Journal of Logical and Algebraic Methods in Programming 88 (2017) 26–44

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp

From description-logic programs to multi-context systems

Luís Cruz-Filipe [a,*], Graça Gaspar [b], Isabel Nunes [b]

[a] Department of Mathematics and Computer Science, University of Southern Denmark, Denmark
[b] BioISI—Biosystems & Integrative Sciences Institute, Faculty of Sciences, University of Lisbon, Portugal

ARTICLE INFO

Article history:
Received 24 February 2016
Received in revised form 16 January 2017
Accepted 28 January 2017

Keywords:
Description logics
Logic programming
Semantics
Rule-based systems
Formal translations

ABSTRACT

The combination of logic program-style rules with other reasoning systems has been a fertile topic of research in the last years, with the proposal of several different systems that achieve this goal. In this work, we look at two of these systems, dl-programs and multi-context systems, which address different aspects of this combination, and include different, incomparable programming constructs. We prove that every dl-program can be transformed into a multi-context system in such a way that the different semantics for each paradigm are naturally related. As a consequence, constructions developed for dl-programs can be automatically ported to multi-context systems. In particular, we show how to model default rules over ontologies with the usual semantics.

1. Introduction

Several approaches combining rules and ontologies have been proposed in the last years for semantic web reasoning, e.g. [4,14,17,18,28,35]. Ontologies are typically expressed through decidable fragments of function-free first-order logic with equality, offering a very good ratio expressiveness/complexity of reasoning [2]. By extending ontologies with rule capabilities, together with non-monotonic features (in particular, the negation-as-failure operator from logic programming), one obtains a very powerful framework for semantic web reasoning where it is possible to write more expressive queries.

In this paper, we look at two of these systems: dl-programs [17,18] and multi-context systems (MCSs) [4], which address different aspects of this combination, and include incomparable programming constructs. These two paradigms differ in their essence – a dl-program is essentially a logic program that can query a single description logic knowledge base $\mathcal{L}$ and may "feed" its view of $\mathcal{L}$ with newly inferred facts, while MCSs consist of several knowledge bases, possibly expressed in different languages, each declaring additional rules that allow communication with the others.

We show that every dl-program can be transformed into a multi-context system. Although this transformation has been informally described earlier [5], the contribution of this work is not only making it precise, but also studying its theoretical properties, namely regarding semantic aspects, and discussing some practical implications. To the authors' knowledge, these aspects have never been addressed before. In particular, we show that our translation preserves the various existing semantics for each paradigm: answer sets for dl-programs become grounded equilibria for multi-context systems, whereas the well-founded semantics of a dl-program corresponds to the notion of well-founded belief set of a multi-context system. These results rely on a new definition of how to view a description logic knowledge base as a context of a multi-context system.

* Corresponding author. Fax: +45 6550 2373.
E-mail addresses: lcf@imada.sdu.dk (L. Cruz-Filipe), gg@di.fc.ul.pt (G. Gaspar), in@di.fc.ul.pt (I. Nunes).

http://dx.doi.org/10.1016/j.jlamp.2017.01.005
2352-2208/

Translations between different knowledge representation formalisms have been studied by several authors [20,29,32,36]. Such translations formally establish a precise relationship between the expressive power of two formalisms. More importantly, they allow for the reuse of tools and techniques developed for the more expressive formalism to the less expressive one, by means of the translation. In our case, our translation makes it possible to apply reasoners for multi-context systems also to the translation of dl-programs, without needing dedicated tools to deal with dl-programs. This is also the idea behind the implementation of the DL-plugin for `dlvhex` [21], which uses a translation of dl-programs into HEX-programs. On the other direction, constructions that were originally proposed for the less expressive formalism can be analyzed into the more general one by means of the translation. We illustrate this potential by showing how the original encoding of default rules in dl-programs [18] can be directly translated into MCSs, yielding a correspondence between extensions in default logic and equilibria in MCSs. Likewise, this translation could be applied e.g. to the systematic constructions for dl-programs described in [10] to yield similar techniques for MCSs.

The structure of the paper is as follows. Section 2 summarizes previous work on dl-programs and introduces our running example. Section 3 introduces multi-context systems and presents a translation from dl-programs to these, together with proofs of equivalence between the different semantics of both systems. Section 4 discusses in more detail two aspects of the translation: how to extend it to a more expressive variant of dl-programs; and how to use it to extend the encoding of default rules in dl-programs from [18] to the framework of MCSs. Section 5 concludes the presentation.

This article extends work previously published in [9,11], namely by including the proofs of all results and a more detailed formal analysis of the design options and their consequences.

1.1. Related work

Combining description logics (DLs) with rules has been a leading topic of research for the past 15 years.

Description logic programs, or dl-programs [17,19], are heterogeneous combinations of rules and ontologies, which are loosely coupled: the set of rules and the ontology are kept separate. The connection between these two components is achieved through the inclusion of *dl-atoms* in rule bodies, which perform queries to the knowledge base, possibly modifying it (adding facts) for the duration/purpose of each individual query. In particular, [17,19] illustrate this approach for the logics of OWL-LITE and OWL-DL. The semantics of dl-programs extends two different semantics for ordinary normal logic programs – answer-set semantics and well-founded semantics.

A number of encouraging results for dl-programs are also presented in [17]: (i) data complexity is preserved for dl-programs reasoning under well-founded semantics as long as dl-queries in the program can be evaluated in polynomial time, and (ii) reasoning for dl-programs is first-order rewritable (and thus in LOGSPACE under data complexity) when the evaluation of dl-queries is first-order rewritable and the dl-program is acyclic. These nice results are a consequence of the use of Datalog: complexity of query evaluation on both Datalog and stratified Datalog$^{\neg}$ programs is data complete for PTIME and program complete for EXPTIME, as shown in [12].

There are other formalisms that allow combination of different reasoning systems. Hybrid MKNF knowledge bases [32] are a homogeneous approach, in the sense that the rules and the description logic knowledge base are not kept separate, as in dl-programs; for this reason, this formalism lacks modularity. On the other hand, various heterogeneous proposals have emerged to cope with several components, allowing to deal with information coming from different sources. HEX-programs [20], which generalize dl-programs, are higher-order logic programs with external atoms that may pose queries to systems using different languages.

In contrast with these approaches, where communication between system components is centralized in the logic program part of the system, multi-context systems [4] (MCSs) were originally designed to bring together characteristics of both heterogeneous monotonic [24,31] and homogeneous non-monotonic systems [7,34], capitalizing on both worlds.

Multi-context systems are similar to HEX-programs, in that they are heterogeneous non-monotonic systems, whose components (called *contexts*) are knowledge bases that can be expressed in different logics (e.g., a theory in classical logic, a description logic knowledge base, a set of modal or temporal logic formulas, or a non-monotonic formalism such as a logic program under answer set semantics, or a set of default logic rules). Unlike HEX-programs, however, MCSs' inter-component communication is distributed among the contexts via sets of (non-monotonic) *bridge rules*.

Since they were originally proposed, several variations of MCSs have been studied that add to their potential fields of application. Managed MCS [5] generalize bridge rules by allowing arbitrary operations on contexts (e.g. deletion or revision operators) to be freely defined. Relational MCSs [22] introduce variables and aggregate expressions in bridge rules, allowing a formal first-order syntax, and extending the semantics of MCSs accordingly. Dynamic MCSs [13] are designed to cope with situations where knowledge sources and their contents may change over time and are not known *a priori*. Evolving MCSs [26] extend this idea by incorporating knowledge resulting from dynamic observations through different belief change operations with different levels of persistence. This framework is further extended in [25] with the notion of evolving bridge rules.

A different line of research addresses techniques for dealing with inconsistency (non-existence of a model) within MCSs. The authors of [15] propose a declarative policy language, together with methodologies to apply it, to provide a means to create policies to avoid or repair inconsistencies in MCSs in a controlled way (e.g. specifying which inconsistencies can be repaired automatically, and which ones need external input by a human operator).