

Unificands	U	$\stackrel{\text{def}}{=}$	$(T + \Phi + \Pi)^+$
Equations	EQ	$\stackrel{\text{def}}{=}$	$U \times U$
Environments	E	$\stackrel{\text{def}}{=}$	$V \to T$
U-Continuations	UC	$\stackrel{\text{def}}{=}$	EQ^*
U-States	US	$\stackrel{\text{def}}{=}$	$EQ \times E \times UC$

Transition Rules

$$\frac{\langle t_1 : T + \{\bot\}, t_2\rangle, e, \langle\rangle}{\langle t_1, t_2\rangle, e, \langle\rangle}\ \{t_1{=}{=}t_2\} \qquad (\textit{done})$$

$$\frac{\langle t_1, t_2\rangle, e, \langle\langle u_1, u_2\rangle, uc\rangle}{\langle u_1, u_2\rangle, e, uc}\ \{t_1{=}{=}t_2\} \qquad (\textit{more})$$

$$\frac{\langle x : V, u : \overline{V}\rangle, e, uc}{\langle e^*(x), u\rangle, e[x \to u], uc} \qquad (\textit{lvar})$$

$$\frac{\langle u : \overline{V}, x : V\rangle, e, uc}{\langle x, u\rangle, e, uc} \qquad (\textit{rvar})$$

$$\frac{\langle x : V, y : V\rangle, e, uc}{\langle e^*(x), e^*(y)\rangle, e[x \to e^*(y)], uc}\ \{x{=}/{=}y\} \qquad (\textit{bvar})$$

$$\frac{\langle x_1 : \langle u_1, v_1\rangle, x_2 : \langle u_2, v_2\rangle\rangle, e, uc}{\langle u_1, u_2\rangle, e, \langle\langle v_1, v_2\rangle, uc\rangle}\ \{x_1{=}/{=}x_2\} \qquad (\textit{push})$$

$$\frac{\langle f_1 : \Phi_{n_1} + \Pi_{n_1}, f_2 : \Phi_{n_2} + \Pi_{n_2}\rangle, e, uc}{\langle\bot, \bot\rangle, \square, \langle\rangle}\ \{f_1{=}/{=}f_2 \text{ or } n_1{=}/{=}n_2\} \qquad (\textit{fail1})$$

$$\frac{\langle\langle u, v\rangle, a : \Phi + \Pi\rangle, e, uc}{\langle\bot, \bot\rangle, \square, \langle\rangle} \qquad (\textit{fail2})$$

$$\frac{\langle a : \Phi + \Pi, \langle u, v\rangle\rangle, e, uc}{\langle\bot, \bot\rangle, \square, \langle\rangle} \qquad (\textit{fail3})$$

Prolog (without cut)

Syntactic Domains

Programs	$: P$	$\stackrel{\text{def}}{=}$	D^*
Definitions	$: D$	$\stackrel{\text{def}}{=}$	C^*
Clauses	$: C$	$\stackrel{\text{def}}{=}$	L :- B
Bodies	$: B$	$\stackrel{\text{def}}{=}$	$\{\mathfrak{true}\} + L \times B$
Goals (And-Continuations)	$: G$	$\stackrel{\text{def}}{=}$	B^*
Or-Continuations	$: O$	$\stackrel{\text{def}}{=}$	$D \times E \times L \times O$
P-States	$: PS$	$\stackrel{\text{def}}{=}$	$L \times E \times G \times O$

Transition Rules

$$\frac{}{\mathfrak{true}, \mathbf{1}_V, \mathit{goal}, \langle\rangle} \qquad (\mathit{initialize})$$

$$\frac{\mathfrak{true}, e, \langle\rangle, o}{\mathfrak{true}, e, \langle \mathit{user}\rangle, o} \qquad (\mathit{success})$$

$$\frac{\mathfrak{fail}, e, g, \langle\rangle}{\mathfrak{fail}, \Box, g, \langle\rangle} \qquad (\mathit{failure})$$

$$\frac{\mathfrak{fail}, e, \langle r, a\rangle, \langle d : \overline{\langle\rangle}, \_, \_, \_\rangle}{\mathit{next}, e, a, \langle d, \_, \_, \_\rangle} \qquad (\mathit{try\_again})$$

$$\frac{\mathfrak{fail}, e, \langle r, \langle r', a\rangle\rangle, \langle\langle\rangle, e', g, o\rangle}{\mathit{next}, e', a, o} \qquad (\mathit{backtrack})$$

$$\frac{\mathit{next}, e, a, \langle\langle\langle h \ :\!- \ l, b\rangle, d\rangle, e', g, o\rangle}{l, e'', \langle b, a\rangle, \langle d, e', g, o\rangle} \ \{\langle \_, e'', \_\rangle = \mathit{unify}(\langle g, h, e'\rangle)\} \qquad (\mathit{next\_try})$$

$$\frac{\mathfrak{true}, e, \langle\langle h, r\rangle, a\rangle, o}{h, e, \langle r, a\rangle, o} \qquad (\mathfrak{true})$$

$$\frac{g : \overline{\{\mathfrak{true}, \mathfrak{fail}, \mathit{next}\}}, e, a, o}{\mathit{next}, e, a, \langle d, e, g, o\rangle} \ \{d = \mathit{definition}(g)\} \qquad (\mathit{prove})$$

References

[1] ACHS, A., AND KISS, A. Fuzzy extension of Datalog. *Acta Cybernetica 12*, 2 (1995), 153–166. [Available online[24]].

[2] AHO, A., HOPCROFT, J., AND ULLMANN, J. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.

[3] AÏT-KACI, H., AND PASI, G. 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[25]].

[4] AÏT-KACI, H., AND PASI, G. *Fuzzy Lattice–Theoretic Operations over Data and Knowledge Structures*. unpublished manuscript draft, 2019. [Available online[26]].

[5] AÏT-KACI, H., AND PASI, G. Fuzzy lattice operations on first-order terms over signatures with similar constructors: A constraint-based approach. *Fuzzy Sets and Systems* (to appear in 2019). [Available online[27]].

[6] ALSINET, T., GODO, L., AND SANDRI, S. Two formalisms of extended possibilistic logic programming with context-dependent fuzzy unification: a comparative description. *Electronic Notes in Theoretical Computer Science 66*, 5 (2002), 21 pages. [Available online[28]].

[24] http://people.inf.elte.hu/kiss/14abea/Achs_1995_ActaCybernetica.pdf
[25] http://www.hassan-ait-kaci.net/pdf/fuzfotlats-lopstr2017-slides.pdf
[26] http://hassan-ait-kaci.net/pdf/fuzzy-fotosf-lattices-book-draft.pdf
[27] http://hassan-ait-kaci.net/pdf/fuzfotlat-preprint.pdf
[28] http://repositori.udl.cat/bitstream/handle/10459.1/57984/001858.pdf

[7] APT, K. R. Logic programming. In *Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed. Elsevier, 1990, pp. 492–574.

[8] ARCELLI, F., AND FORMATO, F. A fuzzy logic programming language. In *Programmazione Logica e Prolog* (1997), L. Console, E. Lamma, P. Mello, and M. Milano, Eds., UTET Università, pp. 319–332. [Available online[29]].

[9] CARDELLI, L. The functional abstract machine. Technical Report TR-107, Bell Laboratories, Murray Hill, NJ (USA), May 1983.

[10] CORNEJO, M. E., MORENO, J. M., AND RUBIO-MANZANO, C. Towards a full fuzzy unification in the Bousi∼Prolog system. In *Proceedings of the International Conference on Fuzzy Systems (FUZZ-IEEE 2018)* (July 2018), F. Gomide, Ed. [Available online[30]].

[11] DUBOIS, D., AND PRADE, H. *Fuzzy Sets and Systems: Theory and Applications*, vol. 144 of *Mathematics in Science and Engineering, Edited by William F. Ames, Georgia Institute of Technology*. Academic Press, 180. [Available online[31]].

[12] GUERINI, S., AND MASINI, A. Proofs, tests and continuation passing style. *ACM Transactions on Computational Logic (TOCL) 10*, 2, article 12 (February 2009), 12:1–12:34. [Available online[32]].

[13] HERBRAND, J. *Recherches sur la théorie de la démonstration*. PhD thesis, Faculté des sciences de l'université de Paris, Paris (France), 1930. [Available online[33]]; English translation in [14].

[14] HERBRAND, J. *Logical Writings*. Harvard University Press, Cambridge, MA, 1971. Edited by Warren D. Goldfarb.

[15] IRANZO, P. J., AND MANZANO, C. R. 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)* (Barcelona, Spain, July 18–23, 2010), IEEE, pp. 658–665. [Available online[34]].

[16] IRANZO, P. J., AND MANZANO, C. R. A sound and complete semantics for a similarity-based logic programming language. *Fuzzy Sets and Systems 317* (2017), 1–26.

[17] IRANZO, P. J., MORENO, G., PENABAD, J., AND VÁZQUEZ, C. A fuzzy logic programming environment for managing similarity and truth degrees. In *Proceedings XIV Jornadas sobre Programación y Lenguajes (PROLE 2014)* (January 2015), S. Escobar, Ed., Electronic Proceedings in Theoretical Computer Science, EPTCS 173, pp. 71–86. [Available online[35]].

[18] LACOSTE-JULIEN, S., PALLA, K., DAVIES, A., KASNECI, G., GRAEPEL, T., AND GHAHRAMANI, Z. SiGMa: Simple greedy matching for aligning large knowledge bases. In *Proceedings of the 19th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2013—Chicago, IL, USA)* (New York, NY (USA), August 11–14, 2013), I. S. Dhillon, Y. Koren, R. Ghani, T. E. Senator, P. Bradley, R. Parekh, J. He, R. L. Grossman, and R. Uthurusamy, Eds., Association for Computing Machinery, ACM, pp. 572–580. [Available online[36]]; see also [Available online[37]].

[19] LANDIN, P. The mechanical evaluation of expressions. *Computer Journal 6* (1963), 308–320.

[20] MARTELLI, A., AND MONTANARI, U. An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems 4*, 2 (April 1982), 258–282. [Available online[38]].

[21] PLOTKIN, G. D. A structural approach to operational semantics. Tech. Rep. DAIMI FN-19, University of Århus, Århus, Denmark, 1981. [Available online[39]].

[22] PLOTKIN, G. D. A structural approach to operational semantics. *Journal of Logic and Algebraic Programming 60–61* (January 30, 2004), 17–139. [Available online[40]]—*N.B.*: Published version of [21].

[29] http://www.programmazionelogica.it/wp-content/uploads/1997/06/319_Fontana1.pdf
[30] https://[...]Towards_a_Full_Fuzzy_Unification_in_the_Bousi_Prolog_system
[31] ftp://[...]/Fuzzy%20Sets%20And%20Systems%20Theory%20And%20Applications.pdf
[32] https://dl.acm.org/citation.cfm?id=1462184
[33] http://archive.numdam.org/article/THESE_1930__110__1_0.pdf
[34] https://pdfs.semanticscholar.org/f084/ae4f51a755c2748ae964b036976a68a18c58.pdf
[35] https://arxiv.org/pdf/1501.02034.pdf
[36] http://snap.stanford.edu/social2012/papers/lacostejulien-palla-etal.pdf
[37] https://arxiv.org/pdf/1207.4525.pdf
[38] http://moscova.inria.fr/~levy/courses/X/IF/03/pi/levy2/martelli-montanari.pdf
[39] http://citeseer.ist.psu.edu/673965.html
[40] http://homepages.inf.ed.ac.uk/gdp/publications/sos_jlap.pdf

[23] ROBINSON, J. A. A machine-oriented logic based on the resolution principle. *Journal of the ACM 12* (January 1965), 23–41. [Available online[41]].

[24] SESSA, M. I. Approximate reasoning by similarity-based SLD resolution. *Theoretical Computer Science 275* (2002), 389–426. [Available online[42]].

[25] VAUCHERET, C., GUADARRAMA, S., AND MUÑOZ-HERNÁNDEZ, S. Fuzzy Prolog: a simple general implementation using CLP($\mathbb{R}$). In *Proceedings of the 9^{th} International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2002)* (Tbilisi, Georgia, October 14–18, 2002), M. Baaz and A. Voronkov, Eds., LNCS 2514, Springer, pp. 450–464. [Available online[43]].

[26] WAYNE, K. Union-find. Tutorial lecture slides based on book "Algorithm Design" by Jon Kleinberg and Éva Tardos (Addison-Wesley, 2015). [Available online[44]].

[41] https://www.academia.edu/[...]/A_[...]_Logic_Based_on_the_Resolution_Principle
[42] http://www.sciencedirect.com/science/article/pii/S0304397501001888
[43] https://cliplab.org/papers/fuzzy-lpar02_bitmap.pdf
[44] https://www.cs.princeton.edu/ wayne/kleinberg-tardos/pdf/UnionFind.pdf

DRAFT—PLEASE DO NOT DISTRIBUTE