Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 137 (2005) 69–103
www.elsevier.com/locate/entcs

Unfolding-based Improvements on Fuzzy Logic Programs [1]

Pascual Julián [2]

Dep. of Computer Science, ESI, Univ. of Castilla–La Mancha
Paseo de la Universidad, 4; 13071 Ciudad Real, Spain

Ginés Moreno [2]

Dep. of Computer Science, EPSA, Univ. of Castilla–La Mancha
Campus Universitario, s/n; 02071 Albacete, Spain

Jaime Penabad [2]

Dep. of Mathematics, EPSA, Univ. of Castilla–La Mancha
Campus Universitario, s/n; 02071 Albacete, Spain

Abstract

Unfolding is a semantics-preserving program transformation technique that consists in the expansion of subexpressions of a program using their own definitions. In this paper we define two unfolding-based transformation rules that extend the classical definition of the unfolding rule (for pure logic programs) to a fuzzy logic setting. We use a fuzzy variant of Prolog where each program clause can be interpreted under a different (fuzzy) logic. We adapt the concept of a *computation rule*, a mapping that selects the subexpression of a goal involved in a computation step, and we prove the independence of the computation rule. We also define a basic transformation system and we demonstrate its strong correctness, that is, original and transformed programs compute the same fuzzy computed answers. Finally, we prove that our transformation rules always produce an improvement in the efficiency of the residual program, by reducing the length of successful Fuzzy SLD-derivations.

Keywords: Fuzzy Logic Programming, Program Transformation.

[1] This work is partially supported by MCYT under grant TIN 2004-07943-C04-03
[2] Emails: {`Pascual.Julian, Gines.Moreno, Jaime.Penabad`}`@uclm.es`

doi:10.1016/j.entcs.2005.01.040

1 Introduction

Logic Programming [12] has been widely used for problem solving and knowledge representation in the past. Nevertheless, traditional logic programming languages do not incorporate techniques or constructs in order to treat explicitly uncertainty and approximated reasoning. *Fuzzy Logic* provides a mathematical background for modeling uncertainty and/or vagueness. Fuzzy logic relays on the concept of fuzzy set, the theory of fuzzy connectives (t-norms, t-conorms, etc.) and the extension of two-values classical predicate logic to a logic where formulas can be evaluated in the range of the $[0, 1]$ real interval (see [22] or [13] for a comprehensive introduction of this subject). Fuzzy sets [23] are objects introduced to deal with the fuzziness or vagueness we find in the real world when we try to describe phenomena that have not sharply defined boundaries. Given a set U, an ordinary subset A of U can be defined in terms of its *characteristic function* $\chi_A(x)$ (that returns 1 if $x \in A$ or 0 otherwise) which neatly specifies whether or not an element x is in A. On the other hand, a *fuzzy subset* A of U is a function $A : U \to [0, 1]$. The function A is called the *membership function*, and the value $A(x)$ represents the *degree of membership* (it is not meant to convey the likelihood that x has some particular attribute such as "young" [13]) of x in the fuzzy set A. Different functions A can be considered for a fuzzy concept and, in general, they will present a soft shape instead of the characteristic function's crisp slope of an ordinary set.

Fuzzy Logic Programming is an interesting and still growing research area that agglutinates the efforts to introduce Fuzzy Logic into Logic Programming. During the last decades, several fuzzy logic programming systems have been developed, where the classical inference mechanism of SLD-Resolution is replaced with a fuzzy variant which is able to handle partial truth and to reason with uncertainty. Most of these systems implement the fuzzy resolution principle introduced by Lee in [10], such as the Prolog-Elf system [4], Fril Prolog system [2] and the F-Prolog language [11].

On the other hand, there is also no agreement about which fuzzy logic must be used when fuzzifying Prolog. Most systems use min-max logic (for modeling the conjunction and disjunction operations) but other systems just use Lukasiewicz logic [7]. Other approaches are parametric with respect the interpretation of the fuzzy connectives, letting them unspecified to obtain a more general framework [21]. Recently, it has been appeared in [20] a theoretical model for fuzzy logic programming which deals with many values implications. Finally, in [19] we find an extremely flexible scheme where, apart from introducing negation and dealing with interval-valued fuzzy sets [8], each clause on a given program may be interpreted with a different logic. In this paper, we