Here's the beef: Answer Set Programming !

Torsten Schaub*

Universität Potsdam, Institut für Informatik, August-Bebel-Str. 89,
D-14482 Potsdam, torsten@cs.uni-potsdam.de

At the occasion of the Third International Conference on Principles of Knowledge Representation and Reasoning [1] in 1992, Ray Reiter delivered an invited talk entitled *"Twelve Years of Nonmonotonic Reasoning Research: Where (and What) Is the beef?"*,[1,2] reflecting the state and future of the research area of Nonmonotonic Reasoning (NMR;[2]). Ray Reiter describes it in [3] as a "flourishing subculture" making many outside researchers "wonder what on earth this stuff is good for." Although he seemed to be rather optimistic about the future of NMR, he nonetheless saw its major contribution on the theoretical side, providing "important insights about, and solutions to, many outstanding problems, not only in AI but in computer science in general." Among them, he lists "Logic Programming implementations of nonmonotonic reasoning".

Although the link between Michael Gelfond and Vladimir Lifschitz' Stable Model Semantics for Logic Programming [4] and NMR formalisms like Ray Reiter's Default Logic [5] were discovered soon after the proposal of Stable Model Semantics,[3] it still took some years until the first such implementation was conceived, namely, the *smodels* system [8, 9]. The emergence of such a highly efficient and robust system has boosted the combination of Logic Programming and NMR and finally led to a novel declarative programming paradigm, referred to as Answer Set Programming (ASP;[10–14]). Since its inception, ASP has been regarded as the computational embodiment of Nonmonotonic Reasoning and a primary candidate for an effective tool for Knowledge Representation and Reasoning. After all, it seems nowadays hard to dispute that ASP brought new life to Logic Programming and NMR research and has become a major driving force for these two fields, helping dispel gloomy prophecies of their impending demise.

Meanwhile, the prospect of ASP has been demonstrated in numerous application scenarios, including bio-informatics [15, 16], configuration [17], database integration [18], diagnosis [19], hardware design [20], insurance industry [21], model checking [22], phylogenesis [23, 24], planing [12], security protocols [25], etc.[4] A highlight among these applications is arguably the usage of ASP for the high-level control of the space shuttle [26, 27]. The increasing popularity of ASP is for one thing due to the availability of efficient off-the-shelf ASP systems [28–32] and for another due to its rich modeling language, jointly allowing for an easy yet efficient handling of

---

* Affiliated with Simon Fraser University, Canada, and Griffith University, Australia.

[1] By then twelve years after the publication of the Special Issue of the Artificial Intelligence Journal on Nonmonotonic Reasoning.

[2] See also http://en.wikipedia.org/wiki/Where's_the_beef

[3] Logic Programming under Stable Model Semantics turned out to be a special case of Default Logic, with stable models corresponding to default extensions [6, 7].

[4] See also http://www.kr.tuwien.ac.at/research/projects/WASP/report.html

knowledge-intensive applications. Essentially all ASP systems that have been developed so far contain two major components. The first of them, a *grounder*, grounds an input program, that is, produces its compact propositional equivalent, often by appeal to advanced database techniques. The input language goes well beyond that of Prolog, offering among others, integrity constraints, classical negation, disjunction, and various types of aggregates. The second component, a *solver*, accepts the ground program and actually computes its answer sets (which amount to the stable models of the original program). Modern ASP solvers rely on advanced Boolean constraint solving techniques, stemming from the area of Satisfiability Checking and allowing for tackling application problems encompassing millions of variables. All in all, ASP has become an efficient and expressive declarative problem solving paradigm, particularly well-suited for knowledge-intensive applications.

Taking up Ray Reiter's challenge after sixteen years, *my* obvious answer is that Answer Set Programming is the *beef* of twenty-eight years of NMR research! Although twenty-eight years appear to be quite a while, successful neighboring areas such as Description Logics (DLs) and Satisfiability Checking (SAT) look back onto similar histories, taking major references in their field, like [33] and [34, 35], respectively. Nonetheless both areas have prospered in recent years due to their success in industrially relevant application areas. SAT is the key technology underlying Bounded Model Checking [36] and DLs have become standard ontology languages for the Semantic Web [37]. Although different factors have abetted these success stories, in the end, their breakthrough was marked by their establishment as salient technologies in their respective application areas. What can ASP learn from this? First of all, we should keep building upon strong formal foundations, just as SAT and DLs do. However, ASP should gear its research vision towards application scenarios in order to make ASP technology more efficient, more robust, more versatile, and in the end ready for real applications. This orientation is such a fruitful approach, being full of interesting and often fundamental research questions.

Second, we have to foster the dissemination of ASP in order to increase its perception. Apart from promoting ASP in our academic and industrial environment, teaching ASP is an important means to enhance the common awareness of it. This does not necessarily mean to teach full-fledged ASP courses, which is difficult in view of many encrusted curricula, but rather to incorporate ASP in AI-related classes as a tool for illustrating typical reasoning patterns in Knowledge Representation and Reasoning, like closed-world reasoning, abduction, planning, etc. And after all, to put it in Ray Reiter's words, it's the ASP community's duty to show "what on earth this stuff is good for."

ASP has staked its claim in being an attractive approach to declarative problem solving in combing an expressive modelling language with efficient solving technology. But how does it scale? In fact, this is not only a matter of performance but also of applicability and usability. Here is my personal view.

Performance Modern ASP solvers are based on advanced Boolean constraint technology and exhibit a similar performance as advanced SAT solvers [38]. Unlike SAT, however, ASP offers a uniform modelling language admitting variables. In fact, grounding non-propositional logical specifications constitutes a major bottleneck

in both ASP and SAT.[5] While this problem is addressed in SAT anew for each application, ASP centralizes this task in its grounders. The more surprising it is that there is so little work devoted to grounding within the ASP community (cf. [39–42]). This is a central yet neglected research topic. Apart from increasing research efforts in grounding, another major research theme is the development of program optimizers. That is, systems that transform highly declarative logic programs into equivalent ones, for which the overall solving time is significantly shorter than for the original program. In view of the vast literature on program equivalence in ASP (cf. [43–47]) the field appears to be well farmed for this endeavor.

Usability At first, many people are impressed by the ease of modelling in ASP. However, once they attack the first more complex problem and draft their first buggy encoding, they become often lost in the flat of declarativity. The solving process is completely transparent. No handle is available for finding out why the wrong or no solution is obtained. Also, when performance matters, it is still an art to come up with an efficient encoding, and often the result trades off declarativity. What is needed are dedicated tools, techniques, and methodologies to facilitate the development of answer set programs. In a nutshell, we need Software Engineering capacities that are adept enough to match ASP's high level of declarativity. First work on this can be found in [48–51] but much more work is needed.

Applicability Many practical applications of ASP motivate extensions or combinations with other problem-solving paradigms. Looking at SAT's breakthrough in planning and model checking [52, 36], it is interesting to observe that both involved dealing with an increasing bound on the solution size. Meanwhile dedicated SAT solvers allow for addressing this issue [53, 54]. Also, a whole sub-area of SAT, known as SAT modulo theories, deals with the integration of other problem-solving paradigms. So far, ASP is making only modest steps in similar directions. For instance, a first approach to incremental ASP solving is described in [55] and the combination of ASP and Constraint Processing is explored in [56–58]. More cross-fertilization with neighboring fields is needed to tackle real applications.

Last but not least, we have to foster the exchange within the ASP community as well as to neighboring fields like Constraint Processing and SAT and moreover re-enforce the link to ASP's parental research areas, Logic Programming and NMR. We need to improve the inter-operability of our systems and tools through specifying interfaces and fixing some standards. We need common benchmark and problem repositories and encourage comprehensive system competitions going beyond specific declarative solving paradigms. Otherwise, I am afraid that we will never turn our beef into a steak!

Acknowledgements I would like to thank Gerd Brewka, Martin Gebser, Mirosław Truszczyński, and Stefan Woltran for fruitful comments on an earlier draft.

References

1. Nebel, B., Rich, C., Swartout, W., eds.: Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR'92). Morgan Kaufman (1992)

[5] Interestingly, in SAT, recourse to first-order theorem proving seems not an option because of the high performance of Boolean constraint technology.

2. Ginsberg, M., ed.: Readings in Nonmonotonic Reasoning. Morgan Kaufman (1987)
3. Reiter, R.: Twelve years of nonmonotonic reasoning research: Where (and what) is the beef? [1] 789
4. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In Kowalski, R., Bowen, K., eds.: Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88). The MIT Press (1988) 1070–1080
5. Reiter, R.: A logic for default reasoning. Artificial Intelligence **13**(1-2) (1980) 81–132
6. Marek, V., Truszczyński, M.: Stable semantics for logic programs and default theories. In Lusk, E., Overbeek, R., eds.: Proceedings of the North American Conference on Logic Programing. The MIT Press (1989) 243–256
7. Bidoit, N., Froidevaux, C.: General logical databases and programs: Default logic semantics and stratification. Information and Computation **91**(1) (1991) 15–54
8. Niemelä, I., Simons, P.: Evaluating an algorithm for default reasoning. In: Working Notes of the IJCAI'95 Workshop on Applications and Implementations of Nonmonotonic Reasoning Systems. (1995) 66–72
9. `http://www.tcs.hut.fi/Software/smodels`
10. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence **25**(3-4) (1999) 241–273
11. Marek, V., Truszczyński, M.: Stable models and an alternative logic programming paradigm. In Apt, K., Marek, W., Truszczyński, M., Warren, D., eds.: The Logic Programming Paradigm: a 25-Year Perspective. Springer (1999) 375–398
12. Lifschitz, V.: Answer set programming and plan generation. Artificial Intelligence **138**(1-2) (2002) 39–54
13. Gelfond, M., Leone, N.: Logic programming and knowledge representation — the A-Prolog perspective. Artificial Intelligence **138**(1-2) (2002) 3–38
14. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
15. Tran, N., Baral, C.: Reasoning about triggered actions in AnsProlog and its application to molecular interactions in cells. In Dubois, D., Welty, C., Williams, M., eds.: Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR'04). AAAI Press (2004) 554–564
16. Dworschak, S., Grell, S., Nikiforova, V., Schaub, T., Selbig, J.: Modeling biological networks by action languages via answer set programming. Constraints **13**(1-2) (2008) 21–65
17. Soininen, T., Niemelä, I.: Developing a declarative rule language for applications in product configuration. In Gupta, G., ed.: Proceedings of the First International Workshop on Practical Aspects of Declarative Languages (PADL'99). Springer (1999) 305–319
18. Leone, N., Greco, G., Ianni, G., Lio, V., Terracina, G., Eiter, T., Faber, W., Fink, M., Gottlob, G., Rosati, R., Lembo, D., Lenzerini, M., Ruzzi, M., Kalka, E., Nowicki, B., Staniszkis, W.: The INFOMIX system for advanced integration of incomplete and inconsistent data. In Özcan, F., ed.: Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'05). ACM Press (2005) 915–917
19. Eiter, T., Faber, W., Leone, N., Pfeifer, G.: The diagnosis frontend of the dlv system. AI Communications **12**(1-2) (1999) 99–111
20. Erdem, E., Wong, M.: Rectilinear Steiner tree construction using answer set programming. [59] 386–399
21. Beierle, C., Dusso, O., Kern-Isberner, G.: Using answer set programming for a decision support system. In Baral, C., Greco, G., Leone, N., Terracina, G., eds.: Proceedings of the Eighth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'05). Springer (2005) 374–378
22. Heljanko, K., Niemelä, I.: Bounded LTL model checking with stable models. Theory and Practice of Logic Programming **3**(4-5) (2003) 519–550

23. Kavanagh, J., Mitchell, D., Ternovska, E., Manuch, J., Zhao, X., Gupta, A.: Constructing Camin-Sokal phylogenies via answer set programming. In Hermann, M., Voronkov, A., eds.: Proceedings of the Thirteenth International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'06). Springer (2006) 452–466
24. Brooks, D., Erdem, E., Erdogan, S., Minett, J., Ringe, D.: Inferring phylogenetic trees using answer set programming. Journal of Automated Reasoning **39**(4) (2007) 471–511
25. Aiello, L., Massacci, F.: Verifying security protocols as planning in logic programming. ACM Transactions on Computational Logic **2**(4) (2001) 542–580
26. Nogueira, M., Balduccini, M., Gelfond, M., Watson, R., Barry, M.: An A-prolog decision support system for the space shuttle. In Ramakrishnan, I., ed.: Proceedings of the Third International Symposium on Practical Aspects of Declarative Languages (PADL'01). Springer (2001) 169–183
27. Balduccini, M., Gelfond, M.: Model-based reasoning for complex flight systems. In: Proceedings of the Fifth AIAA Conference on Aviation, Technology, Integration, and Operations (ATIO'05). (2005).
28. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence **138**(1-2) (2002) 181–234
29. Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Transactions on Computational Logic **7**(3) (2006) 499–562
30. Lin, F., Zhao, Y.: ASSAT: computing answer sets of a logic program by SAT solvers. Artificial Intelligence **157**(1-2) (2004) 115–137
31. Giunchiglia, E., Lierler, Y., Maratea, M.: Answer set programming based on propositional satisfiability. Journal of Automated Reasoning **36**(4) (2006) 345–377
32. Gebser, M., Kaufmann, B., Neumann, A., Schaub, T.: Conflict-driven answer set solving. In Veloso, M., ed.: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07). AAAI Press/The MIT Press (2007) 386–392
33. Brachman, R., Schmolze, J.: An overview of the KL-ONE knowledge representation system. Cognitive Science **9**(2) (1985) 189–192
34. Davis, M., Putnam, H.: A computing procedure for quantification theory. Journal of the ACM **7** (1960) 201–215
35. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Communications of the ACM **5** (1962) 394–397
36. Clarke, E., Biere, A., Raimi, R., Zhu, Y.: Bounded model checking using satisfiability solving. Formal Methods in System Design **19**(1) (2001) 7–34
37. Baader, F., Horrocks, I., Sattler, U.: Description logics as ontology languages for the semantic web. In Hutter, D., Stephan, W., eds.: Mechanizing Mathematical Reasoning. Springer (2005) 228–248
38. Gomes, C., Kautz, H., Sabharwal, A., Selman, B.: Satisfiability solvers. In Lifschitz, V., van Hermelen, F., Porter, B., eds.: Handbook of Knowledge Representation. Elsevier (2008)
39. Syrjänen, T.: Omega-restricted logic programs. In Eiter, T., Faber, W., Truszczyński, M., eds.: Proceedings of the Sixth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'01). Springer (2001) 267–279
40. Syrjänen, T.: Cardinality constraint programs. In Alferes, J., Leite, J., eds.: Proceedings of the Nineth European Conference of Logics in Artificial Intelligence (JELIA'04). Springer (187-199)
41. Leone, N., Perri, S., Scarcello, F.: Backjumping techniques for rules instantiation in the DLV system. In Delgrande, J., Schaub, T., eds.: Proceedings of the Tenth International Workshop on Nonmonotonic Reasoning (NMR'04). (2004) 258–266
42. Gebser, M., Schaub, T., Thiele, S.: GrinGo: A new grounder for answer set programming. [60] 266–271

43. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic **2**(4) (2001) 526–541
44. Turner, H.: Strong equivalence made easy: nested expressions and weight constraints. Theory and Practice of Logic Programming **3**(4-5) (2003) 609–622
45. Eiter, T., Fink, M.: Uniform equivalence of logic programs under the stable model semantics. In Palamidessi, C., ed.: Proceedings of the Nineteenth International Conference on Logic Programming (ICLP'03). Springer (2003) 224–238
46. Eiter, T., Fink, M., Tompits, H., Woltran, S.: Simplifying logic programs under uniform and strong equivalence. In Lifschitz, V., Niemelä, I., eds.: Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'04). Springer (2004) 87–99
47. Oikarinen, E., Janhunen, T.: Modular equivalence for normal logic programs. In Brewka, G., Coradeschi, S., Perini, A., Traverso, P., eds.: Proceedings of the Seventeenth European Conference on Artificial Intelligence (ECAI'06). IOS Press (2006) 412–416
48. Brain, M., de Vos, M.: Debugging logic programs under the answer set semantics. In de Vos, M., Provetti, A., eds.: Proceedings of the Third International Workshop on Answer Set Programming (ASP'05). CEUR Workshop Proceedings (CEUR-WS.org) (2005) 141–152
49. Syrjänen, T.: Debugging inconsistent answer set programs. In Dix, J., Hunter, A., eds.: Proceedings of the Eleventh International Workshop on Nonmonotonic Reasoning (NMR'06). Clausthal University of Technology, Institute for Informatics (2006) 77–83
50. Pontelli, E., Son, T.: Justifications for logic programs under answer set semantics. In Etalle, S., Truszczyński, M., eds.: Proceedings of the Twenty-second International Conference on Logic Programming (ICLP'06). Springer (2006)
51. Brain, M., Gebser, M., Pührer, J., Schaub, T., Tompits, H., Woltran, S.: Debugging ASP programs by means of ASP. [60] 31–43
52. Kautz, H., Selman, B.: Planning as satisfiability. In Neumann, B., ed.: Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92), Wiley (1992) 359–363
53. Eén, N., Sörensson, N.: Temporal induction by incremental SAT solving. Electronic Notes in Theoretical Computer Science **89**(4) (2003)
54. Claessen, K., Sörensson, N.: New techniques that improve MACE-style finite model finding. In Baumgartner, P., Fermüller, C., eds.: Proceedings of the Workshop on Model Computation — Principles, Algorithms, Applications. (2003)
55. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Thiele, S.: Engineering an incremental ASP solver. In Dovier, A., Garcia de la Banda, M., Pontelli, E., eds.: Proceedings of the Twenty-fourth International Conference on Logic Programming (ICLP'08). (2008) To appear.
56. Elkabani, I., Pontelli, E., Son, T.: Smodels with CLP and its applications: A simple and effective approach to aggregates in ASP. [59] 73–89
57. Baselice, S., Bonatti, P., Gelfond, M.: Towards an integration of answer set and constraint solving. In Gabbrielli, M., Gupta, G., eds.: Proceedings of the Twenty-first International Conference on Logic Programming (ICLP'05). Springer (2005) 52–66
58. Mellarkod, V., Gelfond, M.: Integrating answer set reasoning with constraint solving techniques. In Garrigue, J., Hermenegildo, M., eds.: Proceedings of the Ninth International Symposium of Functional and Logic Programming. Springer (2008) 15–31
59. Demoen, B., Lifschitz, V., eds.: Proceedings of the Twentieth International Conference on Logic Programming (ICLP'04). Springer (2004)
60. Baral, C., Brewka, G., Schlipf, J., eds.: Proceedings of the Ninth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07). Springer (2007)