A Similarity-Based WAM for Bousi~Prolog*

Pascual Julián-Iranzo and Clemente Rubio-Manzano
Department of Information Technologies and Systems,
University of Castilla-La Mancha
{Pascual.Julian}@uclm.es, {Clemente.Rubio}@alu.uclm.es

Abstract. Bousi~Prolog is an extension of the standard Prolog language with an operational semantics which is an adaptation of the SLD resolution principle where classical unification has been replaced by a fuzzy unification algorithm based on proximity relations defined on a syntactic domain. In this paper we present the structure and main features of a low level implementation for Bousi~Prolog. It consists in a compiler and an extension of the Warren Abstract Machine (WAM) able to incorporate fuzzy unification.

Keywords: Fuzzy Logic Programming, Fuzzy Prolog, Unification by Similarity, Warren Abstract Machine.

1 Introduction

The programming language we call Bousi~Prolog (BPL for short) [4] is an extension of the standard Prolog language with an operational semantics which is a variant of the SLD resolution procedure where the classical unification algorithm has been replaced by a weak unification algorithm based on proximity relations (fuzzy relations that fulfill the reflexive and symmetric properties). Hence, the operational mechanism is a generalization of the similarity-based SLD resolution principle [9]. Informally, this weak unification algorithm states that two terms $f(t_1,\ldots,t_n)$ and $g(s_1,\ldots,s_n)$ weakly unify if the root symbols $f$ and $g$ are approximate and each of their arguments $t_i$ and $s_i$ weakly unify. Therefore, the weak unification algorithm does not produce a failure when there is a clash of two syntactical distinct symbols, whenever they are approximate, but a success with a certain approximation degree. Hence, Bousi~Prolog computes answers as well as approximation degrees.

The syntax of BPL is mainly the Prolog syntax but enriched with a built-in symbol “∼” used for describing proximity/similarity relations by means of similarity equations of the form: `<symbol> ∼ <symbol> = <degree>`. Although, a similarity equation represents an arbitrary fuzzy binary relation, its intuitive reading is that two constants, n-ary function symbols or n-ary predicate symbols

* This work has been partially supported by FEDER and the Spanish Science and Education Ministry under grants TIN 2004-07943-C04-03 and TIN 2007-65749 and by the Castilla-La Mancha Regional Administration under grant PI1109-0117-4481.
are approximate or similar with a certain degree. That is, a similarity equation \( a \sim b = \alpha \) can be understood in both directions: \( a \) is approximate/similar to \( b \) and \( b \) is approximate/similar to \( a \) with degree \( \alpha \). Therefore, a Bousi-Prolog program is a sequence of Prolog facts and rules followed by a sequence of similarity equations.

Example 1. Suppose a fragment of a database that stores a semantic network with information about people’s names and hair color, as well as the approximate relation between black, brown and blond hair.

\[
\% \text{ BPL directive} \\
:- \text{transitivity(no)}. \\
\% \text{ FACTS} \\
is_a(john, \text{person}). \quad \text{hair_color(john, black).} \quad \text{black~brown}=0.6. \\
is_a(peter, \text{person}). \quad \text{hair_color(peter, brown).} \quad \text{black~blond}=0.3. \\
is_a(mary, \text{person}). \quad \text{hair_color(mary, blond).} \quad \text{blond~brown}=0.6. \\
\% \text{ SIMILARITY EQUATIONS}
\]

In a standard Prolog system, if we ask about whether peter’s hair is blond, “?- hair_color(peter, blond)”, the system fails. However BPL allows us to obtain the answer “Yes with 0.6”.

To obtain this answer, the BPL system operates as follows: i) First it generates the reflexive, symmetric closure of the fuzzy relation \( \{R(\text{black, brown}) = 0.6, R(\text{black, blond}) = 0.3, R(\text{blond, brown}) = 0.6\}\), constructing a proximity relation. ii) Then it tries to unify the goal \textit{hair_color(peter, blond)} and (eventually) the fact \textit{hair_color(peter, brown)}. Because there exists the entry \( R(\text{brown, blond}) = 0.6 \) in the constructed proximity relation (that is, brown is approximate to blond), the unification process succeeds with approximation degree 0.6.

The above example serves to illustrate both the syntax and the operational semantics of the language. Also, it is important to note that, in the last example, to inhibit the construction of the transitive closure (and therefore the construction of a similarity relation) has been crucial to model the information properly and to obtain a convenient result. This effect is obtained by the BPL directive \texttt{transitivity} which disables or enables the construction of the transitive closure of a fuzzy relation during the compilation process. If a similarity relation would be generated, the system would constructed the entries \( R(\text{brown, blond}) = 0.3 \) and \( R(\text{blond, brown}) = 0.3 \), overlapping the initial approximation degree provided by the user and leading to a wrong information modeling.

As it was shown in [4], there exist several practical applications for a language with the aforementioned characteristics: flexible query answering;

\footnote{For the sake of simplicity we only consider programs without variables in the examples. Of course, BPL permits programs ans goals containing variables.} \footnote{In the BPL system, \texttt{transitivity(yes)} is the current default option.} \footnote{Because a person with brown hair should be closer to a person with blond hair than a person with black hair is to a person with blond hair.}