An Efficient Proximity-based Unification Algorithm [*] (*Extended Abstract*)

Pascual Julián-Iranzo
Dept. of Information Technologies and Systems,
University of Castilla-La Mancha
Pascual.Julian@uclm.es

Fernando Sáenz-Pérez
Dept. of Software Engineering and Artificial Intelligence,
Universidad Complutense de Madrid
fernan@sip.ucm.es

Unification is a central concept in deductive systems based on the resolution principle. In turn, *Proximity relations* (i.e., reflexive, symmetric, fuzzy binary relations) are useful in knowledge representation to model the semantic closeness of different syntactic objects and to represent and manage vague or imprecise information.

Proximity relations, in combination with the unification algorithm, allow certain forms of approximate reasoning in a logic programming framework. We have proposed the use of proximity relations in the context of a (fuzzy) logic programming system, called Bousi∼Prolog, as a way for overcoming the limitations introduced by *similarity relations* (i.e. transitive proximity relations) to represent (fuzzy) information correctly.

Recently, we introduced an accurate definition of proximity between expressions (terms or atomic formulas) and a new unification algorithm able to manage proximity relations properly. Given a proximity relation $\mathscr{R}$ on a syntactic domain, our notion of proximity relies on the concept of proximity block (also known as a maximal clique on graph theory), which is a subset of maximally connected symbols. Two expressions e_1 and e_2 are approximate when they have the same set of positions, and their symbols, in their corresponding positions, belong to the same block and each symbol is always assigned to the same block.

However, the so-called *weak unification algorithm*, which is an extension of Martelli and Montanari's unification algorithm supported by the new notion of proximity, does not have an efficient implementation. This is due to the fact that it is necessary to deal with a set of proximity constraints to maintain the coherence of the unification process, which have to be checked for satisfaction in each unification step. This work is done by a satisfaction function, $\mathscr{S}at$. In this paper we present an elaborate method to implement the weak unification algorithm efficiently, which mainly consists in improving the performance of the satisfaction function $\mathscr{S}at$. The method follows the following guidelines:

1) At compile time, we analyze the proximity relation $\mathscr{R}$ extracting the set of proximity blocks and assigning to each symbol involved in the relation its corresponding block.

2) Using the information provided by the last analysis, also at compile time, we extend the proximity relation $\mathscr{R}$ into a new relation $\mathscr{R}\mathscr{B}$, enhancing $\mathscr{R}$ with information about the specific block corresponding to the elements involved in the relation entry. That is, an entry $\mathscr{R}(a,b) = \alpha$ is converted into $\mathscr{R}\mathscr{B}(a,b,B) = \alpha$, being B the block to which the related elements belong.

3) Thanks to the explicit use of information about blocks, we can replace the set of proximity constraints by an association list whose elements are pairs formed by the elements in the domain of the relation $\mathscr{R}\mathscr{B}$ and the block assigned to them by $\mathscr{R}\mathscr{B}$.

4) Finally, we can simplify the definition of the function $\mathscr{S}at$, converting it into a function which essentially performs an efficient membership test on an association list at run time.

[*]This work has been partially supported by FEDER and the State Research Agency (AEI) of the Spanish Ministry of Economy and Competition under grant TIN2016-76843-C4-2-R (AEI/FEDER, UE), projects CAVI-ART (TIN2013-44742-C4-3-R) and CAVI-ART-2 (TIN2017-86217-R), and Madrid regional project N-GREENS Software-CM (S2013/ICE-2731).

Submitted to:
PROLE 2018