# The Design of the Sixth Answer Set Programming Competition – Report –

Martin Gebser[1⋆], Marco Maratea[2], and Francesco Ricca[3]

[1] Helsinki Institute for Information Technology HIIT, Aalto University, Finland
[2] DIBRIS, Università di Genova, Italy
[3] Dipartimento di Matematica e Informatica, Università della Calabria, Italy

**Abstract.** Answer Set Programming (ASP) is a well-known paradigm of declarative programming with roots in logic programming and non-monotonic reasoning. Similar to other closely-related problem-solving technologies, such as SAT/SMT, QBF, Planning and Scheduling, advances in ASP solving are assessed in competition events. In this paper, we report about the design of the Sixth ASP Competition, which is jointly organized by the University of Calabria (Italy), Aalto University (Finland), and the University of Genova (Italy), in affiliation with the 13th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR 2015). This edition maintains some of the design decisions introduced in the last event, e.g., the design of tracks, the scoring scheme, and the adherence to a fixed modeling language in order to push the adoption of the ASP-Core-2 standard. On the other hand, it features also some novelties, like a benchmarks selection stage to classify instances according to their expected hardness, and a "marathon" track where the best performing systems are given more time for solving hard benchmarks.

## 1 Introduction

Answer Set Programming [7, 13–15, 28, 29, 35, 41, 44] is a well-known declarative programming approach to knowledge representation and reasoning, with roots in the areas of logic programming and non-monotonic reasoning as well as close relationships to other formalisms such as SAT, SAT Modulo Theories, Constraint Programming, PDDL, and many others. With the exception of the fifth event,[4] which was held in 2014 in order to join the FLoC Olympic Games at the Vienna Summer of Logic,[5] ASP Competitions are biennial events organized in odd years. The goal of the Answer Set Programming (ASP) Competition series is to access the state of the art in ASP solving (see, e.g., [1, 10, 12, 23, 25, 30, 31, 33, 36, 37, 42, 45] on challenging benchmarks.

In this paper, we report about the design of the Sixth ASP Competition,[6] jointly organized by the University of Calabria (Italy), Aalto University (Finland), and the University of Genova (Italy), in affiliation with the 13th International Conference on Logic

⋆ Also affiliated with the University of Potsdam, Germany.
[4] https://www.mat.unical.it/aspcomp2014/
[5] http://vsl2014.at/
[6] https://aspcomp2015.dibris.unige.it/

Programming and Non-Monotonic Reasoning (LPNMR 2015).[7] This edition maintains some of the design decisions introduced in the last event, e.g., $(i)$ the design of tracks, based on the "complexity" of the encoded problems (as in past events), but also considering the language features involved in encodings (e.g., choice rules, aggregates, presence of queries), $(ii)$ the scoring scheme, which had been significantly simplified, and $(iii)$ the adherence to a fixed modeling language in order to push the adoption of the ASP-Core-2 standard.[8] On the other hand, we also introduce novelties, some of them borrowed from past editions of the SAT and QBF Competitions, i.e., $(i)$ a benchmarks selection stage to classify instances according to their expected hardness, in order to select instances from a broad range of difficulty, and $(ii)$ a "marathon" track where the best performing systems are given more time for solving hard benchmarks, in order to check whether they are able to complete difficult instances in the long run.

The present report is structured as follows. First, Section 2 introduces the setting of the Sixth ASP Competition. Then, Section 3 and 4 present the problem domains and the instance selection process, respectively. Section 5 surveys the participants and systems registered for the competition. The report is concluded by final remarks in Section 6.

## 2 Format of the Sixth ASP Competition

In this section, we discuss the format of the competition event, describe categories and tracks, and recapitulate the scoring scheme along with general rules. Furthermore, we provide some information about the competition infrastructure.

As outlined in Section 1, the Sixth ASP competition maintains choices made in the last event, but also adds some novelties. First, the scoring scheme, which was significantly simplified in the last edition (cf. [11]), remains unchanged. In order to encourage new teams and research groups to join the event, we also maintain the division into tracks, primarily based on language features rather than inherent computational complexity, as in the last edition. Given this, preliminary or otherwise confined systems may take part in some tracks only, i.e., the ones featuring the subset of the language they support. Furthermore, the tracks draw a clearer and more detailed picture about what (combinations of) techniques work well for particular language features, which, in our opinion, is more interesting than merely reporting overall winners.

*Competition format.* The competition is open to any general-purpose solving system, provided it is able to parse the ASP-Core-2 input format. However, following the positive experience of 2014, we also plan to organize an on-site modeling event at LPNMR 2015, in the spirit of the Prolog contest. Regarding benchmarks, this year featured a call to submit new domains (see Section 3) that, together with the domains employed in the last event, are part of the benchmark collection of this edition. For the latter, the Fifth ASP Competition proposed and evaluated a new set of encodings: this year we fix the encodings to those that led to better performance in 2014. For new domains, we consider the encodings provided by benchmark contributors. The whole benchmark set undergoes a benchmark selection phase in order the classify instances based on their

[7] `http://lpnmr2015.mat.unical.it/`

[8] `https://www.mat.unical.it/aspcomp2013/ASPStandardization/`

expected hardness, and then to pick instances of varying difficulty to be run in the competition (see Section 4 for details).

*Competition categories.* The competition consists of *two categories*, depending on the computational resources made available to each running system:

- **SP**: One processor allowed;
- **MP**: Multiple processors allowed.

While the **SP** category aims at sequential solving systems, parallelism can be exploited in the **MP** category.

*Competition tracks.* Both categories of the competition are structured into *four tracks*, which are described next:

- **Track #1**: *Basic Decision*. Encodings: normal logic programs, simple arithmetic and comparison operators.
- **Track #2**: *Advanced Decision*. Encodings: full language, with queries, excepting optimization statements and non-HCF disjunction.
- **Track #3**: *Optimization*. Encodings: full language with optimization statements, excepting non-HCF disjunction.
- **Track #4**: *Unrestricted*. Encodings: full language.

We also plan to introduce a **Marathon** track this year, thus analyzing participant systems along a different dimension. The idea, borrowed from past QBF Competitions, is to grant more time to the best solvers on a limited set of instances that proved to be difficult in regular tracks.

*Scoring scheme.* The scoring scheme adopted is the same as in the Fifth ASP Competition. In particular, it considers the following factors:

- Problems are always weighted equally.
- If a system outputs an incorrect answer to some instance of a problem, this invalidates its score for the problem, even if other instances are correctly solved.
- In case of Optimization problems, scoring is mainly based on solution quality.

In general, 100 points can be earned for each benchmark problem. The final score of a solving system consists of the sum of scores over all problems.

*Scoring details.* For *Decision and Query problems*, the score of a solver $S$ on a problem $P$ featuring $N$ instances is computed as

$$S(P) = \frac{N_S * 100}{N}$$

where $N_S$ is the number of instances solved within the allotted time and memory limits.

For *Optimization problems*, solvers are ranked by solution quality, in the spirit of the MANCOOSI International Solver Competition.[9] Given $M$ participant systems, the

[9] `http://www.mancoosi.org/misc/`

score of a solver $S$ for an instance $I$ of a problem $P$ featuring $N$ instances is computed as

$$S(P, I) = \frac{M_S(I) * 100}{M * N}$$

where $M_S(I)$ is

- 0, if $S$ did neither provide a solution, nor report unsatisfiability, or
- the number of participant systems that did not provide any strictly better solution than $S$, where a confirmed optimum solution is considered strictly better than an unconfirmed one, otherwise.

The score $S(P)$ of a solver $S$ for problem $P$ consists of the sum of scores $S(P, I)$ over all $N$ instances $I$ featured by $P$. Note that, as with Decision and Query problems, $S(P)$ can range from 0 to 100.

*Global ranking.* The global ranking for each track, and the overall ranking, is obtained by awarding each participant system the sum of its scores over all problems; systems are ranked by their sums, in decreasing order. In case of a draw in terms of sums of scores, sums of runtimes are taken into account.

*Competition environment.* The competition is run on a Debian Linux server (64bit kernel), featuring 2.30GHz Intel Xeon E5-4610 v2 Processors with 16MB of cache and 128GB of RAM. Time and memory for each run are limited to 20 minutes and 12GB, respectively. Participant systems can exploit up to 8 cores (i.e., up to 16 virtual CPUs since Intel Hyperthreading technology is enabled) in the **MP** category, whereas the execution is constrained to one core in the **SP** category. The execution environment is composed of a number of scripts, and performance is measured using the *pyrunlim* tool.[10]

## 3 Benchmark Suite

The benchmark domains considered in the Sixth ASP Competition include those from the previous edition, summarized first. Moreover, encodings and instances were provided for six new domains, introduced afterwards.

*Previous domains.* The Fifth ASP Competition featured 26 benchmark domains that had been submitted to earlier editions already, mainly in 2013 when the ASP-Core-2 standard input format was specified. In some domains, however, "unoptimized" encodings submitted by benchmark authors incurred grounding bottlenecks that made participant systems fail on the majority of instances. In view of this and in order to enrich the available benchmark collection, alternative encodings were devised and empirically compared last year for all but two domains dealing with Query answering, which were modeled by rather straightforward positive programs.

The first part of assembling the benchmark suite for the Sixth ASP Competition consisted in the choice of encodings for previously used domains. Table 1 gives an

[10] `https://github.com/alviano/python/`

overview of these domains, outlining application-oriented problems, respective computational tasks, i.e., Decision, Optimization, or Query answering, and tracks. Most importantly, the fourth column indicates whether the encoding made available in 2013 or the alternative one provided last year has been picked for this edition of the ASP Competition. The selection was based on the results from 2014, favoring the encoding variant that exhibited better performance of participant systems in a benchmark domain.

For Decision problems in the *Hanoi Tower*, *Knight Tour with Holes*, *Stable Marriage*, *Incremental Scheduling*, *Partner Units*, *Solitaire*, *Weighted-Sequence Problem*, and *Minimal Diagnosis* domains, all systems benefited from the usage of alternative 2014 encodings. Although the results were not completely uniform, improvements of more systems or greater extent were obtained in *Graph Colouring*, *Visit-all*, *Nomystery*, *Permutation Pattern Matching*, and *Qualitative Spatial Reasoning*. On two remaining Decision problems, *Sokoban* and *Complex Optimization*, no significant performance gaps were observed, and 2014 encodings were picked as they simplify the original submissions, i.e., aggregates are omitted in *Sokoban* and redundant preconditions of rules dropped in *Complex Optimization*. In fact, due to similar simplifications, the Basic Decision track (#1) consists of six domains, while it previously included *Labyrinth* and *Stable Marriage* only. On the other hand, the encodings from 2013 were kept for domains where alternative variants did not lead to improvements or even deteriorated performance, as it was the case in *Graceful Graphs*.

In view of the relative scoring of systems on Optimization problems, the selection of encoding variants could not be based on (uniform) improvements in terms of score here. Rather than that, we investigated timeouts, runtimes, and solution quality of the top-performing systems from last year, thus concentrating on the feasibility of good but not necessarily optimal solutions. In this regard, the alternative 2014 encodings turned out to be advantageous in *Crossing Minimization* and *Maximal Clique*, while the original submissions led to better results in *Connected Still Life* and *Valves Location*, or essentially similar performance in *Abstract Dialectical Frameworks*.

Notably, this edition of the ASP Competition utilizes a revised formulation of *Connected Still Life* (thus marked by '*' in Table 1), where instances specify grid cells that must be "dead" or "alive" according to the Game of Life version considered in this domain. Respective conditions are addressed by side constraints added to the previously available encodings and enable a diversification of instances of same size, while size had been the only parameter for obtaining different instances before. In addition, benchmark authors provided new instance sets for the *Knight Tour with Holes*, *Stable Marriage*, *Ricochet Robots*, and *Maximal Clique* domains. For *Knight Tour with Holes*, the instances from last year were too hard for most participant systems, and too easy in the other three domains. Finally, recall that the 2013 encodings for Query problems in the *Reachability* and *Strategic Companies* domains are reused.

Six of the 26 benchmark domains stemming from earlier editions of the ASP competition are based on particular applications. In more detail, *Incremental Scheduling* [6] deals with assigning jobs to devices such that the makespan of a schedule stays within a given budget. The matching problem *Partner Units* [5] has applications in the configuration of surveillance, electrical engineering, computer network, and railway safety systems. The *Crossing Minimization* [17] domain aims at optimized layouts of hierar-

**Table 1.** Encodings selected for benchmark domains from the Fifth ASP Competition

| Domain | App | Problem | Encoding | |
|---|---|---|---|---|
| *Graph Colouring* | | Decision | 2014 | Track #1 |
| *Hanoi Tower* | | Decision | 2014 | |
| *Knight Tour with Holes* | | Decision | 2014 | |
| *Labyrinth* | | Decision | 2013 | |
| *Stable Marriage* | | Decision | 2014 | |
| *Visit-all* | | Decision | 2014 | |
| *Bottle Filling* | | Decision | 2013 | Track #2 |
| *Graceful Graphs* | | Decision | 2013 | |
| *Incremental Scheduling* | √ | Decision | 2014 | |
| *Nomystery* | | Decision | 2014 | |
| *Partner Units* | √ | Decision | 2014 | |
| *Permutation Pattern Matching* | | Decision | 2014 | |
| *Qualitative Spatial Reasoning* | | Decision | 2014 | |
| *Reachability* | | Query | 2013 | |
| *Ricochet Robots* | | Decision | 2013 | |
| *Sokoban* | | Decision | 2014 | |
| *Solitaire* | | Decision | 2014 | |
| *Weighted-Sequence Problem* | | Decision | 2014 | |
| *Connected Still Life*$^*$ | | Optimization | 2013 | Track #3 |
| *Crossing Minimization* | √ | Optimization | 2014 | |
| *Maximal Clique* | | Optimization | 2014 | |
| *Valves Location* | √ | Optimization | 2013 | |
| *Abstract Dialectical Frameworks* | | Optimization | 2013 | Track #4 |
| *Complex Optimization* | √ | Decision | 2014 | |
| *Minimal Diagnosis* | √ | Decision | 2014 | |
| *Strategic Companies* | | Query | 2013 | |

chical network diagrams in graph drawing. The hydroinformatics problem *Valves Location* [18] is concerned with designing water distribution systems such that the isolation in case of damages is minimized. In contrast to objective functions considered in the Optimization track (#3), the *Complex Optimization* [22] domain addresses subset minimization in the contexts of biological network repair [19] and minimal unsatisfiable core membership [32]. Finally, *Minimal Diagnosis* [27] tackles the identification of minimal reasons for inconsistencies between biological networks and experimental data.

*New domains.* Six new benchmark domains, all of which are application-oriented as indicated in Table 2, were submitted to the Sixth ASP Competition:

- *Combined Configuration* [26] is a Decision problem inspired by industrial product configuration tasks dealing with railway interlocking systems, automation systems, etc. In the considered scenario, orthogonal requirements as encountered in bin packing, graph coloring, matching, partitioning, and routing must be fulfilled by a common solution. Since the combined problem goes beyond its individual subtasks, specialized procedures for either of them are of limited applicability, and the challenge is to integrate all requirements into general solving methods.

**Table 2.** New benchmark domains of the Sixth ASP Competition

| Domain | App | Problem | |
|---|---|---|---|
| *Combined Configuration* | √ | Decision | Tr. #2 |
| *Consistent Query Answering* | √ | Query | |
| *MaxSAT* | √ | Optimization | Track #3 |
| *Steiner Tree* | √ | Optimization | |
| *System Synthesis* | √ | Optimization | |
| *Video Streaming* | √ | Optimization | |

- *Consistent Query Answering* [38] addresses phenomena arising in the integration of data from heterogeneous sources. The goal is to merge as much information as possible, even though local inconsistencies and incompleteness typically preclude a mere data fusion. In particular, the Query problem amounts to cautious reasoning, retrieving consequences that are valid under all candidate repairs of input data.
- *MaxSAT* [34] is the optimization variant of SAT, where so-called soft clauses may be violated to particular costs and the sum of costs ought to be minimal. Industrial instances, taken from the 2014 MaxSAT Evaluation,[11] are represented by facts and encoded as an Optimization problem.
- *Steiner Tree* [16] is concerned with connecting particular endpoints by a spanning tree. The domain deals with the rectilinear version of this problem, where points on a two-dimensional grid may be connected by horizontal or vertical line segments. This setting is of practical relevance as it corresponds to wire routing in circuit design. The accumulated line segments determine a wire length, which is subject to minimization in the considered Optimization problem.
- *System Synthesis* [9] deals with the allocation of parallel tasks and message routing in integrated hardware architectures for target applications. On the one hand, the capacities of processing elements are limited, so that communicating tasks must be distributed. On the other hand, network communication shall avoid long routes to reduce delays. The Optimization problem combines three lexicographically ordered objectives: balancing the allocation of processing elements, minimizing network communication, and keeping routes short.
- *Video Streaming* [46] aims at an adaptive regulation of resolutions and bit rates in a content delivery network. While the bit rates of users and the number of different video formats that can be offered simultaneously are limited, service disruptions are admissible for a fraction of users only. The objective of the Optimization problem is to achieve high user satisfaction with respect to particular video contents.

## 4 Benchmark Selection

For an informed instance selection going beyond the random selection adopted in the 2014 edition of the ASP Competition or the solver-dependent criterion employed in 2013, we utilize an instance selection strategy inspired by the 2014 SAT Competition.[12]

[11] `http://www.maxsat.udl.cat/14/index.html`

[12] `http://www.satcompetition.org/2014/index.shtml`

First, the empirical hardness of all available instances is evaluated by running the top-performing systems from last year, and then a balanced selection is made among instances of varying difficulty.

*Top-performing systems.* We considered the best performing system per team that participated in the Fifth ASP Competition, corresponding to the systems taking the first three places last year, i.e., CLASP, LP2NORMAL2+CLASP, and WASP-1.5. This choice comes close to the ideal state-of-the-art solver that matches the best performing system on each instance.

*Instance classification.* All instances available in the benchmark collection are classified according to the runtimes of the top-performing systems by picking the upmost applicable category as follows:

**(non-groundable)** Instances that could not be grounded by any of the top-performing systems within the timeout of 20 minutes.
**(very easy)** Instances solved by all top-performing systems in less than 20 seconds.
**(easy)** Instances solved by all top-performing systems in less than 2 minutes.
**(medium)** Instances solved by all top-performing systems within the timeout of 20 minutes.
**(hard)** Instances solved by at least one among the top-performing systems within 40 minutes, i.e., twice the timeout.
**(too hard)** Instances that could not be solved (no solution produced in case of Optimization problems) by any of the top-performing systems within 40 minutes.

While non-groundable instances are basically out of reach for traditional ASP systems, very easy ones are highly unlikely to yield any relevant distinction between participant systems. Hence, instances falling into the first two categories are discarded and not run in the competition. Unlike that, easy, medium, and hard instances are expected to differentiate between unoptimized, average, and top-performing competition entries. Albeit they may not be solvable by any participant system within 20 minutes, too hard instances are included to impose challenges and are primary candidates for the Marathon track in which the timeout will be increased.

*Instance selection.* Instances to be run in the competitions will be picked per benchmark domain, matching the following conditions as much as possible:

1. 20 instances are included in each domain.
2. Easy, medium, hard, too hard, and randomly picked (yet excluding non-groundable and very easy) instances shall evenly contribute 20% (i.e., four) instances each.
3. Satisfiable and unsatisfiable instances should be balanced (if known/applicable).
4. The selection among candidate instances according to the previous conditions is done randomly, using the concatenation of winning numbers in the EuroMillions lottery of 23rd June 2015 as seed.

Further criteria will be taken into account to filter domains by need. That is, domains in which instances lack variety, i.e., all instances turn out as easy to medium or (too)

hard, may be excluded in the competition. We do not impose strict conditions, however, as being new, application-oriented, based on ASP-specific language features (e.g., aggregates, recursion, or disjunction) or a particular computational task (Optimization or Query answering) may justify an interest beyond the scalability of available instances.

*Preliminary data.* The instance classification process has been running at the time of writing this report. In the first stage, non-groundable instances were identified and discarded, thus dropping 88 of the available instances (86 from *Incremental Scheduling* and two from *Sokoban*). This leaves 4970 instances for running the three top-performing systems from last year, using a timeout of 40 minutes. We expect to obtain complete results of these runs, on which the instance selection will be based, by 22nd June 2015.

## 5 Participants

In this section, we briefly survey the participants and systems registered for the competition. In total, the competition features 13 systems coming from three teams:

- The Aalto team from Aalto University submitted nine systems, mainly working by means of translations [10, 20, 37, 43]. Two systems, LP2SAT+LINGELING and LP2SAT+PLINGELING-MT, rely on translation to SAT, which includes the normalization of aggregates as well as the encoding of level mappings for non-tight problem instances. The latter are expressed in terms of acyclicity checking [20, 21] on top of ASP, Pseudo-Boolean or SAT formulations, respectively, used in the systems LP2ACYCASP+CLASP, LP2ACYCPB+CLASP, LP2ACYCSAT+CLASP, and LP2ACYCSAT+GLUCOSE. While LP2SAT+LINGELING and LP2SAT+PLINGELING-MT do not support optimization and participate in the Basic and Advanced Decision tracks (#1 and #2) only, the latter systems compete also in the Optimization track (#3). The same applies to LP2MIP and LP2MIP-MT, which run CPLEX as a Mixed Integer Programming solver back-end. Finally, LP2NORMAL+CLASP normalizes aggregates (of small to medium size) and uses CLASP as back-end ASP solver; LP2NORMAL+CLASP participates in all four tracks and thus also in the Unrestricted track (#4). All systems by the Aalto team utilize GRINGO-4 for grounding, and neither of them supports Query problems (*Consistent Query Answering*, *Reachability*, and *Strategic Companies*). The systems LP2SAT+PLINGELING-MT and LP2MIP-MT exploit multi-threading and run in the **MP** category, while the other, sequential systems participate in the **SP** category.
- The ME-ASP team from the University of Genova, the University of Sassari, and the University of Calabria submitted the multi-engine ASP solver ME-ASP [39, 40]. ME-ASP applies a selection policy to decide what is the most promising ASP solver to run, given some characteristics of an input program. The pool of ASP solvers from which ME-ASP can choose is a selection of the solvers submitted to the Fifth ASP Competition, while input characteristics correspond to non-ground and ground features. The ME-ASP system utilizes GRINGO-4 for grounding and participates in all four tracks of the **SP** category.

- The Wasp team from the University of Calabria submitted two systems based on WASP [1, 2, 4], namely WASP and WASP+DLV, as well as the proof-of-concept prototype JWASP, written in Java. WASP is a native ASP solver based on conflict-driven learning, yet extended with techniques specifically designed for solving disjunctive logic programs. It utilizes GRINGO-4 for grounding and participates in all tracks, although with limited support for Query problems. On the other hand, WASP+DLV includes full functionalities for Query answering [3, 33] and competes in all domains. The prototype system JWASP is based on the SAT4J [8] SAT solver and implements some of the algorithms employed in WASP for handling ASP-specific features, which enables its participation in the Basic and Advanced Decision tracks (#1 and #2). All systems by the Wasp team run in the **SP** category.

In sum, similar to past competitions, the vast majority of submitted systems is based on two main approaches to ASP solving: $(i)$ "native" systems, which exploit techniques purposely conceived and/or adapted for dealing with logic programs under the stable models semantics, and $(ii)$ "translation-based" systems, which (roughly) at some stage of the evaluation produce an intermediate specification in some different formalism that is then fed to a corresponding solver. The solvers submitted by the Wasp team as well as ME-ASP and LP2NORMAL+CLASP pursue a native approach, while the remaining systems by the Aalto team utilize translations.

The main novelty among competition entries this year is the "portfolio" solver ME-ASP. Its multi-engine approach differs from CLASPFOLIO [24], which participated last in the 2013 edition of the ASP Competition. Furthermore, it is worth mentioning that, in order to assess the improvements in ASP solving, we also consider the version of CLASP submitted in 2014 for reference, given that CLASP was the overall winner of the Fifth ASP Competition.

## 6 Conclusions

The Sixth ASP Competition is jointly organized by the University of Calabria (Italy), Aalto University (Finland), and the University of Genova (Italy), in affiliation with the 13th International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR 2015). The main goal is measuring advances of the state of the art in ASP solving, where native and translation-based systems constitute the two main approaches. This report presented the design of the event and gave an overview of benchmarks as well as participants. On the one hand, this edition of the ASP Competition maintains design decisions from 2014, e.g., tracks are conceived on the basis of language features. On the other hand, it also introduces some novelties, i.e., a benchmark selection phase and a Marathon track.

The competition results will be announced at LPNMR 2015, at which, following the positive experience of 2014, we also plan to organize another on-site modeling event in the spirit of the Prolog contest. This modeling competition to some extent replaces the Model&Solve track that was included last in the 2013 edition of the ASP Competition, yet the idea is to margin the effort of problem modeling.

## References


1. Alviano, M., Dodaro, C., Faber, W., Leone, N., Ricca, F.: Wasp: A native ASP solver based on constraint learning. In: Proceedings of LPNMR'13, pp. 54–66. Springer (2013)
2. Alviano, M., Dodaro, C., Leone, N., Ricca, F.: Advances in Wasp. In: Proceedings of LPNMR'15. Springer (2015)
3. Alviano, M., Dodaro, C., Ricca, F.: Anytime computation of cautious consequences in answer set programming. Theory and Practice of Logic Programming 14(4-5), 755–770 (2014)
4. Alviano, M., Dodaro, C., Ricca, F.: Preliminary report on Wasp 2.0. In: Proceedings of NMR'14, pp. 68–72. Vienna University of Technology (2014)
5. Aschinger, M., Drescher, C., Friedrich, G., Gottlob, G., Jeavons, P., Ryabokon, A., Thorstensen, E.: Optimization methods for the partner units problem. In: Proceedings of CPAIOR'11, pp. 4–19. Springer (2011)
6. Balduccini, M.: Industrial-size scheduling with ASP+CP. In: Proceedings of LPNMR'11, pp. 284–296. Springer (2011)
7. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
8. Berre, D., Parrain, A.: The Sat4j library, release 2.2. Journal on Satisfiability, Boolean Modeling and Computation 7, 59–64 (2010)
9. Biewer, A., Andres, B., Gladigau, J., Schaub, T., Haubelt, C.: A symbolic system synthesis approach for hard real-time systems based on coordinated SMT-solving. In: Proceedings of DATE'15, pp. 357–362. ACM (2015)
10. Bomanson, J., Gebser, M., Janhunen, T.: Improving the normalization of weight rules in answer set programs. In: Proceedings of JELIA'14, pp. 166–180. Springer (2014)
11. Calimeri, F., Gebser, M., Maratea, M., Ricca, F.: The design of the fifth answer set programming competition. In: Technical Communications of ICLP'14, `http://arxiv.org/abs/1405.3710v4`. CoRR (2014)
12. Dal Palù, A., Dovier, A., Pontelli, E., Rossi, G.: GASP: Answer set programming with lazy grounding. Fundamenta Informaticae 96(3), 297–322 (2009)
13. Eiter, T., Faber, W., Leone, N., Pfeifer, G.: Declarative problem-solving using the DLV system. In: Logic-Based Artificial Intelligence, pp. 79–103. Kluwer Academic Publishers (2000)
14. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive Datalog. ACM Transactions on Database Systems 22(3), 364–418 (1997)
15. Eiter, T., Ianni, G., Krennwallner, T.: Answer set programming: A primer. In: Proceedings of RW'09, pp. 40–110. Springer (2009)
16. Erdem, E., Wong, M.: Rectilinear Steiner tree construction using answer set programming. In: Proceedings of ICLP'04, pp. 386–399. Springer (2004)
17. Gange, G., Stuckey, P., Marriott, K.: Optimal $k$-level planarization and crossing minimization. In: Proceedings of GD'10, pp. 238–249. Springer (2010)
18. Gavanelli, M., Nonato, M., Peano, A., Alvisi, S., Franchini, M.: An ASP approach for the valves positioning optimization in a water distribution system. In: Proceedings of CILC'12, pp. 134–148. CEUR-WS.org (2012)
19. Gebser, M., Guziolowski, C., Ivanchev, M., Schaub, T., Siegel, A., Thiele, S., Veber, P.: Repair and prediction (under inconsistency) in large biological networks with answer set programming. In: Proceedings of KR'10, pp. 497–507. AAAI (2010)
20. Gebser, M., Janhunen, T., Rintanen, J.: Answer set programming as SAT modulo acyclicity. In: Proceedings of ECAI'14, pp. 351–356. IOS (2014)
21. Gebser, M., Janhunen, T., Rintanen, J.: SAT modulo graphs: Acyclicity. In: Proceedings of JELIA'14, pp. 137–151. Springer (2014)

22. Gebser, M., Kaminski, R., Schaub, T.: Complex optimization in answer set programming. Theory and Practice of Logic Programming 11(4-5), 821–839 (2011)
23. Gebser, M., Kaufmann, B., Schaub, T.: Advanced conflict-driven disjunctive answer set solving. In: Proceedings of IJCAI'13, pp. 912–918. IJCAI/AAAI (2013)
24. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T., Schneider, M., Ziller, S.: A portfolio solver for answer set programming: Preliminary report. In: Proceedings of LPNMR'11, pp. 352–357. Springer (2011)
25. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artificial Intelligence 187-188, 52–89 (2012)
26. Gebser, M., Ryabokon, A., Schenner, G.: Combining heuristics for configuration problems using answer set programming. In: Proceedings of LPNMR'15. Springer (2015)
27. Gebser, M., Schaub, T., Thiele, S., Veber, P.: Detecting inconsistencies in large biological networks with answer set programming. Theory and Practice of Logic Programming 11(2-3), 323–360 (2011)
28. Gelfond, M., Leone, N.: Logic programming and knowledge representation – the A-Prolog perspective. Artificial Intelligence 138(1-2), 3–38 (2002)
29. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing 9, 365–385 (1991)
30. Giunchiglia, E., Lierler, Y., Maratea, M.: Answer set programming based on propositional satisfiability. Journal of Automated Reasoning 36(4), 345–377 (2006)
31. Janhunen, T., Niemelä, I., Seipel, D., Simons, P., You, J.: Unfolding partiality and disjunctions in stable model semantics. ACM Transactions on Computational Logic 7(1), 1–37 (2006)
32. Janota, M., Marques-Silva, J.: On deciding MUS membership with QBF. In: Proceedings of CP'11, pp. 414–428. Springer (2011)
33. 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)
34. Li, C., Manyà, F.: MaxSAT. In: Handbook of Satisfiability, pp. 613–631. IOS (2009)
35. Lifschitz, V.: Answer set programming and plan generation. Artificial Intelligence 138(1-2), 39–54 (2002)
36. Lin, F., Zhao, Y.: ASSAT: Computing answer sets of a logic program by SAT solvers. Artificial Intelligence 157(1-2), 115–137 (2004)
37. Liu, G., Janhunen, T., Niemelä, I.: Answer set programming via mixed integer programming. In: Proceedings of KR'12, pp. 32–42. AAAI (2012)
38. Manna, M., Ricca, F., Terracina, G.: Consistent query answering via ASP from different perspectives: Theory and practice. Theory and Practice of Logic Programming 13(2), 227–252 (2013)
39. Maratea, M., Pulina, L., Ricca, F.: A multi-engine approach to answer-set programming. Theory and Practice of Logic Programming 14(6), 841–868 (2014)
40. Maratea, M., Pulina, L., Ricca, F.: Multi-level algorithm selection for ASP. In: Proceedings of LPNMR'15. Springer (2015)
41. Marek, V., Truszczyński, M.: Stable models and an alternative logic programming paradigm. In: The Logic Programming Paradigm: A 25-Year Perspective, pp. 375–398. Springer (1999)
42. Mariën, M., Wittocx, J., Denecker, M., Bruynooghe, M.: SAT(ID): Satisfiability of propositional logic extended with inductive definitions. In: Proceedings of SAT'08, pp. 211–224. Springer (2008)
43. Nguyen, M., Janhunen, T., Niemelä, I.: Translating answer-set programs into bit-vector logic. In: Proceedings of INAP'11 and WLP'11, pp. 105–116. Springer (2013)
44. Niemelä, I.: Logic programs with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence 25(3-4), 241–273 (1999)

45. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artificial Intelligence 138(1-2), 181–234 (2002)
46. Toni, L., Aparicio-Pardo, R., Simon, G., Blanc, A., Frossard, P.: Optimal set of video representations in adaptive streaming. In: Proceedings of MMSys'14, pp. 271–282. ACM (2014)