The Fourth Answer Set Programming Competition: Preliminary Report*

Mario Alviano[1], Francesco Calimeri[1], Günther Charwat[2], Minh Dao-Tran[2], Carmine Dodaro[1], Giovambattista Ianni[1], Thomas Krennwallner[2], Martin Kronegger[2], Johannes Oetsch[2], Andreas Pfandler[2], Jörg Pührer[2], Christoph Redl[2], Francesco Ricca[1], Patrik Schneider[2], Martin Schwengerer[2], Lara Katharina Spendier[3], Johannes Peter Wallner[2], and Guohui Xiao[2]

[1] Dipartimento di Matematica e Informatica, Università della Calabria, Italy
[2] Institute of Information Systems, Vienna University of Technology, Austria
[3] Institute of Computer Languages, Vienna University of Technology, Austria

Abstract. Answer Set Programming is a well-established paradigm of declarative programming in close relationship with other declarative formalisms such as SAT Modulo Theories, Constraint Handling Rules, PDDL and many others. Since its first informal editions, ASP systems are compared in the nowadays customary ASP Competition. The fourth ASP Competition, held in 2012/2013, is the sequel to previous editions and it was jointly organized by University of Calabria (Italy) and the Vienna University of Technology (Austria). Participants competed on a selected collection of benchmark problems, taken from a variety of research areas and real world applications. The Competition featured two tracks: the Model& Solve Track, held on an open problem encoding, on an open language basis, and open to any kind of system based on a declarative specification paradigm; and the System Track, held on the basis of fixed, public problem encodings, written in a standard ASP language.

1 Introduction

Answer Set Programming is a declarative approach to knowledge representation and programming proposed in the area of nonmonotonic reasoning and logic programming [9, 11, 23–25, 35, 36, 43, 46]. Among the advantages of ASP are its declarative nature combined with a comparatively high expressive power [19, 42]. After pioneering work [10, 42, 49, 50], several systems supporting ASP and its variants are born from the initial offspring [2, 3, 16, 18, 31, 33, 37, 39–42, 44, 45, 47, 49, 52].

Since the first informal editions (Dagstuhl 2002 and 2005), ASP systems are compared in the nowadays customary ASP Competition series [20, 16, 34], which reached now its fourth official edition. The Fourth ASP Competition featured two tracks: the Model& Solve Track, held on an open problem encoding, open language basis, and

* This research is supported by the Austrian Science Fund (FWF) projects P20841 and P24090. Carmine Dodaro is partly supported by the European Commission, European Social Fund and Regione Calabria.

P. Cabalar and T.C. Son (Eds.): LPNMR 2013, LNAI 8148, pp. 42–53, 2013.

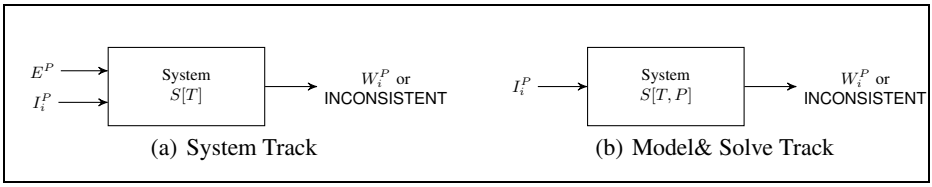

Fig. 1. Competition Setting

open to any system based on a declarative specification paradigm; and the System Track, held on the basis of fixed problem encodings, written in a standard ASP language.

In this paper we illustrate the overall setting of the fourth ASP Competition, its participants and the benchmark suite. A more detailed report, including a complete description of the entire Competition, outcomes of non-participant systems, and comparisons with other state-of-the-art systems is under preparation. The competition had 23 participants which were evaluated on a suite of 27 benchmark domains, for each of which about 30 instances were selected, for a total of about 50'000 separate benchmark runs. Results of the competition were disclosed during the LPNMR 2013 conference.

The remainder of this paper is structured as follows. In Section 2 we illustrate the competition format, especially discussing updates which were introduced with respect to the previous editions. In section 3 we illustrate the new standard language ASP-Core-2. Section 4 illustrates the benchmark problems used in this edition and Section 5 presents the participants to the Competition.

2 Format of the Fourth ASP Competition

We illustrate here the settings of the competition focusing on changes introduced with respect to the Third Competition's edition.

Competition format. The 4th ASP Competition retains the distinction between Model& Solve and System Track. Both tracks run on a selected suite of benchmark domains, which were chosen during an open *Call for Problems* stage.

The System Track was conceived with the aim of *(i)* fostering the standardization of the ASP input language, and *(ii)* let the competitors compare each other in fixed, predefined conditions, excluding e.g., domain-tailored evaluation heuristics and custom problem encodings. The System Track is run as follows (Figure 1-a): for each problem P a corresponding, fixed, declarative specification E^P of P, and a number of instances $I_1^P, \ldots, I_n^P$, are given. Each participant system $S[T]$, for T a participating team, is fed with all the couples $\langle E^P, I_i^P \rangle$, and challenged to produce a *witness* solution to $\langle E^P, I_i^P \rangle$ (denoted by W_i^P) or to report that no witness exist, within a predefined amount of allowed time. A score is awarded to each $S[T]$ per each benchmark, as detailed later in this Section. Importantly, problem encodings were fixed for all participants: specialized solutions on a per-problem basis were not allowed, and problems were specified in the recently-released ASP-Core-2 language. This setting has been introduced in order to give a fair, objective measure of what one can expect when switching from a system to

another, while keeping all other conditions fixed, such as the problem encoding and the default solver settings and heuristics.

Differently from the System Track, the Model& Solve Track has been instead left open to any (bundle of) solver systems loosely based on a declarative specification language. Thus no constraints were set on the declarative language used for encoding solutions to be solved by participants' systems. Indeed, the spirit of this Track is to (i) encourage the development of new expressive declarative constructs and/or new modeling paradigms; (ii) to foster the exchange of ideas between communities in close relationships with ASP; (iii) and, to stimulate the development of new ad-hoc solving methods, refined problem specifications and solving heuristics, on a per benchmark domain basis.

In more detail, each participant team T was allowed to present a version $S[T, P]$ of their system(s) possibly customized for each problem domain P in terms of solving heuristics and declarative problem specification. Each system $S[T, P]$, for T a participating team, is challenged to solve some instances of problem P. $S[T, P]$ is expected to produce, within a predefined amount of allowed time, a *witness* solution for each instance in input (or to report that no witness exists). For both tracks, a total score is awarded to each team T summing up the scores obtained by each $S[T, P]$ (or by $S[T]$) on each benchmark, as detailed below.

Scoring system. The competition scoring system was inherited from the third edition of the competition and improved in some specific aspects. In detail, each participant is awarded of a score per each benchmark P proportional to: *a)* the percentage of instances solved within time ($S_{solve}(P)$); *b)* the evaluation time ($S_{time}(P)$); and *c)* the quality of the computed solution in case of optimization problems ($S_{opt}(P)$).[1]

Comparing the scoring system with the one of the former edition, some adjustments were introduced to the logarithmic time scoring quota S_{time}, which has been redefined as follows:

$$S_{time}(P) = \frac{100 - \alpha}{N\gamma} \sum_{i=1}^{N} \left(1 - \left(\frac{\log(\max(1, t_i) + s)}{\log(t_{out} + s)}\right)\right)$$

where P is the problem domain at hand; t_{out} is the maximum allowed time; t_i the time spent by system S while solving instance i (t_i is assumed to be lesser or equal to t_{out}); s is a factor which mildens the logarithmic behavior of S_{time}; γ is a normalization factor (having an effect detailed below); and α is a percentage factor balancing the impact of $S_{time}(P)$ w.r.t. the $S_{solve}(P)$ quota. Indeed, $S_{solve}(P)$ assigns a score that is linearly proportional to the percentage of solved instances for P as follows:

$$S_{solve}(P) = \alpha \frac{N_P}{N}$$

where N_P is the number of instances of problem P solved by S within the timeout.

As in the third edition of the Competition, $S_{time}(P)$ is specified in order to take into account the "perceived" performance of a system according to a logarithmic scoring.

[1] Solution quality is intended in terms of normalized percent distance from the optimal solution.

Moreover, the parameters of $S_{time}(P)$ were set in order to obtain a reasonable behavior that is expected to be stable w.r.t. minor fluctuations in measured execution times. In particular, we set for this edition of the competition $s = 10$ (it was previously set to 1) to avoid excessive differences in scoring when solving time was below 10 seconds; also, the correction $\max(1, t_i)$ prevents any score difference at all when t_i is below 1 second. In this way there is basically no difference in assigned score when execution times are very low and close to the order of magnitude of measurement errors. As in the previous competition, α was set to 0.5, so that the time and the instance quota are evenly balanced; finally, γ was chosen in such a way that the time score quota awarded for solving a single instance i within the timeout (i.e., $0 \leq t_i \leq t_{out}$) is normalized in the range $[0, (100 - \alpha)/N]$, thus we set

$$\gamma = 1 - \frac{\log(1 + s)}{\log(t_{out} + s)}$$

Other improvements were made w.r.t. the scoring system employed in the 3rd edition, e.g., with the introduction of a formal averaging policy for coping with multiple runs of the benchmark suite and other minor refinements. The scoring system of the 4th ASP Competition is extensively described in [5].

Instance selection process. Concerning instance selection, we introduced in this edition an new method for the random selection of instances, by taking into account that (i) the selection process should depend on a unique, not controllable by the organizer, random seed value; (ii) instances should be roughly ordered by some difficulty criterion provided by domain maintainers; (iii) hash values of instance files, and the fixed ordering of instances should be known before the Competition run; (iv) the random sequence used for selection should be unique and applied systematically to each benchmark domain, i.e. it must be impossible in practice, for organizers, to possibly forge the selection of instances in one domain without altering, out of control, the selection of instances in the other domains. (v) the selection method should approximately select a set of instances with a good balance between "hard" and "easy" instances.

The above considerations led us to adopt a variant of the statistical systematic sampling technique for the instance selection process. In detail, let S be the a random seed value chosen from an objective random source, and R be the number of instances per benchmark to be selected. Let D a benchmark domain, L_D its list of available instances, made available from benchmark domain maintainers roughly sorted by difficulty level, with $|L_D| = N_D$. We denote as $L_D[i]$ the i-th instance. over the whole family L_D, as follows: let *Start*, $\mathit{Perturb}_1, \ldots, \mathit{Perturb}_R$ be values systematically generated from S where *Start* ranges from 0 to 1 and each $\mathit{Perturb}_i$ ranges from -1.5 to 1.5. Then, we set $\mathit{Step} = \frac{N_D}{R}$ and $\mathit{Start}_D = \mathit{Step} * \mathit{Start}$. Then we select, for all i $(1 \leq i \leq R)$, all the instances

$$L_D\big[\text{round}(\max(0, \min(N_D, \mathit{Start}_D + i * \mathit{Step} + \mathit{Perturb}_i)))\big]$$

Here round(n) is n rounded to the nearest integer. When $\mathit{Step} > \mathit{Perturb}_{i+1} - \mathit{Perturb}_i$ for some i, we conventionally selected $L_D[h + 1]$ as the $(i + 1)$-th instance, for h the index of the i-th instance.

Software and Hardware settings. The Competition has been run using the purposely developed VCWC environment (Versioning Competition Workflow Compiler) [17]. This tool takes as input the participating solvers and dedicated benchmark sets and generates a workflow description for executing all necessary (sub-)tasks for generating the final solver rankings and statistics. As jobs may fail during the execution, VCWC supports a gradual refinement of the competition workflow and allows to add or update solvers, instances, benchmarks, or further runs after the machinery has been brought up. Generated jobs where scheduled on the Competition hardware using the HTCondor [51] high throughput computing platform.

Concerning hardware, the competition has been run on several Ubuntu Server 12.04 LTS x86-64 machines featuring two AMD Opteron Magny-Cours 6176 SE CPUs (total of 24 cores) running at 2.3 GHz with 128GiB of physical RAM. To accommodate multi-core evaluations, runs were classified into sequential and parallel. Sequential runs have been evaluated in a single-core Linux control group, while parallel runs were limited to a six-core control group; all of the six cores form one NUMA node to prevent memory access overhead to remote NUMA nodes. For both kind of runs only memory with the lowest distance to the corresponding NUMA node has been used. The memory reserved to each control group was constrained to 6 GiB (1 GiB = 1 gibibyte = 2^{30} bytes), while the total CPU time available was 600 seconds. Competitors were instructed about how to reproduce the software environment, in order to properly prepare and test their systems.

Reproducibility of the results. The committee did not disclose any submitted material until the end of the Competition; nonetheless, willingly participants were allowed to share their own work at any moment. In order to guarantee transparency and reproducibility of the Competition results, all participants were asked to agree that any kind of submitted material (system binaries, scripts, problems encodings, etc.) was to be made public after the Competition.

3 Competition Language Overview

Since the first Edition of the competition, *standardization* has always been one of the main goals of the ASP Competition Series. The efforts to find a common language basis, started with the LPNMR 2004 language draft [6], and prosecuted with the ASP-Core [15] standard adopted in 3rd edition of the Competition. ASP-Core was published along with the ASP-RfC proposal, which preceded the work of the *ASP Standardization Working Group*, that produced the ASP-Core-2 standard, adopted for the System Track in the 4th edition of the Competition. The ideas that guided the work are in the trail of the latest version of the standard: to safeguard the original A-Prolog language [36]; to include, as extensions, a number of features both desirable and mature; and, eventually, to have a language with non-ambiguous semantics over which widespread consensus has been reached. The basis of ASP-Core-2 is hence a rule language allowing disjunctive heads and strong and negation-as-failure (NAF) negation, with no need for domain predicates. Arithmetic and Herbrand-interpreted functional terms are explicitly allowed, as well as aggregate literals and queries; choice atoms and weak constraints complete the list of new features.

The semantics of non-ground ASP-Core-2 programs extends the traditional notion of Herbrand interpretation. Concerning the semantics of propositional programs, it is based on [36], extended to aggregates according to [26]; choice atoms [49] are treated in terms of a proper translation. To promote declarative programming as well as practical system implementations, a number of restrictions are imposed. For instance, semantics is restricted to programs containing non-recursive aggregates; reasonable restrictions are applied for ensuring that function symbols, integers and arithmetic built-in predicates are finitely handled.

The ASP-Core-2 specification is rich in new features and is partially backward-compatible with older common input formats. Participants were thus allowed to join the System Track using slightly syntactically different problem encodings. Each statement of alternative problem encodings was kept in strict one-to-one correspondence with the reference ASP-Core-2 encoding.

The work on standardization is beyond the scope of the 4th ASP Competition, and new features (such as *maximize/minimize* statements for optimization, and more) have lately been incorporated into the standard. The detailed ASP-Core-2 language specification used for this Competition can be found at [12], while the ongoing standardization activity can be followed at [13].

4 Benchmark Suite

The benchmark suite has been constructed during a *Call for problems* stage, after which 26 benchmark domains were selected, 13 of which were confirmed from the previous edition. The whole collection was suitable for a proper ASP-Core-2 [12] specification. All 26 problems appeared in the System Track, while the Model& Solve Track featured only 15 domains. The complete list of benchmarks, whose details are available at [4], is reported in Table 1. Concerning legacy benchmark domains, problem maintainers were asked to produce refined specifications and/or better instances sets whenever necessary. The presence of a star (*) in the fourth column means that the corresponding problem was changed in its specifications w.r.t. its third Competition version. The selection criteria for problems aimed to collect a number of domains as balanced as possible in terms of (i) academic vs applicative provenance, (ii) computational complexity, type of domain and type of reasoning task, and (iii) research group provenance.

Problems belonged to a variety of areas, like general artificial intelligence, databases, formal logics, graph theory, planning, natural sciences and scheduling; in addition, the benchmark suite included a synthetic domain and some combinatorial and puzzle problems. Concerning the type of reasoning task to be executed in each domain, we kept the categorization in term of of *Search*, *Query* and *Optimization* problems[2].

Problems were further classified according to their computational complexity in the categories P (polynomially solvable), NP (NP-Hard), Beyond-NP (more than NP-Hard). Apart from this categorization, we classified in the Opt category (optimization problems) all the problems in which the minimization/maximization of a numerical goal function could be identified. The first three categories approximately reflect the

[2] The reader is referred to [14] for details concerning the three categories.

Table 1. 4th ASP Competition – Benchmark List

ID	Problem Name	Category	Domain	2011	M&S Track
N01	Permutation Pattern Matching	NP	Combinatorial	NO	YES
N02	Valves Location	Opt	Combinatorial	NO	YES
N04	Connected Maximum-density Still Life	Opt	AI	NO	YES
N05	Graceful Graphs	NP	Graph	NO	YES
N06	Bottle Filling	NP	Combinatorial	NO	YES
N07	Nomystery	NP	Planning	NO	YES
N08	Sokoban	NP	Planning	YES*	YES
N09	Ricochet Robot	NP	Puzzle	NO	YES
O10	Crossing Minimization	Opt	Graph	YES	YES
O11	Reachability	P	Graph	YES*	YES
O12	Strategic Companies	Σ_2^P	AI	YES*	YES
O13	Solitaire	NP	Puzzle	YES	YES
O14	Weighted Sequence	NP	Database	YES	YES
O15	Stable Marriage	P*	Graph	YES	YES
O16	Incremental Scheduling	NP	Scheduling	YES	YES
N17	Qualitative Spatial Temporal Reasoning	NP	Formal logic	NO	NO
N18	Chemical Classification	P*	Natural Sciences	NO	NO
N19	Abstract Dialectical Frameworks Well-founded Model	Opt	Formal logic	NO	NO
N20	Visit-all	NP	Planning	NO	NO
N21	Complex Optimization of Answer Sets	Σ_2^P	Synthetic	NO	NO
N22	Knight Tour with Holes	NP	Puzzle	YES*	NO
O23	Maximal Clique	Opt	Graph	YES	NO
O24	Labyrinth	NP	Puzzle	YES	NO
O25	Minimal Diagnosis	Σ_2^P	Diagnosis	YES	NO
O26	Hanoi Tower	NP	AI	YES	NO
O27	Graph Colouring	NP	Graph	YES	NO

data complexity [48] of the underlying decisional problem, with some exception. In particular, STABLE MARRIAGE [27, 38], for which polynomial algorithms are known, has been re-proposed in the System Track with a natural declarative encoding which makes usage of the Guess & Check paradigm; also, the CHEMICAL CLASSIFICATION benchmark featured sets of Horn rules as input instances, thus, strictly speaking, it is to be considered a NP problem under *combined* complexity. It is worth noting that the computational complexity of a problem has also impact on features of solvers which were put under testing. Polynomial problems are mostly, but not exclusively, useful for testing grounding modules, while the role of model generator modules is more prominent when benchmarking is done in domains in the NP category.

5 Participants

In this Section we briefly present all participants; we refer the reader to the official Competition website [14] for further details.

System Track Participants. The System Track of the Competition featured 16 systems; these can be roughly grouped into two main classes: (i) "native" systems, which exploit techniques purposely conceived/adapted for dealing with logic programs under the stable models semantics, and (ii) "translation-based" systems, which (roughly), at some stage of the evaluation process, produce an intermediate specification in a different formalism; such specification is then fed to an external solver. The first category includes `clasp` – and variants thereof – and DLV+*wasp*, while the second counts IDP3 (which is FO(.)-based), LP2BV-1 and LP2BV-2 (relying on SMT solvers), LP2MIP and LP2MIP-MT (relying on integer programming tools), and LPD2SAT, LP2SAT-MT and LP2SOLRED-MT (relying on SAT solvers). Interestingly, several parallel (multi-threaded) solutions are officially present in this edition of the Competition; such systems are denoted by means of the "-mt" suffix.

- The group from University of Potsdam presented a number of solvers. `clasp` [33] is an answer set solver for (extended) normal logic programs featuring state-of-the-art techniques from the area of Boolean constraint solving. `claspfolio` [29] chooses the best suited configuration of `clasp` to process the given input program, according to machine-learning techniques. `claspD-2` [31] is an extension of `clasp` that allows for solving disjunctive logic programs using a new approach to disjunctive ASP solving that aims at an equitable interplay between "generating" and "testing" solver units, and `claspD-2` is a version supporting the ASP-Core-2 standard [12]. Multi-threaded versions `clasp-mt` [30], `claspfolio-mt` and `claspD-2-mt` were also present.
- The research group from Aalto University presented different solvers, all of them working by means of translations. With LP2BV-1 and LP2BV-2 [47], a given ASP program is grounded by `Gringo`, simplified by Smodels, normalized by getting rid of extended rule types (e.g., choice rules), translated to bit vectors and finally solved by *BOOLECTOR* for LP2BV-1 and *Z3* for LP2BV-1. LP2SAT, LP2SAT-MT [39] and LP2SOLRED-MT [39, 52] work similarly, but rely on translations to SAT rather than bit vectors; *PRECOSAT*, *PLINGELING* and *GLUCORED* work under the hood for LP2SAT, LP2SAT-MT, and LP2SOLRED-MT, respectively. LP2MIP [45] and LP2MIP-MT, finally, translate to mixed integer programs, which are processed by *CPLEX*.
- The team from KU Leuven presented IDP3, using FO(ID,Agg) + Lua as input language [53]. Model generation/optimization was achieved by lifted unit propagation + grounding with bounds (possibly using XSB for evaluating definitions) and applying MiniSat(ID) as search algorithm.
- *wasp*+DLV. *wasp* [2] is a native ASP solver built upon a number of techniques originally introduced in SAT, which were extended and properly combined with techniques specifically defined for solving disjunctive ASP programs. Among them are restarts, constraints learning and backjumping. Grounding is carried out by an enhanced version of the DLV grounder able to cope with the ASP-Core-2 features. Team members were affiliated to the University of Calabria.

Model& Solve Track Participants. Seven teams participated to the Model& Solve Track, each presenting a custom approach, often explicitly differentiated depending on the domain problem at hand: short descriptions follow.

- B-Prolog [54] provides several tools for tackling combinatorial optimization problems, including tabling for dynamic programming problems, CLP(FD) for CSPs, and a compiler that translates CSPs into SAT.
- Enfragmo [1] is a grounding-based solver. From a given input (expressed in multi-sorted first order logic extended with arithmetic and aggregate operators) a propositional CNF formula is produced, representing the solutions to the instance, which is processed by a SAT solver.
- EZCSP [7, 8] freely combines different ASP (such as `Gringo`/`clasp`, `Clingo`, `Clingcon`, possibly extended for supporting non-Herbrand functions) and CP (such as B-Prolog) solvers to be selected according to the features of the target domain.
- IDP3 [53] is the same system participating in the System Track, with proper custom options depending on the benchmark problem; IDP2 [53] consists of the grounder GidL and the search algorithm MiniSat(ID).
- *inca* [21, 22] implements Constraint Answer Set Programming (CASP) via Lazy Nogood Generation (LNG) and a selection of translation techniques. It integrates `Gringo` (for grounding CASP specifications), `clasp`, and a small collection of constraint propagators enhanced with LNG capacities [22].
- The team of Potassco [32] used `Gringo` 3, `clasp` 2, and `iClingo` 3 (an incremental ASP system [28] implemented on top of `Clingo`, featuring a combined grounder and solver that keep previous states while increasing an incremental parameter, trying to avoid re-producing already computed knowledge). Search settings were manually chosen (w.r.t. few instances) per problem class.

Acknowledgments. All of us feel honored of the appointment of TU Vienna and University of Calabria as host institutions: we want to thank all the members of the Database and Artificial Intelligence Group, the Knowledge-based Systems Group, and the Theory and Logic Group of Vienna University of Technology (TU Vienna), as well as the Computer Science Group at the Department of Mathematics and Computer Science of University of Calabria (Unical) for their invaluable collaboration, which made this event possible. A special thanks goes to all the members of the ASP Standardization Working Group and to all the members of the scientific community which authored, reviewed and helped in setting up all problem domains; and, of course, to the participating teams, whose feedback, once again, significantly helped at improving competition rules and benchmark specifications. We also want to acknowledge Thomas Eiter as Head of the Institute for Information Systems of the Vienna University of Technology, and Nicola Leone as the Director of the Department of Mathematics and Computer Science of University of Calabria, which provided us with human and technical resources. Eventually, we want to give a special thank to Pedro Cabalar and Tran Cao Son for their support as LPNMR-2013 conference chairs and proceedings editors.

References

1. Aavani, A., Wu, X(N.), Tasharrofi, S., Ternovska, E., Mitchell, D.: Enfragmo: A system for modelling and solving search problems with logic. In: Bjørner, N., Voronkov, A. (eds.) LPAR-18 2012. LNCS, vol. 7180, pp. 15–22. Springer, Heidelberg (2012)
2. Alviano, M., Dodaro, C., Faber, W., Leone, N., Ricca, F.: WASP: A native ASP solver based on constraint learning. In: Cabalar, P., Corunna, Son, T.C. (eds.) LPNMR 2013. LNCS, vol. 8148, pp. 55–67. Springer, Heidelberg (2013)
3. Anger, C., Gebser, M., Linke, T., Neumann, A., Schaub, T.: The nomore++ Approach to Answer Set Solving. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS (LNAI), vol. 3835, pp. 95–109. Springer, Heidelberg (2005)
4. 4th ASP Competition Organizing Committee, T.: Official Problem Suite (2013), https://www.mat.unical.it/aspcomp2013/OfficialProblemSuite
5. 4th ASP Competition Organizing Committee, T.: Rules and Scoring (2013), https://www.mat.unical.it/aspcomp2013/ParticipationRules
6. Core language for ASP solver competitions, minutes of the steering committee meeting at LPNMR 2004 (2004), https://www.mat.unical.it/aspcomp2011/files/Corelang2004.pdf
7. Balduccini, M.: Representing Constraint Satisfaction Problems in Answer Set Programming. In: ICLP 2009 Workshop on Answer Set Programming and Other Computing Paradigms (ASPOCP 2009) (July 2009)
8. Balduccini, M.: An Answer Set Solver for non-Herbrand Programs: Progress Report. In: Costa, V.S., Dovier, A. (eds.) Technical Communications of the 28th International Conference on Logic Programming (ICLP 2012). Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2012)
9. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
10. Bell, C., Nerode, A., Ng, R.T., Subrahmanian, V.: Mixed Integer Programming Methods for Computing Nonmonotonic Deductive Databases. Journal of the ACM 41, 1178–1215 (1994)
11. Calimeri, F., Cozza, S., Ianni, G., Leone, N.: Enhancing asp by functions: Decidable classes and implementation techniques. In: Fox, M., Poole, D. (eds.) AAAI. AAAI Press (2010)
12. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F., Schaub, T.: ASP-Core-2: 4th ASP Competition Official Input Language Format (2013), http://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.01c.pdf
13. Calimeri, F., Faber, W., Gebser, M., Ianni, G., Kaminski, R., Krennwallner, T., Leone, N., Ricca, F., Schaub, T.: ASP Standardization Activity (2013), http://www.mat.unical.it/aspcomp2013/ASPStandardization/
14. Calimeri, F., Ianni, G., Krenwallner, T., Ricca, F.: The 4th ASP Competition Organizing Committee: The Fourth Answer Set Programming Competition homepage (2013), http://www.mat.unical.it/aspcomp2013/
15. Calimeri, F., Ianni, G., Ricca, F.: Third ASP Competition, File and language formats (2011), http://www.mat.unical.it/aspcomp2011/files/LanguageSpecifications.pdf
16. Calimeri, F., Ianni, G., Ricca, F.: The third open answer set programming competition. Theory and Practice of Logic Programming FirstView, 1–19 (2012), http://dx.doi.org/10.1017/S1471068412000105

17. Charwat, G., Ianni, G., Krennwallner, T., Kronegger, M., Pfandler, A., Redl, C., Schwengerer, M., Spendier, L., Wallner, J.P., Xiao, G.: VCWC: A versioning competition workflow compiler. In: Cabalar, P., Son, T.C. (eds.) LPNMR 2013. LNCS (LNAI), vol. 8148, pp. 233–238. Springer, Heidelberg (2013), http://www.kr.tuwien.ac.at/staff/tkren/pub/2013/lpnmr2013-vcwc.pdf
18. Dal Palù, A., Dovier, A., Pontelli, E., Rossi, G.: GASP: Answer set programming with lazy grounding. Fundamenta Informaticae 96(3), 297–322 (2009)
19. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and Expressive Power of Logic Programming. ACM Computing Surveys 33(3), 374–425 (2001)
20. Denecker, M., Vennekens, J., Bond, S., Gebser, M., Truszczyński, M.: The Second Answer Set Programming Competition. In: Erdem, E., Lin, F., Schaub, T. (eds.) LPNMR 2009. LNCS, vol. 5753, pp. 637–654. Springer, Heidelberg (2009)
21. Drescher, C., Walsh, T.: A translational approach to constraint answer set solving. Theory and Practice of Logic Programming 10(4-6), 465–480 (2010)
22. Drescher, C., Walsh, T.: Answer set solving with lazy nogood generation. In: Dovier, A., Costa, V.S. (eds.) ICLP (Technical Communications). LIPIcs, vol. 17, pp. 188–200. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2012)
23. Eiter, T., Faber, W., Leone, N., Pfeifer, G.: Declarative Problem-Solving Using the DLV System. In: Minker, J. (ed.) Logic-Based Artificial Intelligence, pp. 79–103. Kluwer Academic Publishers (2000)
24. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive Datalog. ACM Transactions on Database Systems 22(3), 364–418 (1997)
25. Eiter, T., Ianni, G., Krennwallner, T.: Answer Set Programming: A Primer. In: Tessaris, S., Franconi, E., Eiter, T., Gutierrez, C., Handschuh, S., Rousset, M.-C., Schmidt, R.A. (eds.) Reasoning Web 2009. LNCS, vol. 5689, pp. 40–110. Springer, Heidelberg (2009)
26. Faber, W., Leone, N., Pfeifer, G.: Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence 175(1), 278–298 (2011)
27. Falkner, A., Haselböck, A., Schenner, G.: Modeling Technical Product Configuration Problems. In: Proceedings of ECAI 2010 Workshop on Configuration, Lisbon, Portugal, pp. 40–46 (2010)
28. Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., Thiele, S.: Engineering an incremental ASP solver. In: Garcia de la Banda, M., Pontelli, E. (eds.) ICLP 2008. LNCS, vol. 5366, pp. 190–205. Springer, Heidelberg (2008)
29. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T., Schneider, M.T., Ziller, S.: A portfolio solver for answer set programming: Preliminary report. In: Delgrande, J.P., Faber, W. (eds.) LPNMR 2011. LNCS, vol. 6645, pp. 352–357. Springer, Heidelberg (2011)
30. Gebser, M., Kaufmann, B., Schaub, T.: Multi-threaded ASP solving with clasp. Theory and Practice of Logic Programming 12(4-5), 525–545 (2012)
31. Gebser, M., Kaufmann, B., Schaub, T.: Advanced conflict-driven disjunctive answer set solving. In: Rossi, F. (ed.) Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI 2013). IJCAI/AAAI (to appear, 2013)
32. Gebser, M., Kaufmann, B., Kaminski, R., Ostrowski, M., Schaub, T., Schneider, M.T.: Potassco: The Potsdam Answer Set Solving Collection. AI Communications 24(2), 107–124 (2011)
33. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artificial Intelligence 187-188, 52–89 (2012)
34. Gebser, M., Liu, L., Namasivayam, G., Neumann, A., Schaub, T., Truszczyński, M.: The first answer set programming system competition. In: Baral, C., Brewka, G., Schlipf, J. (eds.) LPNMR 2007. LNCS (LNAI), vol. 4483, pp. 3–17. Springer, Heidelberg (2007)
35. Gelfond, M., Leone, N.: Logic Programming and Knowledge Representation – the A-Prolog perspective. Artificial Intelligence 138(1-2), 3–38 (2002)

36. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, 365–385 (1991)
37. Giunchiglia, E., Lierler, Y., Maratea, M.: Answer set programming based on propositional satisfiability. Journal of Automated Reasoning 36(4), 345–377 (2006)
38. Gusfield, D., Irving, R.W.: The stable marriage problem: structure and algorithms. MIT Press, Cambridge (1989)
39. Janhunen, T., Niemelä, I.: Compact translations of non-disjunctive answer set programs to propositional clauses. In: Balduccini, M., Son, T.C. (eds.) Logic Programming, Knowledge Representation, and Nonmonotonic Reasoning. LNCS, vol. 6565, pp. 111–130. Springer, Heidelberg (2011)
40. Janhunen, T., Niemelä, I., Seipel, D., Simons, P., You, J.H.: Unfolding Partiality and Disjunctions in Stable Model Semantics. ACM Transactions on Computational Logic 7(1), 1–37 (2006)
41. Lefèvre, C., Nicolas, P.: The first version of a new ASP solver: ASPeRiX. In: Erdem, E., Lin, F., Schaub, T. (eds.) LPNMR 2009. LNCS, vol. 5753, pp. 522–527. Springer, Heidelberg (2009)
42. 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), 499–562 (2006)
43. Lifschitz, V.: Answer Set Planning. In: Schreye, D.D. (ed.) Proceedings of the 16th International Conference on Logic Programming (ICLP 1999), pp. 23–37. The MIT Press, Las Cruces (1999)
44. Lin, F., Zhao, Y.: ASSAT: Computing Answer Sets of a Logic Program by SAT Solvers. Artificial Intelligence 157(1-2), 115–137 (2004)
45. Liu, G., Janhunen, T., Niemelä, I.: Answer set programming via mixed integer programming. In: 13th International Conference on Principles of Knowledge Representation and Reasoning (KR 2012), pp. 32–42 (2012)
46. Marek, V.W., Truszczyński, M.: Stable Models and an Alternative Logic Programming Paradigm. In: Apt, K.R., Marek, V.W., Truszczyński, M., Warren, D.S. (eds.) The Logic Programming Paradigm – A 25-Year Perspective, pp. 375–398. Springer (1999)
47. Nguyen, M., Janhunen, T., Niemelä, I.: Translating answer-set programs into bit-vector logic. CoRR abs/1108.5837 (2011)
48. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1994)
49. Simons, P., Niemelä, I., Soininen, T.: Extending and Implementing the Stable Model Semantics. Artificial Intelligence 138, 181–234 (2002)
50. Subrahmanian, V., Nau, D., Vago, C.: WFS + Branch and Bound = Stable Models. IEEE Transactions on Knowledge and Data Engineering 7(3), 362–377 (1995)
51. Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: the Condor experience. Concurrency and Computation: Practice and Experience 17(2-4), 323–356 (2005)
52. Wieringa, S., Heljanko, K.: Concurrent clause strengthening. In: Järvisalo, M., Van Gelder, A. (eds.) SAT 2013. LNCS, vol. 7962, pp. 116–132. Springer, Heidelberg (2013)
53. Wittocx, J., Mariën, M., Denecker, M.: The IDP system: a model expansion system for an extension of classical logic. In: Denecker, M. (ed.) International Workshop on Logic and Search (Lash), pp. 153–165 (2008)
54. Zhou, N.F.: The language features and architecture of B-Prolog. Theory and Practice of Logic Programming 12(1-2), 189–218 (2012)