PROLOG for Expert Knowledge Using Domain–Specific and Controlled Natural Languages

Dietmar Seipel*, Falco Nogatz*, Salvador Abreu†

* Department of Computer Science, University of Würzburg
Am Hubland, 97074 Würzburg, Germany
{ dietmar.seipel, falco.nogatz }@uni-wuerzburg.de

† LISP and Department of Computer Science, University of Évora
Rua Romão Ramalho, 59, 7000 Évora, Portugal
spa@uevora.pt

Abstract

For representing knowledge in intelligent systems in a declarative and natural way, two major approaches have been used in the past: (1) domain–specific languages (DSLs), and (2) controlled natural languages (CNLs). Early DSLs were defined by technical and abstract syntaxes, e.g., for EBNF and regular expressions. More recently, there is a trend in defining DSLs based on natural language, e.g., for the standard relational query language SQL. On the other hand, CNLs stem from computer linguistics, driven by the idea of representing knowledge as a subset of natural languages. Both approaches have proven very useful in many applications.

In this paper, we discuss the two approaches for knowledge representation, namely using DSLs and CNLs. PROLOG has a long history in both worlds: with the help of user–defined operators, term expansions, and definite clause grammars, it has proven to be very suitable for defining new DSLs with a natural language flavour.

1. Introduction

In the field of logic programming, Alain Colmerauer pioneered using the PROLOG language (Colmerauer, 1990) for natural language processing (NLP). The past few decades have brought substantial progress in natural language processing as well as reasoning in knowledge bases, including non–montonic reasoning. In this paper, we investigate knowledge representation using controlled natural languages (CNL) and domain–specific languages (DSL). In both cases, logic programming concepts can be very helpful and may lead to a very natural and transparent syntax.

Recently, Domain–Specific Languages (Kosar et al., 2010) have become a common approach in modeling knowledge for a specific application area. The aim is to provide an interface that can be easily used by domain experts of the given field, who are often not familiar with a general–purpose programming language (GPL). Due to the close affinity to their application areas, DSLs are often more appropriately called specifications, definitions, or descriptions (Mernik et al., 2005). There are many cases where DSLs are just a formal representation of expert knowledge without the ability to be executable at all, e.g. the Unified Modeling Language (UML) or the extended Backus–Naur form (EBNF) for grammars.

PROLOG is well suited for the definition of new DSLs, as it provides three user–level mechanisms to extend the programming language: (1) operators; (2) a macro–like term and goal expansion; and (3) a powerful yet intuitive way to express grammars using Definite Clause Grammars (DCG) (Pereira and Warren, 1980). Together with the quasi–quotations (Wielemaker and Hendricks, 2013) recently introduced in SWI–PROLOG – a concept adopted from languages like Haskell and JavaScript –, external DSLs can be directly used within existing PROLOG programs. Given the conciseness and flexibility of these mechanisms, PROLOG is very suitable for rapid–prototyping of a newly defined DSL (Kosar and Mernik, 2006).

A CNL is a subset of a natural language designed for the documentation, specification and representation of expert knowledge. A survey and classification of CNLs is given in (Kuhn, 2014). E.g., Attempto Controlled English (ACE) (Fuchs and Schwitter, 1996) is a PROLOG–based CNL with an automatic and unambiguous translation into first–order logic. (Fuchs et al., 2006) argues that ACE meets the challenges of knowledge representation, reasoning, interoperability and user interfaces. Recently, a multi–lingual semantic wiki based on ACE and grammars has been introduced in (Kaljurand and Kuhn, 2013).

2. Modelling Declarative Expert Knowledge

We use logic programming and PROLOG for modelling and managing expert knowledge. Following the declarative idea, this knowledge is controlled by standard evaluation methods (Kowalski, 1979). Declarative expert knowledge is often given in the form of rules. However, the syntax of these quickly becomes too complicated for domain experts not familiar with existing formalisms like RDF. Using PROLOG, the rules can be specified in a DSL inspired by natural language. Thus, the actual modeling DSL becomes executable, and it can be managed in deductive databases and evaluated using the well–known forward–chaining approach. Because of the existing work in using PROLOG for knowledge representation, e.g., with SWI–PROLOG's libraries for RDF handling (Wielemaker et al.,

2016), the expert knowledge can be enriched by integrating ontologies. Based on the declarative rules and the integration with the PROLOG–based deductive database system DDBASE, multiple rules given in a PROLOG DSL can be compared, graphically analysed by domain–experts, and evaluated, resulting in an extensible system for hybrid expert knowledge. Forward–chaining derivations can be visualised using proof trees, and automated reasoning helps to detect contradictions. This combination of declarative modeling using a natural language–like DSL with a declarative rule evaluation mechanism seems promising, as it results in a reasonable and easily adaptable rule base, while still being consultable and executable.

We define a DSL whose main part is a set of *if–then* rules which hold the expert knowledge in a declarative, textual format. This language may be directly mapped to PROLOG. In previous work, we introduced a rule concept for expert knowledge and methods for querying and visualising the rules in various application domains, including medical diagnosis (Seipel et al., 2005). In (von der Weth et al., 2016), we transposed this concept to the field of organisational psychology, and gave a simple definition of the rule language as an internal DSL in PROLOG. In (Seipel et al., 2016), we formally describe the rule language using a context–free grammar with a focus on the integration into a workflow for collecting rules in the field of change management. (Seipel et al., 2017) contains some concepts developed in earlier papers on medical diagnosis (Seipel et al., 2005; Seipel and Baumeister, 2008).

We discuss two approaches of implementing the rule language both as an internal and as an external DSL in PROLOG. In order to provide a simple interface with meaningful error messages, we integrate the rule language directly into PROLOG source code using quasi–quotations.

3. Controlled Natural Languages in PROLOG

Controlled Natural language can be used as a declarative and application–dependent specification language. CNL is a subset of natural language that can be accurately and efficiently processed by a computer, since it is formal and executable, but it is expressive enough to allow natural usage by non–specialists.

In principle, there is a huge overlap between DSLs and CNLs. E.g., the DSL for if–then rules introduced in (Seipel et al., 2017) is similar to a CNL. Further CNLs can be derived from (Seipel et al., 2005) for rule bases in medicine.

There are several languages which are DSLs but not CNLs, and vice versa, for instance context–free grammars, the relational query language SQL, and regular expressions are DSLs but non CNLs.

In the future, modern intelligent information systems can benefit greatly from natural *query languages*. A PROLOG–based query language for hybrid knowledge bases including relational and XML databases has been developed by (Seipel, 2015). This might be extended to a natural language interface using PROLOG–technology.

In the system Attempto, specifications in CNL are translated to PROLOG using a DCG enhanced by feature structures. Inter–text references of the specification, e.g. anaphora, are resolved with the help of discourse representation theory (DRT).

4. Conclusion

To summarise, PROLOG has proven effective to rapidly implement both DSLs and CNLs. It is flexible and expressive enough to act as a vehicle for domain experts to code complex sets of rules, without having to learn a formal computer language.

5. References

Colmerauer, Alain, 1990. An Introduction to Prolog III. *Communications of the ACM*, 33(7):69–90.

Fuchs, N.E., K. Kaljurand, and G. Schneider, 2006. Attempto Controlled English Meets the Challenges of Knowledge Representation, Reasoning, Interoperability and User Interfaces. In *Proc. FLAIRS*.

Fuchs, N.E. and R. Schwitter, 1996. Attempto Controlled English (ACE). In *Proc. CLAW*.

Kaljurand, K. and T. Kuhn, 2013. A Multilingual Semantic Wiki Based on Attempto Controlled English and Grammatical Framework. In *Proc. ESWC*.

Kosar, Tomaž and Marjan Mernik, 2006. Embedded domain–specific languages in Prolog. *Acta Electrotechnica et Informatica*, 6(1):3.

Kosar, Tomaž, Nuno Oliveira, Marjan Mernik, Maria João Varanda Pereira, Matej Črepinšek, Daniela da Cruz, and Pedro Rangel Henriques, 2010. Comparing general–purpose and domain–specific languages: An empirical study. *Computer Science and Information Systems*, 7(2):247–264.

Kowalski, Robert, 1979. Algorithm = logic + control. *Communications of the ACM*, 22(7):424–436.

Kuhn, T., 2014. A Survey and Classification of Controlled Natural Languages. *Computational Linguistics*, 40(1):121–170.

Mernik, Marjan, Jan Heering, and Anthony M Sloane, 2005. When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, 37(4):316–344.

Pereira, Fernando CN and David HD Warren, 1980. Definite clause grammars for language analysis – a survey of the formalism and a comparison with augmented transition networks. *Artificial intelligence*, 13(3):231–278.

Seipel, D., 2015. Knowledge Engineering for Hybrid Deductive Databases. In *Proc. 29th Workshop on Logic Programming (WLP 2015)*.

Seipel, D. and J. Baumeister, 2008. Declarative Specification and Interpretation of Rule–Based Systems. In *Proc. 21st International Florida Artificial Intelligence Research Society Conference (FLAIRS 2008)*. AAAI Press.

Seipel, D., J. Baumeister, and M. Hopfner, 2005. Declaratively Querying and Visualizing Knowledge Bases in XML. In *Proc. 15th International Conference on Applications of Declarative Programming and Knowledge Management (INAP 2004)*, LNAI 3392. Springer.

Seipel, D., F. Nogatz, and S. Abreu, 2017. Domain–Specific Languages in PROLOG for Declarative Expert

Knowledge in Rules and Ontologies. *Computer Languages, Systems & Structures*.

Seipel, D., R. von der Weth, S. Abreu, F. Nogatz, and A. Werner, 2016. Declarative Rules for Annotated Expert Knowledge in Change Management. In *5th Symposium on Languages, Applications and Technologies (SLATE)*, volume 51.

von der Weth, R., D. Seipel, K. Schubach, F. Nogatz, and A. Werner, 2016. Modellierung von Handlungswissen aus fragmentiertem und heterogenem Rohdatenmaterial durch inkrementelle Verfeinerung in einem Regelbanksystem. In *Journal Psychologie des Alltagshandelns*.

Wielemaker, Jan, Wouter Beek, Michiel Hildebrand, and Jacco van Ossenbruggen, 2016. ClioPatria: A SWI–Prolog Infrastructure for the Semantic Web. *Semantic Web*, 7(5):529–541.

Wielemaker, Jan and Michael Hendricks, 2013. Why It's Nice to be Quoted: Quasiquoting for Prolog. In *Proc. 23rd Workshop on Logic–based Methods in Programming Environments (WLPE 2013)*.