The Answer Set Programming Competition

Francesco Calimeri
Giovambattista Ianni
Thomas Krennwallner
Francesco Ricca

{calimeri,ianni,ricca}@mat.unical.it
tkren@kr.tuwien.ac.at

September 20, 2012

Abstract

The Answer Set Programming (ASP) Competition is a biannual event for evaluating declarative knowledge representation systems on hard and demanding AI problems. The competition consists of two main tracks: the ASP System Track and the Model & Solve Track. The traditional System Track compares dedicated answer set solvers on ASP benchmarks, while the Model & Solve Track invites any researcher and developer of declarative knowledge representation systems to participate in an open challenge for solving sophisticated AI problems with their tools of choice. This article provides an overview of the ASP Competition series, reviews its origins and history, giving insights on organizing and running such an elaborate event, and briefly discusses about the lessons learned so far.

1 A Brief History

Answer Set Programming (ASP) is a well-established paradigm of declarative programming with roots in the stable models semantics for logic programs (Gelfond and Lifschitz, 1991; Niemelä, 1999; Marek and Truszczyński, 1999). The main goal of ASP is to provide a versatile declarative modeling framework with many attractive characteristics. These features allow to turn—with little to no effort—problem statements of computationally hard problems into executable formal specifications, also called Answer Set Programs. These programs can be used to describe and reason over problems in a large variety of domains, such as commonsense and agent reasoning, diagnosis, deductive databases, planning, bioinformatics, scheduling and timetabling. See (Brewka et al., 2012) for an overview, while for introductory material on ASP, the reader might refer to (Baral, 2003; Eiter et al., 2009).

ASP has a close relationship to other declarative modeling paradigms and languages, such as SAT Solving, SAT Modulo Theories (SMT), Constraint Handling Rules (CHR), the Planning Domain Definition Language (PDDL), Automated Theorem Proving, and many others. All these formalisms have in common that they are built for solving demanding AI problems.[1]

In September 2002, participants of the Dagstuhl Seminar on Nonmonotonic Reasoning, Answer Set Programming and Constraints (Brewka et al., 2002) decided to establish an infrastructure for benchmarking ASP solvers (Borchert et al., 2004), following good practices already in place in neighboring fields of satisfiability testing and constraint programming, and with the explicit aim of fostering the development of ASP. A first informal competition took place during the workshop, featuring five systems: DLV, Smodels, ASSAT, Cmodels and Aspps, respectively from TU Vienna/U. of Calabria, Helsinki UT, Hong Kong UST, UT-Austin, and U. of Kentucky. Since then, after a second informal edition at the Dagstuhl Seminar in 2005, ASP systems compare themselves in the nowadays customary ASP Competition.

The 4th ASP Competition will be organized jointly by U. of Calabria, Italy, and TU Vienna, Austria, and will take place in the first half of 2013. Former ASP Competitions were held at U. of Potsdam, Germany (Gebser et al., 2007), at KU Leuven, Belgium (Denecker et al., 2009), and at U. of Calabria, Italy (Calimeri et al., 2012). The competition takes place biennially, and results are officially announced at the International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR).

2 The Competition

The ASP competition format is consolidated into two tracks: the *"Model & Solve"* Track, and the *"System"* Track. Participating systems are compared on a selected set of benchmarks, and scores are given to computational performance. The origin of these tracks and the evolution of the competition format is shown in Fig. 1.

[1] Concerning ASP, the comprehensive survey of Dantsin et al. (2001) gives an overview on complexity and expressiveness results for ASP and other formalisms related to logic programming.

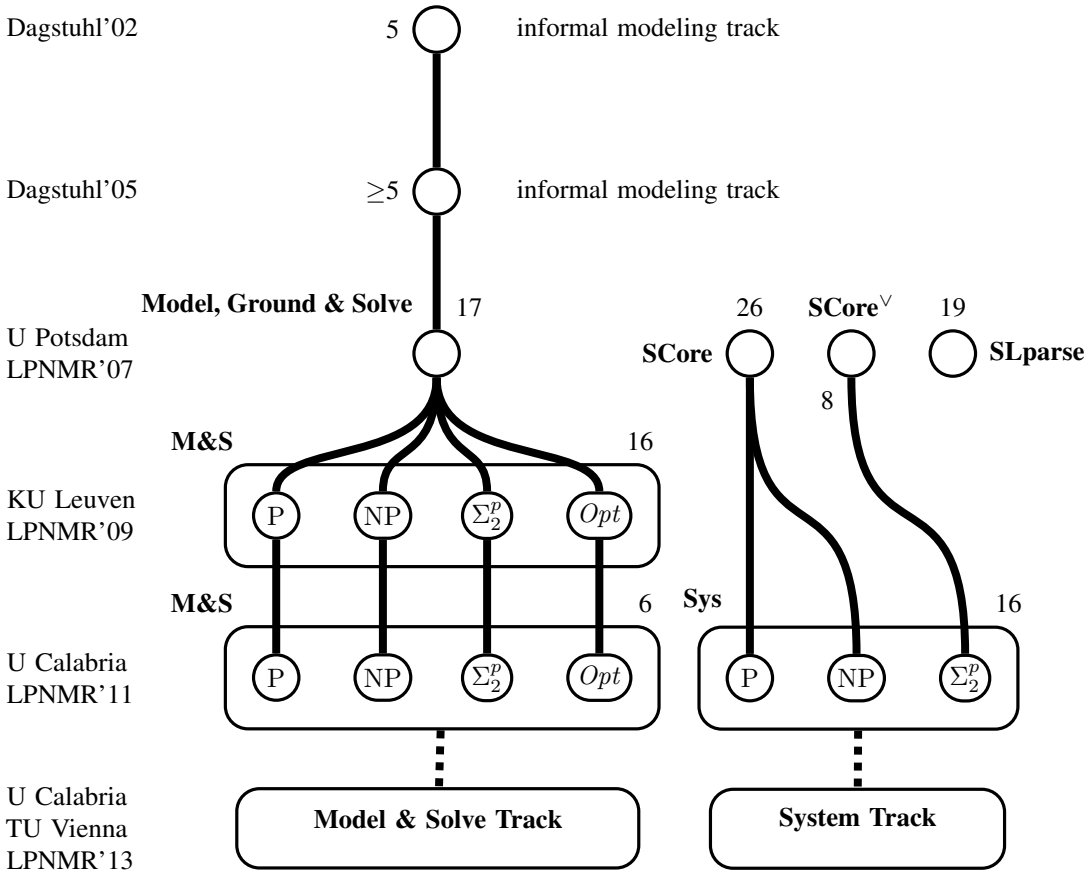

Figure 1: Evolution of the ASP Competition (numbers denote participant count in corresponding tracks)

The goal of the *"Model & Solve"* (M&S) Track is to integrate scientific communities and bring them closer together. This track is thus open to all types of solvers with declarative modeling capabilities: ASP systems, SAT solvers, SAT Modulo Theories solvers, Constraint Programming systems, automated theorem provers, Description Logics reasoners, planning reasoners, or any other.[2] It is worth noting that this track is not a programming contest: the focus is on comparing participant systems, and not on the ability of participant teams to model tricky problems. To this end, M&S competitors receive textual descriptions from a variety of problem domains, and each team has several months to produce working and efficient declarative specifications in case they are not available yet. Contestants are encouraged to exchange their solutions freely. For a particular benchmark domain, each team submits solutions in form of a domain specification together with a working system of choice, which can be configured on a per-benchmark basis.

The *"System"* Track is instead designed for problems modeled in a fixed language based on the answer set semantics, and formal domain specifications together with benchmark instances are provided by the organizers. The standard language for this track evolved from the core language specification drafted in 2004 (ASP Standardization Steering Committee, 2004)—called $SCore$ in the first competition in 2007—to ASP-Core, which will be extended for the forthcoming 4th ASP Competition 2013. The explicit purpose of this track is to foster language standardization and encourage the merging of various ASP dialects. Also, different from the M&S Track, each participating system must have a unique configuration for the whole class of benchmark problems, in order to compare off-the-shelf system performance.

Participants systems are compared on problems in which declarativity plays a central role. Take, as an example, the classic Towers of Hanoi planning prob-

[2]To avoid bias, comparisons are made in benchmark domains in which there is some degree of overlap in the application area of the systems at hand.

```
% read data: N is on M at time 0
on(0,M,N) :- on0(N,M).
onG(K,M,N) :- ongoal(N,M), steps(K).

% specify valid arrangements of disks:
% smaller disks are on larger ones
:- time(T), on(T,M,N), M >= N.

% specify a valid move (only for T < K)
% pick a disk to move
move(T,N) | noMove(T,N) :- disk(N), time(T),
                           steps(K), T < K.
:- move(T,N), move(T,M), N != M.
diskMoved(T) :- move(T,X).
:- time(T), steps(K), T < K, not diskMoved(T).

% pick a disk onto which to move
where(T,N) | noWhere(T,N) :- disk(N), time(T),
                             steps(K), T < K.
:- where(T,N), where(T,M), N != M.
diskWhere(T) :- where(T,X).
:- time(T), steps(K), T < K, not diskWhere(T).

% pegs 1..4 cannot be moved
:- move(T,N), N < 5.

% move only top-most discs
:- on(T,N,M), move(T,N).

% place disks on top only
:- on(T,N,M), where(T,N).

% no disk is moved in two consecutive moves
:- move(T,N), move(S,N), T = S - 1.

% specify effects of a move
on(S,M,N) :- move(T,N), where(T,M),
             S = T + 1.
on(S,N,M) :- time(T), steps(K), T < K,
             on(T,N,M), not move(T,M),
             S = T + 1.

% goal description
:- not on(K,N,M), onG(K,N,M), steps(K).
:- on(K,N,M), not onG(K,N,M), steps(K).

% solution: put disk N on top of M at step T
put(T,M,N) :- move(T,N), where(T,M),
              steps(K), T < K.
```

Figure 2: Towers of Hanoi with four pegs: ASP-Encoding from ASP Competition 2011

lem with three pegs and n disks. Initially, all disks are on the left-most peg. The goal of this problem is to move all n disks to the right-most peg by temporarily placing the disks on the other pegs, complying with the following rules: (i) move exactly one disk at a time; (ii) only the top-most disk on a peg can be moved; and (iii) larger disks cannot be placed on top of smaller ones. This problem has known optimal solution length, i.e., the plan of moving all n disks from the left-most peg to the right-most peg consists of $2^n - 1$ moves. The ASP competition uses a variant of this problem: instead of three, we consider four pegs with n disks. In this case, no known formula for the solution length exists, and no (proven) efficient algorithm for this kind of puzzle is known. The declarative specification of this problem, encoded in the ASP-Core language, is in Fig. 2. Essentially, the ASP input format is constituted by a set of (implicitly) universally quantified sentences, where syntax is mostly inherited from the Prolog language.

Declarative specifications are not limited to the standard ASP-Core language within the Model & Solve Track. In the ASP Competition 2011, each participating team provided its own custom specifications: solutions ranged from ASP-based (from the *aclasp* and *potassco* teams), constraint programming-based (from *bpsolver*, and *ezcsp*, a lightweight integration of ASP and Constraint Programming), planning-based (from *fastdownward*, a Planning Domain Definition Language solver), to first-order logic with inductive definitions (from the *idp* team, using the FO(ID) formalism).

Concerning evaluation techniques of ASP solvers, the state-of-the-art systems feature an input processing work-flow composed of a grounding module, generating a propositional theory, coupled with a subsequent propositional solver module (Brewka et al., 2012; Gebser et al., 2011; Faber et al., 2012; Alviano et al., 2011; Giunchiglia et al., 2006; Simons et al., 2002; Janhunen et al., 2009). The latter module generates an *answer set* according to the stable model semantics, or proves inconsistency. The ASP solvers that entered the *System Track* of the last edition belong to different categories depending on the inherent evaluation strategy of their solver module: *SAT-Based*, employing translation techniques to enforce correspondence between answer sets and satisfying assignments of SAT formulas, so that state-of-the-art SAT solvers can be used to compute answer sets; *Difference Logic-Based*, exploiting a translation from propositional ASP programs to Difference Logic theories, in order to perform the computation of answer sets via Satisfiability Modulo Theories solvers; and *Native ASP* solvers, which feature customized propositional search techniques often inspired by work done in the areas of constraint programming and SAT solving.

Essentially all systems competing in the first editions of the competition had international academic background, but recently industrial research teams such as Microsoft and Kodak have joined the game. The overall number of participants is following an

increasing trend with up to 22 participant systems in the 2011 edition. The events so far have shown a steady trend of performance improvement of the competitors: the winners of a competition outperform—usually by far—their predecessor's top ranking participants. Notably, variants of the `clasp` system (Gebser et al., 2012), which are developed by the winning team of the ASP Competition 2011, continue to perform excellently in related competitions such as the CADE Automated Theorem Proving System Competitions (CASC'11/'12), the Mancoosi International Solver Competition (MISC'11), the Pseudo-Boolean Competitions (PB'09/'11/'12), SAT competitions such as SAT'09/'11 and the SAT Challenge'12.

The benchmark suite of the competition is maintained and updated by the organizers, who choose benchmark instances from common planning domains, temporal and spatial scheduling problems, known combinatory puzzles, classic graph problems, and a number of industrial domains that are taken, e.g., from the database, information extraction, circuit layout, and natural sciences fields. Problems are classified into *Search*, *Query* and *Optimization*, and, according to the computational complexity of the underlying decision problem, into the categories *Polynomial*, *NP*, and *Beyond-NP*. The scoring system combines the number of solved instances, the running time performance, and the quality of the found solutions for optimization problems, and awards are assigned to the winners of each track and to the best performing participants of each sub-category. Technical details and more references concerning evaluation strategies, benchmarks, and scoring can be found in the report of the 3rd ASP Competition held in 2011 (Calimeri et al., 2012).

3 Lessons Learned

The Answer Set Programming community is following a steadily growing trend in terms of number of scientists, theoretical results and developed systems. Notably, ASP has recently appeared as a core technology in several industrial applications (Brewka et al., 2012; Grasso et al., 2011). A key outcome of the competition series is the standardization of the input format, which is now reaching maturity; the organization of each competition had been an opportunity for pushing language standardization within the community. A second noteworthy effect of the competition concerns the effort of attracting neighbor communities. A byproduct of the Model & Solve track is indeed a comparison across several axes: (a) among heterogeneous systems, even from different communities; (b) among participant systems, state-of-the-art solutions, and ad-hoc algorithms purposely chosen for selected problems; and (c) among purely declarative solutions and custom-tailored approaches. Concerning (b) and (c), purely declarative approaches have the reputation for non-optimal performance, but we learned that they are often very efficient, and sometimes outperform comparatively custom approaches and ad-hoc algorithms (Calimeri et al., 2012).

Comparison and interaction across communities are definitely required. One initiative that pushes into this directions is StarExec,[3] a cross community logic solving service. The ASP Competition chairs are part of the Advisory Committee of StarExec, whose aim is to provide the computational backbone for competitions in knowledge representation and set a common infrastructure for these. This way competitions and benchmarking will become easier to setup and make computational power more accessible.

A Glimpse at the Next Event. The 4th ASP Competition 2013 will again feature both the Model & Solve and the System Track. The submission deadline for the next ASP Competition is March 1st, 2013. Detailed regulations and further information can be found at http://aspcomp2013.mat.unical.it/.

Acknowledgements. This research has been supported by the Austrian Science Fund (FWF) projects P20841 & P24090, and the Regione Calabria and the EU under POR Calabria FESR 2007-2013 within the PIA project of DLVSystem s.r.l. The authors would like to thank all members of the Organization Committee of the 4th Answer Set Programming Competition 2013 as well as Carlos Linares López for useful comments on this report.

References

M. Alviano, F. Calimeri, W. Faber, N. Leone, and S. Perri. Unfounded sets and well-founded semantics of answer set programs with aggregates. *Journal of Artificial Intelligence Research*, 42:487–527, 2011.

ASP Standardization Steering Committee. Core language for ASP competitions, 2004. Logic Programming and Nonmonotonic Reasoning'04 Steering Committee Meeting. https://www.mat.unical.it/aspcomp2011/files/Corelang2004.pdf.

[3] http://www.starexec.org

C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003. ISBN 0-52181802-8.

P. Borchert, C. Anger, T. Schaub, and M. Truszczyński. Towards Systematic Benchmarking in Answer Set Programming: The Dagstuhl Initiative. In *Logic Programming and Nonmonotonic Reasoning'04*, pp. 3–7. Springer, 2004.

G. Brewka, I. Niemelä, T. Schaub, and M. Truszczyński. Dagstuhl Seminar Nr. 0238, Nonmonotonic Reasoning, Answer Set Programming and Constraints, Sept. 2002. http://www.dagstuhl.de/02381.

G. Brewka, T. Eiter, and M. Truszczyński. Answer set programming at a glance. *Communications of the ACM*, 54(12):92–103, 2012. doi: 10.1145/2043174.2043195.

F. Calimeri, G. Ianni, and F. Ricca. The third open Answer Set Programming Competition. *Theory and Practice of Logic Programming*, 2012. To appear.

E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and Expressive Power of Logic Programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

M. Denecker, J. Vennekens, S. Bond, M. Gebser, and M. Truszczyński. The Second Answer Set Programming Competition. In *Logic Programming and Nonmonotonic Reasoning'09*, pp. 637–654. Springer, 2009.

T. Eiter, G. Ianni, and T. Krennwallner. Answer Set Programming: A Primer. In *Reasoning Web'09*, pp. 40–110, Springer, 2009.

W. Faber, N. Leone, and S. Perri. The Intelligent Grounder of DLV. In *Correct Reasoning*, vol. 7265 of *LNCS*, pp. 247–264. Springer, 2012.

M. Gebser, L. Liu, G. Namasivayam, A. Neumann, T. Schaub, and M. Truszczyński. The first answer set programming system competition. In *Logic Programming and Nonmonotonic Reasoning'07*, pp. 3–17, Springer, 2007.

M. Gebser, B. Kaufmann, R. Kaminski, M. Ostrowski, T. Schaub, and M. Schneider. Potassco: The Potsdam Answer Set Solving Collection. *AI Communications*, 24(2):107–124, 2011.

M. Gebser, B. Kaufmann, and T. Schaub. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, 187–188:52–89, 2012.

M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.

E. Giunchiglia, Y. Lierler, and M. Maratea. Answer Set Programming Based on Propositional Satisfiability. *Journal of Automated Reasoning*, 36(4): 345–377, 2006.

G. Grasso, N. Leone, M. Manna, and F. Ricca. ASP at work: Spin-off and applications of the DLV system. In *Logic Programming and Nonmonotonic Reasoning'11*, pp. 432–451, 2011.

T. Janhunen, I. Niemelä, and M. Sevalnev. Computing Stable Models via Reductions to Difference Logic. In *Logic Programming and Nonmonotonic Reasoning'09*, pp. 142–154. Springer, 2009.

V. W. Marek and M. Truszczyński. Stable Models and an Alternative Logic Programming Paradigm. In *The Logic Programming Paradigm – A 25-Year Perspective*, pp. 375–398. Springer, 1999.

I. Niemelä. Logic Programming with Stable Model Semantics as Constraint Programming Paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3–4):241–273, 1999.

P. Simons, I. Niemelä, and T. Soininen. Extending and Implementing the Stable Model Semantics. *Artificial Intelligence*, 138:181–234, June 2002.