Towards Fuzzy Linguistic Logic Programming

Clemente Rubio-Manzano*1 and Pascual Julián-Iránzo2

1Dep. of Information Systems, Univ. of the Bio-Bio, Chile
2Dep. of Technologies and Information Systems, Univ. of Castilla-La Mancha, Spain

Email: C. Rubio-Manzano* - clrubio@ubiobio.cl; P.Julián-Iránzo - PascualJulian@uclm.es;

Abstract Knowledge representation is one of the central concepts in Artificial Intelligence. It is very common that knowledge about a field is expressed in natural language (English, Spanish, etc). Therefore, most of the times, knowledge representation using a logic programming language derives into a translation problem. This translation consists in the formalization of the statements, belonging to the knowledge level, which are converted into formulas of the so called symbolic level. Knowledge may be imprecise or vague and, in order to deal with vagueness using declarative techniques, fuzzy logic programming amalgamates classical logic programming and fuzzy logic. Fuzzy logic programming has mainly led to programming languages that use annotations (i.e., truth degrees, certainty factors or degrees of confidence) to represent vagueness. But vagueness is a linguistic phenomenon which is implicit in the statements of the knowledge level. Hence, the natural connection existing between these two levels is broken when annotations are employed, since they introduce weights in a symbolic level which are not present in the knowledge level and converts knowledge representation in a more complex, counterintuitive task.

In order to overcome this problem, we propose a fuzzy linguistic logic framework which allows the treatment of imprecision through (crisp or fuzzy) linguistic resources. This framework makes a clean separation between precise knowledge and vague knowledge. In this paper, we argue that this separation is more declarative than the one dispensed by the approach based on annotations, and can be beneficial for modeling a problem.

Submitted: 10/03/14. Accepted and Published: 24/11/14.
1 Introduction and motivation

The Webster’s Dictionary affirms that knowledge is “the fact or condition of knowing something with familiarity gained through experience or association”. Intelligence requires knowledge and therefore knowledge and knowledge representation (KR) have been important issues in Artificial Intelligence (AI) from the first times. In 1959, McCarthy, one of the founders of AI, consistently advocated a research methodology that uses logical techniques to formalize reasoning problems [7].

Over the ensuing decades, there was much research on knowledge representation in formal logic and on inference algorithms to manipulate that knowledge [4,41,42,60]. Logic strengthens the declarative approach to knowledge representation, where knowledge is represented by means of sentences of a formal language. The arguments for a declarative knowledge representation are: it allows the explicit representation of knowledge, it is modular and it supports modification far more easily than procedural knowledge. On the other hand, although various notations for knowledge representation have been designed separately (semantic nets, frames, objects, etc.), jointly with their specific notions of entailment, and have been used for many different purposes, each one can usefully be written down in logic [60]. Therefore, the logical approach (intending “logic” in a broad sense) have gained credence among the AI community\(^1\).

Classical logic has limitations to represent uncertain knowledge and reasoning with it. In order to overcome this problem AI researches have followed diverse approaches depending on the nature of uncertainty. Probability theory has been used for random uncertainty and to express degrees of belief [43]. When uncertainty is a linguistic phenomenon, and comes from the fact that the concepts are vague and have not well defined boundaries or they are permeated by subjectivity, then Fuzzy Logic is the best option [61]. In this work we concentrate on this last kind of uncertainty.

There is a vast number of proposals that study the integration of Fuzzy Logic [61] into Logic Programming [28,46] with the purpose of dealing with vagueness. The first works on this field [19,20,34] incorporated Lee’s fuzzy resolution [30] in Prolog systems. At the same time, other studies on the implementation of fuzzy sets in Prolog came on the scene, thus, originating the deliberation of whether it is necessary an automatic, implicit treatment of vagueness [26,38].

On the other hand, the consolidation of fuzzy logic motivates the emergence of richer, complete proposals giving origin to the field of Fuzzy Logic Programming (FLP). These proposals can be grouped together into two great lines of work: (a) Extensions of Resolution and New Inference Mechanisms (1,}

\(^1\)Of course, there is a community of researches which believes that human intelligence is too complex to be represented using mathematical logic and has devised non logical methods to automate reasoning.
programs are fuzzy theories with facts, rules and goals annotated by weights; the inference mechanism is modified so that operations on those weights can be performed in order to propagate them; in these approaches, classical unification remains intact; (b) Extensions of Unification \cite{15,16,33,51,54,57,58}: they lead to non-annotated languages which are based on fuzzy relations (weights are specified separately in the definition of those fuzzy relations); the resolution procedure is, essentially, let unaltered and only the unification algorithm is modified, being replaced by a fuzzy one which is in charge of the propagation of the weights during the computation process.

These two lines of work lead to completely different ways of representing vagueness. For example, the statements “john is young” and “a young person is fast” can be represented in plane first order logic as the sentences:

\begin{verbatim}
young(john).
fast(X) :- young(X).
\end{verbatim}

But, “young” and “fast” are vague concepts. First note that vagueness is implicit in natural language and, thus, in knowledge. In order to represent the implicit vagueness of a statement, it must be made explicit in the symbolic level of representation. Annotated languages annotate the facts and rules of the program with weights in order to represent vagueness, this might not be adequate in a linguistic context because in the regular speech of a person there are not weights or degrees of truth, nobody uses to say “john is young with 0.8” or “if a person is young, then he is rapid with 0.5” and hence the translation problem becomes a more complex process. On the other hand, non-annotated languages consider words like “young” and “fast” as linguistic labels which are specified separately from the facts and rules of the program. These linguistic labels are defined through fuzzy relations which put them in association with other appropriate linguistic labels. Then the question about if programs must be annotated in order to treat vagueness gains importance. The answer to this question may be a determinant in terms of an adequate knowledge representation.

The objective of this work is to confirm our hypothesis that non-annotated FLP languages are more adequate to the specific task of representing vagueness. As we shall try to establish along this paper, the reasons are that this kind of languages do not break the natural connection between the knowledge level and the symbolic level and, because they define vagueness separately from the program rules, they produce a more declarative representation of knowledge.

Before ending this introductory section note that, as a position paper we present an arguable opinion about an issue. Our goal is to convince the audience that our opinions are valid and worth listening to. Although we do not present a completed research work, we try to support our claims with reasonable evidences.
Also note that our intention is not to write a complete survey of all existing FLP frameworks, because the field is quite dense of publications. Moreover, we are interested in the expressive power of these programming languages to deal with vagueness, imprecision and the closeness of a concept in a linguistic way.

We are aware of the existence of other frameworks with similar objectives like, for instance, Quantitative Logic Programming [55] and its derivatives [29, 47]. More recently, the adoption of description logic for modeling the semantic web has led up to the proposal of fuzzy logic descriptive languages which employ annotation domains for the treatment of vagueness [53]. A good survey on this complementary topic is [52], which is focused on managing uncertainty and vagueness in description logics, and description logic programs and it treats with logic programs to a lesser extent. These last aforementioned approaches are not considered in this paper, which is simply restricted to the specific field of FLP.

2 Fuzzy Logic Programming

Fuzzy Logic Programming (FLP) integrates Logic Programming and Fuzzy Logic with the purpose of managing vagueness and/or imprecision by means of declarative techniques. Here, we briefly describe the two approaches to FLP aforementioned in the last section. Our description is mainly focused on syntactic aspects, as we are interested on expressiveness, and it summarizes their main features instead of describing a concrete language.

2.1 FLP extending Resolution.
By simplicity, we describe here a reduced FLP framework that extends the classical resolution rule\(^2\). In this framework, programs are annotated. Then, the representation of vagueness is through truth degrees associate to the facts and the rules of a program.

More precisely, a fuzzy logic program would be composed of rules of the form:

\[
\begin{equation}
C \equiv A \leftarrow [:: \alpha] \langle B_1, :: \beta_1 \rangle, \ldots, \langle B_n, :: \beta_n \rangle
\end{equation}
\]

where \(A\) is an atom (that is, an atomic formula), called the head, and \(B_1, \ldots, B_n\) is a conjunction of atoms, called the body, \(\alpha\) is a truth degree\(^3\) annotating the rule \(C\) and \(\beta_1, \ldots, \beta_n\) are truth degrees annotating the body atoms.

Annotations of the body of the rule \(C\) are constraining the applicability of the whole rule (or constraining the answer to a goal). More precisely, if for each

\(^2\)In order to obtain a detailed formalization of more elaborated or specific proposals see [59] or [35].

\(^3\)In Fuzzy Logic the concept of truth is a matter of degree. Normally, truth values are taken from the \([0, 1]\) real interval, which is a totally ordered set. However, in other proposals, intervals of truth or values of a more generic lattice on a partially ordered set are allowed.
1 \leq i \leq n, B_i \text{ is provable with truth degree } \alpha_i \geq \beta_i, \text{ the rule } \mathcal{C} \text{ is applicable and the atom } A \text{ can be inferred with a truth degree } \gamma = \alpha \bigtriangleup^{-1}(\alpha_1 \bigtriangleup^{n}(\alpha_i)), \text{ where } \bigtriangleup^{-} \text{ and } \bigtriangleup \text{ are fuzzy connectives (usually t-norms).}

This, informally described, operational semantics can be seen as a kind of residuated logic. Note that an annotated atom, \( B_i[: 0] \), does not introduce any applicability restriction. Annotated atoms of the form \( B_i[: 0] \) are denoted as \( B_i \). Similarly, a rule \( A \leftarrow [: 1] \langle B_1, [: \beta_1] \rangle, \ldots, \langle B_n, [: \beta_n] \rangle \) is denoted by \( A \leftarrow \langle B_1, [: \beta_1] \rangle, \ldots, \langle B_n, [: \beta_n] \rangle \).

A fact is a rule without body. A fact of the form \( A \leftarrow 1 \) is simply denoted by \( A \). On the contrary, a rule without head is said a goal. A fuzzy goal can be of the form \( \leftarrow [: \alpha] \langle B_1, [: \beta_1] \rangle, \ldots, \langle B_n, [: \beta_n] \rangle \), where \( \alpha \) can be a constant, that indicates the maximum truth degree that can be attained during a computation, or a variable, that will be bounded to the final computed truth degree.

Now, in this framework, when a goal is launched, in case of success, a final truth degree is given besides the traditional computed substitution.

Example 1 Suppose we want to model in this FLP framework a fragment of a database containing information about films along with film critic opinions summarized by the following set of labels: \{ bad, so_so, good, very_good \}. A potential program would be the following (using a syntax similar to Prolog):

```
%% FACTS
film(dracula,so_so).
film(the_godfather,very_good).
film(the_lords_of_the_rings,good).

%% RULES HANDLING VAGUENESS
recommend(X):- film(X,very_good).
recommend(X):- [:0.8] film(X,good).
recommend(X):- [:0.5] film(X,so_so).
```

Now, using an operational mechanism like the one explained in this subsection and assuming that we employ the minimum t-norm, acting as the conjunction connective, if we would launch the goal “?-[:F] recommend(X)”, the response of the system would be: “X=the_godfather” with “F=1.0”; “X=the_lords_of_the_rings” with “F=0.8” and “X=dracula” with “F=0.5”. The first answer is obtained because, once the goal “?-[:F] recommend(X)” is launched, the first rule defining the predicate “recommend” is applicable with truth degree “1”. Then, the subgoal “?-[:F.1] film(X.1, very_good)” is launched, and the syntactic unification with the fact “film(the_godfather, very_good)” is performed. Hence, the partial answer “X.1=the_godfather” with degree “F.1=1.0” is obtained, leading to the final computer answer. The rest of computer answers are obtained similarly.

Furthermore, it has been studied the incorporation of fuzzy sets into these frameworks. The most common solution has been to use a first order fuzzy predicate [34] acting as membership function of a fuzzy subset (many current systems, such as the fuzzy module of Ciao-Prolog [18], the fuzzy extensions of the descriptive logic [53] or FLOPER [37], have adhere to this idea). This notion is illustrated with the following example.
Example 2  A fuzzy subset “young” over the universe of discourse (years) $U_{\text{age}} = [0, 100]$ can be represented by a fuzzy first-order predicate “young($x$)” which is defined as a membership function of that fuzzy subset young over $U_{\text{age}}$. For instance:

$\text{young}(0) = 0$. $\text{young}(21) = 1$. $\text{young}(29) = 0.75$. $\text{young}(31) = 0.25$. $\text{young}(40) = 0$.

This mechanism may be seen as an unnatural way to deal with fuzzy subsets in the framework of a FLP language as “young” is not handled as a linguistic term able to be related with other linguistic labels (but as a mere function or predicate symbol).

First note that, in these annotated languages, it is not easy to define linguistic modifiers. They need to use ad-hoc techniques. For instance, in [37] “very” is a function (implemented as a built-in operator) which is applied to atomic formulas instead of linguistic terms (e.g., “very(young(john))”), what requires a specific, less general treatment. On the contrary, a FLP extending unification allows the treatment of the symbol “very” as a linguistic modifier which is able to produce the linguistic term “very#young”. This new linguistic term is added to the signature and can be used in a standard way in combination with the other symbols of the first order language.

On the other hand, note that, to establish a relationship between the terms “young” and “very young”, in those annotated FLP languages, such a relation should be programmed in an explicit way. In this case, by computing the degree of similarity of both linguistic terms through the degree of membership of the values of the domain with regard to the fuzzy subsets associated to each linguistic term. In contrast, in a FLP language extending unification this is done in a transparent, automatic way (See later).

Note however that, within these frameworks, there have been proposals that can be considered as hybrid systems, since they implement a semantic unification between constants, what provides a greater functionality to these languages (see, for instance, [6]).

2.2 FLP extending unification

The objective of frameworks that extend unification is to obtain a more flexible question answering mechanism [21, 51]. At least, there are two ways to achieve this goal: Semantic Unification [2, 5], where the meaning of each linguistic constant term is assigned, for example, by means of fuzzy subsets associated to them and the unification of that constants is performed on the basis of their meaning (for example, through a matching procedure); and Weak Unification [51] which is a fuzzy extension of the classic unification algorithm. However, as it was shown in [22], semantic unification can be reduced to weak unification. Therefore, in this section, we are focused on the last approach.
Roughly speaking, the *weak unification* algorithm mimics the classical unification algorithm but replacing the syntactic equality relation, with a similarity [51] or proximity [23, 25] relation on a syntactic domain. By simplicity, in this section we concentrate on similarities, which are a special case of proximities.

A *similarity relation* on a set $U$ is a fuzzy binary relation on $U \times U$, that is, a mapping $R: U \times U \rightarrow [0, 1]$, holding the following properties: reflexive; symmetric and transitive. In this context, “transitive” means that $R(x, z) \geq R(x, y) \triangle R(y, z)$ for any $x, y, z \in U$; where the operator ‘$\triangle$’ is an arbitrary t-norm. We are specially interested in similarity relations on a syntactic domain where $\triangle = \land$ (that is, it is the minimum of two elements). A similarity relation $R$ on the alphabet of a first order language can be extended to terms by structural induction:

1. $R(x, x) = 1$;

2. Let $f$ and $g$ be two $n$-ary function symbols and let $t_1, \ldots, t_n, s_1, \ldots, s_n$ be terms. $R(f(t_1, \ldots, t_n), g(s_1, \ldots, s_n)) = R(f, g) \land (\land_{i=1}^{n} R(t_i, s_i))$.

Otherwise, the approximation degree of two terms is zero. The extension to atoms can be done in a completely analogous form.

Hence, following [51], by introducing a similarity relation $R$ on the alphabet of a first order language, it is possible to treat as indistinguishable two syntactic symbols which are related by $R$ with a certain degree greater than zero. Now the unification algorithm does not terminate when two terms differ in some symbol if these symbols are similar. Furthermore, the resolution procedure must be adapted in order to propagate and combine the approximation degrees obtained during the weak unification process at each resolution step.

A fuzzy program, in this framework, is built as a set of Horn clauses and a similarity relation between the symbols of a first order alphabet. That is, formulas of the form $A \leftarrow B_1, \ldots, B_n$, where $A$ is an atom, called the head, and $B_1, \ldots, B_n$ denotes a conjunction of atoms, called the body.

A goal is any body. On the other hand, an entry $R(a, b) = \alpha$, defining a similarity relation $R$, will be denoted by a *proximity equation* as $a \sim b = \alpha$.

The operational semantics of these programming languages are based on a generalization of the SLD resolution principle that we call *Weak SLD resolution* (WLSD-resolution). A WLSD-resolution step can be defined as follows. Let $\Pi$ be a program and $R$ be a similarity relation. Given a goal $G \equiv \leftarrow A', Q'$ if exists a standardized clause in $\Pi, C \equiv (A \leftarrow Q)$, such that the atoms $A'$ and $A$ weakly unify with a weak most general unifier $\sigma$ and $\beta = R(A', A) > 0$, then $G' \equiv \leftarrow (Q, Q')\sigma$, with an approximation degree $\alpha$, is the weak resolvent of $G$ and $C$.

We denote a WLSD-resolution step as $G \xrightarrow{[C, \sigma, \beta]}_{\text{WLSD}} G'$.

A WLSD-derivation for $\Pi \cup \{G_0\}$ and $R$ is a sequence of WLSD-resolution steps: $G_0 \xrightarrow{[C_1, \theta_1, \beta_1]}_{\text{WLSD}} \cdots \xrightarrow{[C_n, \theta_n, \beta_n]}_{\text{WLSD}} G_n$. A WLSD-derivation ending in the empty
clause is a WSLD-refutation for $\Pi \cup \{G_0\}$ and $\mathcal{R}$. Then, $\theta = \theta_1 \theta_2 \ldots \theta_n$ is the computed substitution and $\beta = \bigwedge_{i=1}^{n} \beta_i$ is its approximation degree. The output of a WSLD-refutation is the pair $\langle \theta(\mathcal{V} \mathcal{A}r(\mathcal{G})), \beta \rangle$, which is said to be a computed answer for $\Pi \cup \{G_0\}$ and $\mathcal{R}$.

**Example 3** Suppose we want to model the problem described in Example 1 in this FLP framework. A possible solution is the following program:

```plaintext
%% HANDLING VAGUENESS EXPLICITLY
bad˜so_so = 0.25. good˜very_good= 0.8. so_so˜good=0.5.

%% FACTS AND RULES WITH IMPLICIT VAGUENESS
film(dracula, so_so).
film(the_lords_of_the_rings, good).
film(the_godfather, very_good).

recommend(X):=- film(X, very_good).
```

Note that, if we launch the goal “?- recommend(X)”, the system first responds: “X= dracula” with “0.5”. The informal reason is that this subgoal weakly unifies with the head of the last rule, launching the subgoal “?- film(X, very_good)”. This subgoal, in turn, weakly unifies with the fact “film(dracula, so_so)”, obtaining the partial answer “X= dracula” with approximation degree “0.5” (because, by transitivity, “so_so” and “very_good” are similar with that approximation degree).

Analogously, the system provides the remainder of answers: “X=the_godfather” with “1.0” and “X=the_lords_of_the_rings” with “0.8”.

Analyzing this example, we can appreciate a simplification of its representation with regard to Example 1: we obtain a more intelible, condensed representation. Moreover, if we have to model a more complex problem than the one we are considering, we would have seen how the modeling process (using FLP language extending resolution) would have become more complex and difficult to treat: increasing the number of additional program rules and the difficulties to ascribe correct annotations.

Note that FLP languages that extend unification allow a natural integration of linguistic variables through the use of efficient algorithms for semantic unification which enable to associate to a (constant, function or predicate) symbol of the language a linguistic term [22]. However, the combination of weak unification with semantic unification features is not the norm but the exception in these languages. In Section 4 we argue that the acquisition of this feature is fundamental for a FLP language to be considered a Fuzzy Linguistic Logic Programming language.

Ending this section, let us to say that other authors have combined similarity relations into a FLP framework. Particularly, in [36] they present a model to define a similarity-based fuzzy unification mechanism inside the framework.
of multi-adjoint logic programming. In particular, a similarity-based unification approach is constructed by simply adding equality-like axioms for fuzzy similarities and using classical unification. This provides a semantic framework for logic programming with different notions of similarity. Although, from the theoretical point of view, this is a very elegant, general framework, its treatment of similarity relations have not been implemented yet and we think it might produce inefficiencies due to the huge proliferation of rules associated with the addition of the necessary equality-like axioms to the original program ¹.

3   FLP languages and levels of representation

A shared feature of all knowledge representation is that we have to manipulate two kind of entities: i) the facts, truths of a certain world that we want to represent; and ii) the representation of that facts using a formalism. These entities induce the necessity to distinguish between two levels [39]: i) the knowledge level, where facts and behavior are described; and ii) the symbolic level, where the knowledge level objects are represented by means of symbols of a formalism (with a clear operational semantics which may be implemented in a computer system). Due to its deductive capabilities, the symbolic level can simulate the real reasoning and to infer new knowledge.

Moreover, we have to distinguish between precise and uncertain or vague knowledge. When uncertainty is a linguistic phenomenon, it occurs implicitly at the knowledge level while it must be done explicit at the symbolic level. We mean that, if we say “John is young” and “Mary is middle-aged” the statements are precise but the concepts “young” and “middle-aged” are vague. Hence, at the symbolic level, you have to represent the precise statement that, for instance, “John is young” but also you have to characterize what is understood by “young” in some way.

In the last section we have presented two lines of FLP languages conceived to treat with uncertainty. The frameworks that extend resolution annotate the facts and rules with truth or confidence degrees and they break the natural connection between the knowledge level and the symbolic level. This is to say that, when representing knowledge, the treatment of precise and vague knowledge is mixed in these languages. The rupture of the natural connection between both levels causes that the representation of knowledge in these languages would be less adequate and more complex because it turns counterintuitive (see Example 1 and Table 1).

On the contrary, frameworks based on fuzzy unification allow a separate representation of precise and vague knowledge.

¹It is necessary to introduce one axiom for each constant, function and relation symbol existing in the program alphabet.
Table 1: Connection between the knowledge level and the symbolic level in fuzzy logic languages

<table>
<thead>
<tr>
<th>Knowledge Level</th>
<th>Symbolic Level</th>
<th>LP Framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>“John is young”</td>
<td>age(john, young)</td>
<td>Prolog (A standard LP language fails to represent vagueness)</td>
</tr>
<tr>
<td>“Mary is middle”</td>
<td>age(mary, middle)</td>
<td>Annotated FLP</td>
</tr>
<tr>
<td></td>
<td>age(john, young): 1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>age(john, middle): 0.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>age(mary, young): 0.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td>age(mary, middle): 1.0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>age(john, young)</td>
<td>Similarity-based FLP (Precise and vague Knowledge are separated)</td>
</tr>
<tr>
<td></td>
<td>age(mary, middle)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>young ~ middle = 0.8</td>
<td></td>
</tr>
</tbody>
</table>

In particular, Example 3 and Table 1 show that frameworks based on weak unification allow a natural integration of the knowledge level and the symbolic level, as they provide an independent treatment of vagueness. We think that this makes them more adequate for vague knowledge representation, as the translation of knowledge in the framework of these programming languages is more direct, intuitive and declarative.

4 Fuzzy Linguistic Logic Programming

Fuzzy Linguistic Logic Programming (FLLP) is centered in computing with linguistic terms and relations instead of computing with numbers and annotations (degrees or intervals of truth). As we have discussed, numeric annotations are not adequate to deal with vagueness and/or imprecision in a logic programming framework as they break the natural connection between the knowledge level and the symbolic level.

FLLP should be designed to make a clear distinction between the treatment of vagueness and precise knowledge.

To solve this problem, a programming language should have the ability to interpret a syntactic symbol as a linguistic term or word, in order to facilitate the connection between the natural language and the formal language.

Vagueness and/or imprecision may be handled by putting in relation linguistic terms or words by using semantic unification. Hence, semantic unification must be an important feature to be implemented efficiently.

FLLP must allow to work with fuzzy relations further than similarity relations, in order to model concepts or to represent terms ontologies.

The transitivity constrains imposed by similarity relations may produce conflicts with user’s specifications and may cause wrong modeling of vague
information. Let be the following sequence of colors: \{white, pale grey, grey, dark grey, black\}. You can admit that a color is close to its adjacent colors, or even that pale grey and dark grey have some closeness relation, but white is not close to black.

Also it should allow the access to fuzzy linguistic tools in a natural way, in order to facilitate the definition of fuzzy relations to the users of a FLLP system. We mean, tools like WordNet::Similarity\(^5\) or ConceptNet\(^6\).

Bousi∼Prolog (BPL for short) [21,24,49] is a FLP language that extends unification. It is freely available at the URL: http://dectau.uclm.es/bousi. Its syntax and operational semantics mainly correspond with the ones described in Section 2.2, but it may work with fuzzy relations that are not similarity relations. It can be seen as an incipient FLLP system since its design has been conceived to make a clear separation between precise knowledge, vague knowledge and control. Vague knowledge can be represented by fuzzy binary relations (proximity relations, similarity relations or even hierarchical relations) or the definition of linguistic terms (using specific directives —See Example 4). Linguistic terms acquire meaning through the association of fuzzy subsets to those terms. The association between syntactic symbols and fuzzy subsets is made at compilation time and the precise techniques needed to built this relationship have been investigated in [22]. Briefly, the method can be summarized into the following steps: first, we declare the linguistic labels of a linguistic variable, giving their membership functions; then we compare the membership functions using standard matching functions [8,13]; finally, as a result of that comparison, we obtain a fuzzy relation that, in general, is not a similarity relation. Once the fuzzy relation has been generated, the operational mechanism of the language manipulates the linguistic terms (fuzzy subsets) in a totally standard way as regular symbols of a first order language (that is, as constants, functions, or even predicate symbols) which are capable of participating in a weak unification process at the same level as the rest of symbols of the language alphabet. Therefore, we are able to manipulate a semantic unification process by means of a weak unification algorithm which is syntactic in nature.

On the other hand, since Bousi∼Prolog is able to work with electronic linguistic tools it can represent and manipulate linguistic relationships (as the synonymy) in a very natural, transparent way. For example, Bousi∼Prolog allows to work with WordNet::Similarity [44] which is based on WordNet [14]. Wordnet is a thesaurus and also an ontology. It groups English words into sets

---

\(^5\)WordNet::Similarity [44] (http://wn-similarity.sourceforge.net) is a software package that offers different measures of relatedness between pairs of concepts (or word senses), based on the WordNet lexical database [14].

\(^6\)ConceptNet [32] is a common sense knowledge base and natural language processing toolkit (http://concept.net.media.mit.edu/). ConceptNet is constructed as a network of semi-structured natural language fragments.
of synonyms called synsets. It also provides short, general definitions, and records the various semantic relations between these synonym sets. Thus, it is possible to know the meaning of a word and, at the same time, to associate it with other words using ontological relations like synonymy, antonymy, etc. The semantic relationships and the synsets can be used to obtain a degree of closeness between two words and thus, the set of words related to another word through the WordNet::Similarity API.

The following BPL program fragment illustrates some of the commented features.

**Example 4** Suppose a fragment of a deductive database that stores a semantic network with information about people’s names and eye colors and other features.

% BPL DIRECTIVES
:- transitivity(no).
% Definition of the linguistic variable age.
:-domain(age,0,100,years).
:-fuzzy_set(age, [young(0,0,30,50),
                    middle(20,40,60,80), old(50,80,100,100)]).

% PROXIMITY EQUATIONS
mixed˜brown=0.48. mixed˜gray=0.45. light˜green=0.4. light˜blue=1.0.
mixed˜green=0.6. dark˜brown=0.52. light˜gray=0.55. dark˜black=1.0.

% FACTS
is_a(john,person). age(john, young). eye_color(john, gray).
is_a(peter,person). age(mary, middle). eye_color(mary,blue).
is_a(mary,person). age(peter, age#68). eye_color(peter,brown).

% RULES
wise(X) :- age(X, very#old).

This small program fragment encapsulates many of the features mentioned as useful for a FLLP framework. First, a clean separation between precise and vague knowledge is performed. Vague concepts are defined separately either by specification of linguistic variables (directives domain and fuzzy_set which associates meaning to the linguistic labels) or by a set of proximity equations. In this example, the proximity equations are defining a proximity relation (because the directive “:-transitivity(no).” disables the automatic computation of the transitive closure for the fuzzy binary relation defined by the simple proximity equations; in fact, starting from the proximity equations, only the reflexive and symmetric closure needed to build a proximity relation are computed at compilation time). Precise knowledge are represented by facts and rules where crisp and vague concepts are related. Note also that it is possible to use modifiers over linguistic labels.

In this context, it is very natural to launch queries like “?- wise(X).” (who is wise?) or “?- eye_color(X, light), age(X, young).” (who is young and has light-colored eyes?) and to obtain a reasonable answer while other systems fail. This is because the BPL system is able to reason in presence of vagueness through
an extended operational semantics based on a weak unification algorithm that uses the information condensed into the proximity equations and the specification of the linguistic variables.

The computational steps performed by the weak resolution algorithm in order to solve the goal “?- eye_color(X, light), age(X, young).”, can be described informally as follows: first, the subgoal “eye_color(X, light)” is launched, and it weakly unifies with the fact “eye_color(john, gray)” obtaining the partial answer “X= john” with approximation degree “0.55” (because “gray” and “light” are close with that approximation degree). Then, the last subgoal “?- age(john, young)” is launched and it (weakly) unifies with the fact “age(john, young)” with approximation degree “1.0”. Therefore, when the whole computation terminates we obtain the computed answer ‘X=john’ with approximation degree “0.55” (the minimum of both approximation degrees).

As in a classical Prolog system, by backtracking we can obtain the other remaining answers. Specifically, for this program and goal, the subgoal “eye_color(X, light)” can weakly unify with the fact “eye_color(mary, blue)”, obtaining the partial answer “X= mary” with approximation degree “1.0” (because “blue” and “light” are close with that approximation degree). Then, the subgoal “?- age(mary, young)” is launched and it (weakly) unifies with the fact “age(mary, middle)” with approximation degree “0.375” (because the linguistic terms “young” and “middle” match with approximation degree “0.375”). Now, when the whole computation terminates we alternatively obtain the computed answer ‘X=mary’ with approximation degree “0.375”.

Finally observe that, the Bousi~Prolog system computes the relation between the linguistic terms “young” and “middle” by using standard matching techniques. The process is developed at compilation time and generates a set of proximity equations. In particular, the proximity equation “young ∼ middle = 0.375” is obtained, which is used (later on, at execution time) to produce the referred computed answer. All this process is automatic and transparent for the user.

Also, note that the syntax of the last program is very similar to the one of a standard Prolog, since the language is not substantially modified at syntactical level, except for the expressive resources that Bousi~Prolog provides to facilitate the management of vagueness and, specifically, the definition of proximity equations and linguistic variables, when it is required to establish semantical relations between symbols. Hence the design and implementation of the precise part of a program can be performed by using well-known techniques taken from the classical case.

These features convert Bousi~Prolog in a well-positioned system to implement the computing with words paradigm, as it provides two important capabilities: (a) the capability to specify the meaning of words drawn from natural language by using proximity relations, what allows to model semantical relations between words (e.g., synonymy, antonymy, hyperonymy, etc) and
to employ linguistic terms in order to treat with vagueness and/or impreci-
sion; (b) the ability to reason and compute with specified words and clauses
by means of a semantic unification algorithm based on weak unification that
allows to compute with semantic relations.

5 Scalability and Practical Applications

Scalability is a critical aspect of a software system. One of the interesting
things of our proposal is that we reduce semantic unification to (weak) syn-
tactic unification, which is always tractable and has good properties. For in-
stance, it always terminates returning a success or a failure. This reduction
is done at compilation time, what improves program execution. The tech-
niques to do that were first described in [22]. Tractability is possible as we
convert a fuzzy semantic unification problem into a crisp syntactic unifica-
tion problem that may be managed using the efficient operational machinery
of Prolog. Although, in the case of Bousi~Prolog proximity equations con-
siderably increases the number of program rules; indexing [45] and big data
techniques [10] could come into our help.

From a practical point of view, it is convenient to say that we are just solv-
ing real-world practical applications using the Bousi~Prolog system. In [49]
we described how Bousi~Prolog may contribute to resolve several problems
extracted from different application areas, where it is mandatory to deal with
vagueness and imprecision, such as: flexible deductive databases, fuzzy con-
trol, fuzzy experts systems, data retrieval or approximate reasoning.

Additionally, we are investigating how fuzzy logic programming could be
useful for real problems. We think that this kind of systems in which the spec-
ification of a problem can be performed in a linguistic and declarative way
could be useful for several applications.

• **Text Cataloging and Analysis.** This kind of declarative programming
  language has been employed with success for classifying unlabeled short-
texts [48]. This research can contribute to performing Sentic Computing
  as we can model lexical relations among symbols which could be used to
  represent affect relations by using sentimental words.

• **NoSQL Databases and Natural Language Queries on Databases.** This
  kind of declarative programming language has been employed with success
  for declarative linguistic queries on relational database [50]. It was a
  first approximation, our research can contribute to improve this method
  and to achieve a natural language queries on database (relational and
  not only relational). This helps to inexperienced users to formulate nat-
  ural queries without the necessity of knowing the database structure or
  a specific language.
• **Computing with Words and Perceptions:** lexical knowledge can contribute to improve the inference mechanisms of Computing with Words and Perceptions systems [62].

• **Semantic Web:** lexical knowledge can contribute to improve the inference mechanism of Semantic Web languages, for example in order to propose some extensions for OWL [17, 56].

• **Automatic 2D/3D scene generation from declarative linguistic description in order to build video tutorials by using natural language.** Traditional approaches for designing and constructing 2D/3D scenes are often not automatic, difficult, and tedious for non-expert users. Our idea is to study how a proximity-based logic programming language can be used for describing a 3D scene and how we could use its capability of inference and intuitive reasoning in order to create a motion in these scenes automatically generated.

6 Conclusions

In this paper we have discussed the representation of vague knowledge using two different kinds of fuzzy logic programming languages: annotated languages and those that extend the classical unification procedure.

We have shown that annotated fuzzy logic programming languages represent vague knowledge in a very low level, using (numeric) annotations in the rules and mixing precise and vague knowledge inextricably. On the other hand fuzzy logic programming languages that extend unification represent vague knowledge in a more abstract level, making a clear distinction between precise knowledge (represented by the facts and rules of a program) and vague knowledge (represented by fuzzy relations or linguistic variables). From our point of view, this separation is more declarative than the one dispensed by the approach based on annotations, and provides valuable benefices: it preserves the natural connection between the knowledge level and the symbolic level and facilitates the programming tasks.

Going further, we have proposed a fuzzy linguistic logic framework which allows the definition of vague concepts by means of fuzzy relations (in addition to similarities) or using linguistic terms. Also it allows the treatment of imprecision through (crisp or fuzzy) linguistic resources. We argue that this new framework, which is a natural evolution of the fuzzy logic programming approaches based on similarity relations, is well-positioned to implement the computing with words paradigm and it has a true representative in Bousi~Prolog.
Acknowledgments
We gratefully acknowledge the anonymous referees for their valuable comments and suggestions which greatly improve the readability of this paper.

This work has been partially supported by FEDER and the Spanish Ministry of Economy and Competition under grant TIN2013-45732-C4-2-P and it is done in collaboration with the research group, SOMOS (Software - MOdelling - Science), funded by the Research Agency and the Graduate School of Management of the Bío-Bío University under grant 130415 GI/EF.

References