

What Next for ASP? (A Not-Entirely-Well-Informed Opinion)

James Delgrande

School of Computing Science
Simon Fraser University
Burnaby BC, V5A 1S6, Canada
jim@cs.sfu.ca

Preamble: The inception of Answer Set Programming (ASP) can be marked by the appearance of the stable model semantics [GL88], something over 20 years ago. The roots of ASP in turn can be traced to work in nonmonotonic reasoning, notably Default Logic [Rei80]. With the advent of efficient ASP solvers, as exemplified by *smodels* [NS97] and *dlv* [ELM⁺97], there was a great deal of interest and excitement over the application of ASP (broadly taken) to various problems, along with its use as a modelling tool. Indeed, applications have been proposed in a wide variety of fields, including bioinformatics, configuration, database integration, diagnosis, hardware design, insurance industry applications, model checking, phylogenesis, planning, security protocols, and high-level control of the US space shuttle [Sch08]. In concert with these applications, there has been a widespread flowering of ASP solvers built on various technologies [DVB⁺09].

With these successes, attention has turned to the application of ASP to other interesting and challenging problems. Since it can be argued (or at least taken as a position for debate) than *any* research in CS should be with an eye to eventual practical application, the question of the application of ASP can be seen as obliquely asking, *What is the potential role of ASP in AI/CS/the world at large?*

Applying ASP: Broadly, and for purposes of discussion,¹ we can consider three non-exclusive and non-exhaustive areas of application: to other areas of AI, to areas of interest in mathematics, and to “real world” problems. Consider each in turn:

ASP and AI: To consider the role of ASP in AI is to essentially ask about the suitability of ASP for providing general KR languages. Certainly, earlier surveys such as [BG94,DB96] regarded Knowledge Representation and Reasoning as the principal focus of ASP, as implicitly does [Bar03].

Two points can be made in this regard. First, the trend with respect to applications is *away* from KR. Witness, for example, the problem suite in [DVB⁺09],

¹ which is to say this classification shouldn't be taken too seriously

which would seem to not have a great deal to do with KR.² As well, as implementations have developed and evolve, enhancements have been added to the language; these can be declarative (e.g. aggregates and cardinality constraints) or procedural (e.g. adding options to control search). In the latter case, representational force is lost, in favour of procedural gain. Second, and counter to the first point, research in ASP on KR can nonetheless be regarded as alive and well, given work on strongly related topics such as action languages and causal representations, and representations of interesting domains using such formalisms (e.g. [AEL⁺04]). This suggests of course the potential role of ASP as a target language for higher-level encodings of problems.³

ASP and mathematics: Most of the challenge problems for ASP are from graph theory, combinatorics, or number theory. This arguably reflects the shift in emphasis toward solving constraint problems (or maybe just the prevalence of toy problems from these areas). Regardless, most proposed applications seem to lie in these areas, one way or another. So one possibility is to attempt to solve specific open problems in mathematics via ASP. One example is determining the fifth Schur number;⁴ a second is suggested at the conclusion of this article.

ASP and “real world” problems: This of course is a highly interesting, difficult, and potentially important long-term use of ASP. There are various impediments that need to be addressed; in particular, current implementations would need to scale up in various ways. Grounding is clearly a bottleneck. As well, there is a need for a general programming methodology, and perhaps a better understanding of the relation between problem type and search strategy. Similarly there would seem to be a need for programming environments and tools for the construction of large programs. Work on locality (perhaps involving conditional independence structures) and structuring blocks of rules would be useful. Certainly if similar work in KR (e.g. [Mor98]) is anything to go by, such applications promise to be messy.

A Modest Proposal: A specific problem area that falls into the second category above, yet may have practical application while skirting issues concerning scalability and software engineering, is that of *balanced incomplete block designs* (BIBD) [CD06]. Roughly a BIBD is a set X and a collection of subsets of X , called *blocks*, such that each block is the same size and each element of X appears

² For a perhaps unfair comparison, consider in contrast the Challenge Problems for Commonsense Reasoning at www-formal.stanford.edu/leora/commonsense.

³ This also suggests an interesting project, comparing the two leading logic programming paradigms via their respective ease (or lack thereof) for representing action formalisms: The University of Toronto action group translates the situation calculus into PROLOG typically, while action languages may be translated into extended logic programs.

⁴ I’ve shamelessly cribbed this example from Torsten Schaub’s position paper, who in turn credits it to Mirek Truszczyński.

in the same number of blocks. Hence BIBDs come with a compact, austere, theoretical specification. They also have significant real world application, as they are fundamental in experimental design, and have applications in software testing and cryptography. They encompass a large class of problems, and include as subareas Steiner triple systems, finite projective planes, and Latin squares. Moreover, they would seem ideally suitable for ASP encodings, as the general problem for BIBDs is to find a solution for a given set of parameters or show that no solution exists.

References

- [AEL⁺04] V. Akman, S. Erdoğan, J. Lee, V. Lifschitz, and H. Turner. Representing the zoo world and the traffic world in the language of the causal calculator. *Artificial Intelligence*, 153(1-2):105–140, 2004.
- [Bar03] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.
- [BG94] Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19:73–148, 1994.
- [CD06] Charles J. Colbourn and Jeffrey H. Dinitz. *Handbook of Combinatorial Designs, Second Edition*. Chapman & Hall/CRC, 2006.
- [DB96] J. Dix and G. Brewka. Knowledge representation with logic programs. *Fachberichte Informatik 15–96*, Universität Koblenz-Landau, 1996.
- [DVB⁺09] M. Denecker, J. Vennekens, S. Bond, M. Gebser, and M. Truszczyński. The second answer set programming competition. In E. Erdem, F. Lin, and T. Schaub, editors, *Proc. LPNMR*, volume 5753 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 2009.
- [ELM⁺97] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, , and F. Scarcello. A deductive system for nonmonotonic reasoning. In J. Dix, U. Furbach, and A. Nerode, editors, *Proc. LPNMR*, volume 1265 of *Lecture Notes in Artificial Intelligence*, pages 363–374. Springer Verlag, 1997.
- [GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *Proc. ICLP*, pages 1070–1080. The MIT Press, 1988.
- [Mor98] L. Morgenstern. Inheritance comes of age: Applying nonmonotonic techniques to problems in industry. *Artificial Intelligence*, 103:1–34, 1998.
- [NS97] I. Niemelä and P. Simons. Smodels: An implementation of the stable model and well-founded semantics for normal logic programs. In J. Dix, U. Furbach, and A. Nerode, editors, *Proc. LPNMR*, volume 1265 of *Lecture Notes in Artificial Intelligence*, pages 420–429. Springer-Verlag, 1997.
- [Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.
- [Sch08] T. Schaub. Here’s the beef: Answer set programming ! In A. Dovier, M. Garcia de la Banda, and E. Pontelli, editors, *Proc. ICLP*, volume 5366 of *Lecture Notes in Computer Science*. Springer-Verlag, 2008.