TLP **3** (4 & 5): 387–391, 2003.
DOI: 10.1017/S1471068403001820 Printed in the United Kingdom

Introduction to the special issue on Programming with answer sets

CHITTA BARAL

Department of Computer Science & Engineering, Arizona State University, Tempe, AZ 85287, USA
(*e-mail:* chitta@asu.edu)

ALESSANDRO PROVETTI

Department of Physics – Computer Science Section, University of Messina, Messina, I-98166 Italy
(*e-mail:* ale@unime.it)

TRAN CAO SON

Computer Science Department, New Mexico State University, Las Cruces, NM, USA
(*e-mail:* tson@cs.nmsu.edu)

Introduction

The search for an appropriate characterization of negation as failure in logic programs in the mid 1980s led to several proposals. Amongst them the stable model semantics – later referred to as *answer set semantics*, and the well-founded semantics are the most popular and widely referred ones. According to the latest (September 2002) list of most cited source documents in the CiteSeer database (http://citeseer.nj.nec.com) the original stable model semantics paper (Gelfond and Lifschitz, 1988) is ranked 10th with 649 citations and the well-founded semantics paper (Van Gelder *et al*., 1991) is ranked 70th with 306 citations. Since 1988 – when stable models semantics was proposed – there has been a large body of work centered around logic programs with answer set semantics covering topics such as: systematic program development, systematic program analysis, knowledge representation, declarative problem solving, answer set computing algorithms, complexity and expressiveness, answer set computing systems, relation with other non-monotonic and knowledge representation formalisms, and applications to various tasks.

This large body of building-block results makes logic programming with answer sets (sometimes called A-Prolog or AnsProlog) an attractive and suitable language for declarative programming, knowledge representation, and further development. In this direction, the new millennium has seen an AAAI Symposium (Spring 2001), a Dagstuhl seminar (in 2002), and the publication of the first textbook on the topic (Baral, 2003). This special issues follows those events.

The origin: AAAI Spring 2001 symposium

This special issue is, to some extent, consequential to the AAAI Spring 2001 Symposium: *ASP2001: Answer Set Programming: Towards Efficient and Scalable Knowledge Representation and Reasoning*. It was held in Stanford University on March 26-28 2001 and it was the first event ever dedicated to Answer set programming. We believe that it was a very successful event, with a turn out of about 50. While the majority of the participants were academics and students working on logic programming, several where from Silicon Valley industries or academics from neighboring areas. The invited speaker, Bart Selman from Cornell University, gave a survey on satisfiability techniques and algorithms. Overall, 32 articles were presented, some as plenary presentations and some as posters. We felt that the poster session was particularly successful in making attendees aware of each other's research, ask questions and sometimes offer hints for improvement. The idea of the special issue grew out of that experience. Contributions were solicited with an open call in summer 2001, and 15 articles were submitted. All the articles were very good, and due to page limitations and time constraints (limiting another revision based on referee's suggestions) we could only accommodate eight articles in this volume.

Programming with answer sets and answer set programming

Programming with answer sets refers to the use of stable model semantics (Gelfond and Lifschitz, 1988), which gives a declarative meaning to negation-as-failure and establishes a very direct connection with Reiter's Default logic and other relevant non-monotonic reasoning formalisms. The crux of the stable model (answer set) semantics is to consider rules (elsewhere called extended clauses) of a logic program Π as if they were default inference rules. Then, an interpretation S of Π is an answer set of Π if it is the minimal model of the Gelfond–Lifschitz transformation Π^S of Π. Answer sets of a program Π are *closed* under the rules of Π and each element of S is *supported* by at least one rule in Π.

The following short programs illustrate how the answer set semantics works. The program $\{p \leftarrow p.\}$ has $\emptyset$ as the only answer set, since $\{p\}$ although closed is not minimal. The program $\{p \leftarrow not\ q.\}$ has the only answer set $\{p\}$, whereas the interpretation $\{q\}$ although closed and minimal, is not supported. The program $\{p \leftarrow not\ q.\ q \leftarrow not\ p\}$ has two answer sets $\{p\}$ and $\{q\}$. The presence of two answer sets represents a non-deterministic choice between p and q. Finally, the program $\{p \leftarrow not\ p, a.\ a \leftarrow .\}$ admits no answer set. It is supposed to represent contradictory information: the first rule saying that a should result in contradiction, while the second rule asserts a.

We can see two main differences between answer sets and many other logic programming semantics such as the Well-founded semantics. First, the Well-founded semantics (and many others), assign to a program one and exactly one model. The Answer Set semantics, in contrast, can assign zero, one or several models to a program. This choice may seem unusual from the point of view of programming, since one thinks of the programmer as having in mind exactly one execution path.

https://doi.org/10.1017/S1471068403001820 Published online by Cambridge University Press

However, it seems particularly adequate from the point of view of Knowledge Representation. Inconsistency (zero answer sets) describes a partial, conflicting state of knowledge; one model describes total knowledge while a plurality of models describes alternative beliefs stemming from partial knowledge. Depending on the application, one may want to compute one or more answer sets.

Answer Set Programming (ASP) is a particular style of programming with answer sets where solutions to a problem are represented by answer sets (sets of atoms), and not by answer substitutions produced in response to a query, like in traditional logic programming. Although this might seem to be a drawback if one wants to query the program in the style of Prolog, i.e. ask about truth of a particular atom, since one or all models (depending on the type of query) need to be completely computed, the use of data base techniques such as magic sets can potentially overcome the drawbacks. Current implementations of ASP tend to compute models in a bottom-up fashion. As a result, true function symbols are disallowed, else models could be infinite and impossible to compute. However, one could still write a program where function symbols are allowed a limited number of nestings and then feed it to a grounder that substitutes them for constants.

To conclude this presentation, we like to mention the very strong implementations that are now available for Answer Set Programming. Thanks to them we can now experiment and apply ASP to an increasing number of problems and allow ASP technology to be *used*. Since complexity of finding answer sets is at least NP, scalability is an important factor and, in the end, one of the keys to industrial application. In the recent years ASP solvers have become more and more scalable, and are now a viable alternative to SAT solvers. Without pretense of completeness, we would like to mention the *smodels* solver and its companion parser *lparse* which have been developed at Helsinki University of Technology (Niemelä and Simons, 1997), and *DLV* developed for Disjunctive Logic Programs (Eiter *et al.*, 2000) at the Vienna Technical University and University of Calabria; those are two key systems that helped advance the field. New, promising solvers are being developed by Truszczyński *et al.* at the University of Kentucky, by Linke *et al.* at the University of Potsdam, by Lifschitz *et al.* at the University of Texas at Austin, by Gelfond *et al.* at Texas Tech University, and by Lin *et al.* at the Hong Kong University of Science and Technology.

Overview of the contributions

This special issue contains eight papers whose focus range from theoretical characterizations based on answer set semantics to application of answer set programming to various problems.

The paper by Arenas, Bertossi and Chomicki addresses the problem of retrieving consistent information when general first order queries are posed to an inconsistent relational database. Disjunctive logic programs with exceptions are used to specify database repairs in inconsistent instances. Answer set programming is used to compute the consistent answers.

https://doi.org/10.1017/S1471068403001820 Published online by Cambridge University Press

The paper by Balduccini and Gelfond shows how answer set programming can be applied in dynamic diagnosis. It describes an architecture for a software agent that operates a device, can observe the environment, tests and repairs the device's components. It shows that several tasks of this agent can be reduced to computing answer sets of logic programs.

The paper by Eiter, Faber, Leone and Pfeifer presents a meta-interpreter for computing answer sets of logic programs with preferences. Different approaches to dealing with preferences between rules are discussed and implemented. The paper also highlights the elegance and ease of using answer set programming in the development of prototypes for knowledge representation formalisms.

Erdem and Lifschitz generalize the notion of *tightness,* introduced by Fages to characterize logic programs with stable models, to programs with nested expressions. In particular, they show that the result proved by Fages can be extended to the new class of Tight Logic Programs.

Heljanko and Niemelä present another interesting application of answer set programming. They show how bounded model checking of asynchronous concurrent systems, exemplified by 1-safe Petri net, can be done using answer set programming. The main idea of this paper is to translate the network and a requirement on its behavior into a logic program whose answer sets correspond to solutions of the bounded model checking problem for the net. This approach allows for a compact representation of the problem.

Marek and Remmel address a fundamental question of answer set programming by showing that all search problems in the class **NP** can be solved using answer set programming. This helps to distinguish between answer set programming (answer set solvers) from traditional PROLOG technology.

Schaub and Wang develop a semantic framework for preference handling in answer set programming. They introduce preference preserving consequence operators and provide different characterizations of preferred answer sets for prioritized logic programs. This allows for a better comparison between several approaches to dealing with preferences in logic programming.

Turner provides a direct characterization of strong equivalence of logic programs with nested expression and weight constraints. It defines a notion, called SE-model, for nested programs, and shows that two nested programs are strong equivalent if and only if they have the same SE-models. This result is then extended to cover programs with weight constraints.

Acknowledgements

We would like to take this opportunity to sincerely thank the referees of the articles that were submitted for this special issue. Without their dedication and timely reviewing this volume would not have been possible. We would also like to thank the NSF grants 0070463, and 0126294 for supporting Chitta Baral during the period this volume was edited. Chitta Baral and Tran Cao Son were supported by NASA grant NCC2-1232. Alessandro Provetti was supported by MIUR COFIN project *Aggregate- and number-reasoning for computing: from decision algorithms to*

constraint programming with multisets, sets, and maps. Tran Cao Son would also like to acknowledge the support of NSF grants EIA-0130887 and EIA-0220590.

References

Baral, C. 2003. *Knowledge representation, reasoning and declarative problem solving*. Cambridge University Press.

Eiter, T., Faber, W., Leone, N. and Pfeifer, G. 2000. Declarative problem solving in dlv. In: J. Minker (Ed.), *Logic Based Artificial Intelligence*, pp. 79–103. Kluwer Academic.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In: R. Kowalski and K. Bowen (Eds.), *Logic Programming: Proceedings Fifth International Conference and Symposium*, pp. 1070–1080. MIT Press.

Niemelä, I. and Simons, P. 1997. Smodels – an implementation of the stable model and well-founded semantics for normal logic programs. In: J. Dix, U. Furbach and A. Nerode (Eds.), *Proceedings 4th International Conference on Logic Programming and Non-monotonic Reasoning*, pp. 420–429. Springer.

Van Gelder, A., Ross, K. and Schlipf, J. 1991. The well-founded semantics for general logic programs. *Journal of ACM 38,* 3, 620–650.