Theory of Generalized Annotated Logic Programming and its Applications*

Michael Kifer†
Department of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11794.
E-Mail: kifer@cs.sunysb.edu

V.S. Subrahmanian
Department of Computer Science
University of Maryland
College Park, Maryland 20742.
E-Mail: vs@cs.umd.edu

Abstract

Annotated logics were introduced in [43] and later studied in [5, 7, 31, 32]. In [31], annotations were extended to allow variables and functions, and it was argued that such logics can be used to provide a formal semantics for rule-based expert systems with uncertainty. In this paper we continue to investigate the power of this approach. First, we introduce a new semantics for such programs based on ideals of lattices. Subsequently, some proposals for multivalued logic programming [5, 7, 32, 47, 40, 18] as well as some formalisms for temporal reasoning [1, 3, 42] are shown to fit into this framework. As an interesting by-product of this investigation, we obtain a new result concerning multivalued logic programming: a model theory for Fitting's bilattice-based logic programming, which until now has not been characterized model-theoretically. This is accompanied by a corresponding proof theory.

1 Introduction

Large knowledge bases can be inconsistent in many ways. Nevertheless, certain "localizable" inconsistencies should not be allowed to significantly alter the intended meaning of such knowledge bases. As classical logic semantics decrees that inconsistent theories have no models (and hence are meaningless from a model-theoretic point of view), classical logic is not the appropriate formalism for reasoning about inconsistent knowledge bases.

As a step towards the solution of this problem, annotated logic programs were introduced by Subrahmanian in [43] and were subsequently studied in [5, 7] by Blair and Subrahmanian. In [32, 33], Kifer and Lozinskii extended the theory to a full-fledged logic, and it was shown that a sound and complete proof procedure exists. More efficient proof procedures have been recently obtained, and implementations of these theorem provers have been designed (cf. [12, 26]). Kifer and Li [31] extended annotated programs in a different direction by allowing variables and evaluable function terms to appear as annotations. We will call such programs *generalized* annotated programs (*GAPs*, for short). The utility of annotated logics for reasoning with

*A preliminary report on this research has appeared in [34]

†Work supported in part by the NSF grant IRI-8903507.