A second life for Prolog

Jan Wielemaker

Centrum voor Wiskunde en Informatica (CWI)
Science Park 123, 1098 XG Amsterdam, Netherlands
J.Wielemaker@cwi.nl

Abstract

The logic programming language Prolog was 'invented' by Alain Colmerauer in 1972 with as main target natural language parsing and translation. Since then, the dominant paradigm for natural language handling moved from symbolic logic to statistical models. In a series of three tutorials and hands-on sessions we will (re-)introduce Prolog and explain how Prolog is complementary to recent programming tools.

Prolog for language

Prolog has its foundation in first order logic. Specifically, knowledge is represented as Horn clauses and an answer is searched for using SLD resolution. In a nutshell, this implies Prolog searches for possible values of variables that satisfy the rules and facts of the program using depth-first search, possibly backtracking to previous states if the current path doesn't lead to success. The resulting language has the following properties: (1) the program has a logical reading: an obtained answer is true as a consequence of a logical specification, (2) the program has a procedural reading: as the search strategy is easily understood and predictable, we can make a Prolog program perform actions in a specific order and (3), Prolog programs are non-deterministic: they can offer —on demand— multiple answers to the same problem.

The above properties are promising for natural language processing. Prolog's non-determinism can deal with the ambiguity of natural language, while its logic foundation allow expressing knowledge about the world and reason about the relation between alternative interpretations of natural language and world knowledge. Colmerauer's view created one clean and abstract formalism to deal with language and knowledge.

As we have seen, symbolic logic proved insufficient for dealing with the ambiguity of language. Prolog was replaced by statistical models, trained from large annotated text corpora. Not completely though. For example the leading parser for Dutch, Alpino, is written in Prolog. Watson's understanding of questions is in part written in Prolog and we find parsers for controlled languages such as ACE written in Prolog.

Prolog as a language

In the meanwhile Prolog evolved. XSB-Prolog (Freire et al., 1997) added SLG resolution which avoids both repetitive re-computation of the same answers and non-termination due to left-recursion. It forms the basis of F-logic which provides the knowledge representation for the CYC project (Lenat et al., 1990). Added in the 90s, constraint logic programming allow for incorporating domain

knowledge to find answers in a smarter way than generate and test as well as lazy evaluation. ProbLog (Bruynooghe et al., 2010) and similar formalism allow annotating facts with probabilities. This can be used to learn probabilities, estimate the probability of a derivation or find the most likely derivation.

The implementations evolved, providing much better memory management, indexing, concurrency and interfaces to many modern tools, data formats and protocols. Notably SWI-Prolog offers extensive support for web services including Pengines, Prolog engines on the web that can be programmed through an extensive web-based IDE and may be controlled from several languages.

Where are the opportunities?

Prolog is not the ideal language for implementing neural networks or statistical analysis tools. It can however naturally represent and reason over a large number of popular types of data, notably tabular data (relational databases), tree structured data (XML) and graph data (RDF). For small to medium sized data (up to millions of facts), the native in-memory representation of modern Prolog systems provide excellent performance. For really large data sets it can bridge to specialised external tools, providing the same concise and flexible interface to the Prolog programmer. Concisely represented rule sets are the perfect glue to bridge the gaps between RDF (linked data), relational databases, parse trees, etc.

Acknowledgements

This research was partially supported by the VRE4EIC project, a project that has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 676247.

References

Bruynooghe, Maurice, Theofrastos Mantadelis, Angelika Kimmig, Bernd Gutmann, Joost Vennekens, Gerda Janssens, and Luc De Raedt, 2010. Problog technology for inference in a probabilistic first order logic. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*. Amsterdam, The Netherlands, The Netherlands: IOS Press.

Freire, Juliana, David S. Warren, Konstantinos Sagonas, Prasad Rao, and Terrance Swift, 1997. XSB: A system for efficiently computing well-founded semantics. In Proceedings of LPNMR 97. Berlin, Germany: Springer Verlag. LNCS 1265.

Lenat, Douglas B., R. V. Guha, Karen Pittman, Dexter Pratt, and Mary Shepherd, 1990. Cyc: Toward programs with common sense. Commun. ACM, 33(8):30–49