Available online at www.sciencedirect.com
ScienceDirect
Journal of Applied Logic 5 (2007) 545–573
JOURNAL OF APPLIED LOGIC
www.elsevier.com/locate/jal

Disjunctive logic programming with types and objects: The DLV$^+$ system

Francesco Ricca, Nicola Leone *

Department of Mathematics, University of Calabria, Via P. Bucci, cubo 30b, Rende (CS) 87036, Italy

Received 27 October 2005; accepted 13 February 2006

Available online 14 August 2006

Abstract

The paper presents DLV$^+$, a Disjunctive Logic Programming (DLP) system with object-oriented constructs, including classes, objects, (multiple) inheritance, and types. DLV$^+$ is built on top of DLV (a state-of-the art DLP system), and provides a graphical user interface that allows one to specify, update, browse, query, and reason on knowledge bases. Two strong points of the system are the powerful type-checking mechanism and the advanced interface for visual querying.

DLV$^+$ is already used for the development of knowledge based applications for information extraction and text classification.

Keywords: Disjunctive logic programming; Objects; Types; Ontologies; Reasoning

1. Introduction

Disjunctive Logic Programming (DLP) is an advanced formalism for Knowledge Representation and Reasoning (KR&R) [12]. DLP is very expressive in a precise mathematical sense: it is able to express all problems belonging to the complexity class Σ_2^P.[1] Moreover, the availability of a pair of efficient DLP systems, like DLV [19], GnT [16] and, more recently, the disjunctive version of Cmodels [20] make DLP a powerful tool for developing advanced knowledge-based applications [3,22].

The recent application of DLP in the areas of Knowledge Management (KM), Security, and Information Integration [18,24], has confirmed, on the one hand, the viability of the DLP exploitation. On the other hand, it has evidenced some limitations of DLP language and systems. As far as the language is concerned, the need to represent complex real-world entities, like classes, objects, compound objects, and taxonomies, has emerged [24]. Moreover, DLP systems are missing tools for supporting the programmers, like type-checkers and easy-to-use graphical environments, to manage the large and complex domains to be dealt with in real-world applications.

This paper describes the DLV$^+$ system, a first step towards overcoming the above limitations. It is a cross-platform development environment for knowledge modeling and advanced knowledge-based reasoning. The DLV$^+$ system allows for the development of complex applications and allows one to perform advanced reasoning tasks in a user

* Corresponding author.
E-mail addresses: ricca@mat.unical.it (F. Ricca), leone@mat.unical.it (N. Leone).

[1] The class of all decision problems solvable in nondeterministic polynomial time by a Turing machine with an oracle in NP.

1570-8683/$ – see front matter
doi:10.1016/j.jal.2006.02.002

friendly visual environment. The DLV$^+$ system seamlessly integrates the DLV system [19] exploiting the power of a stable and efficient DLP solver.

A strong point of the system is its powerful language, extending DLP by object-oriented features. In particular, the language includes, besides the concept of *relation*, the object-oriented notions of *class, object* (class instance), *object-identity, complex object*, (*multiple*) *inheritance*, and the concept of modular programming by means of *reasoning modules*.

A *class* can be thought of as a collection of individuals that belong together because they share some features. An individual, or *object*, is any identifiable entity in the universe of discourse. Objects, also called class instances, are unambiguously identified by their object-identifier (oid) and belong to a class. A class is defined by a name (which is unique) and an ordered list of attributes, identifying the properties of its instances. Each attribute has a name and a type, which is, in truth, a class. This allows for the specification of *complex objects* (objects made of other objects).

Classes can be organized in a specialization hierarchy (or data-type taxonomy) using the built-in *is-a* relation (*multiple inheritance*).

Relationships among objects are represented by means of *relations*, which, like classes, are defined by a (unique) name and an ordered list of attributes (with name and type).[2]

As in DLP, logic programs are sets of logic rules and constraints. However, DLP$^+$ extends the definition of logic atom by introducing class and relation predicates, and complex terms (allowing for a direct access to object properties). In this way, the DLP$^+$ rules merge, in a simple and natural way, the declarative style of logic programming with the navigational style of the object-oriented systems. In addition, DLP$^+$ logic programs are organized in *reasoning modules*, taking advantage of the benefits of modular programming.

Importantly, the strongly-typed nature of DLP$^+$ allowed for the implementation of a number of *type-checking* routines that verify the correctness of a specification on the fly, resulting in an help for the programmer.

Moreover, DLV$^+$ offers several important facilities driving the development of both the knowledge base and the reasoning modules. Using DLV$^+$, developers and domain experts can create, edit, navigate and query object-oriented knowledge bases by an easy-to-use *visual environment*, enriched by a graphic *query interface* à la QBE.

In short, the contribution of the paper is twofold:

- We define a new language, named DLP$^+$, for Knowledge Representation and Reasoning, extending DLP with relevant constructs of the object-oriented paradigm, like Classes, Types, Objects and Inheritance. We provide a formal definition of both syntax and semantics of DLP$^+$, and illustrate its knowledge modeling features by examples. We analyze the computational complexity of the main decisional problems arising in the context of DLP$^+$.
- We design and implement a system supporting DLP$^+$, named DLV$^+$.
  The system offers all features of DLP$^+$, it provides a user friendly Graphical User Interface, and a powerful type checking mechanism, which supports the user in a fast development of error-free ontologies. DLV$^+$ is endowed also with a visual query interface, allowing to combine navigation and querying for powerful information extraction.
  The system is already employed in practice in a couple of applications for text classification and information extraction (see Conclusion).

The remainder of this paper is structured as follows. In the next section, we present an informal overview of the DLP$^+$ language. After that, we report in Section 3 the formal definition of DLP$^+$ ontologies; while Section 4 gives a formal account of Axioms and Reasoning Modules. Section 6 overviews the architecture and implementation of the DLV$^+$ system. In Section 5 we analyze the complexity of the main decisional problems arising in the context of DLP$^+$. In Section 7 we compare DLP$^+$ with a number of related languages. Finally, Section 8 we draw our conclusions.

[2] Note that, unlike objects, relation instances are not identified by means of oid's.