

The Design of the Fifth Answer Set Programming Competition

Francesco Calimeri and Francesco Ricca

Dipartimento di Matematica e Informatica, Università della Calabria, Italy

Martin Gebser*

Helsinki Institute for Information Technology HIIT

Department of Information and Computer Science, School of Science, Aalto University, Finland

Marco Maratea

Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi, Università di Genova, Italy

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

Answer Set Programming (ASP) is a well-established paradigm of declarative programming that has been developed in the field of logic programming and nonmonotonic reasoning. Advances in ASP solving technology are customarily assessed in competition events, as it happens for other closely-related problem-solving technologies like SAT/SMT, QBF, Planning and Scheduling. ASP Competitions are (usually) biennial events; however, the Fifth ASP Competition departs from tradition, in order to join the FLoC Olympic Games at the Vienna Summer of Logic 2014, which is expected to be the largest event in the history of logic. This edition of the ASP Competition series is jointly organized by the University of Calabria (Italy), the Aalto University (Finland), and the University of Genova (Italy), and is affiliated with the 30th International Conference on Logic Programming (ICLP 2014). It features a completely re-designed setup, with novelties involving the design of tracks, the scoring schema, and the adherence to a fixed modeling language in order to push the adoption of the ASP-Core-2 standard. Benchmark domains are taken from past editions, and best system packages submitted in 2013 are compared with new versions and solvers.

1 Introduction

Answer Set Programming (Baral 2003; Eiter et al. 2000; Eiter et al. 1997; Eiter et al. 2009; Gelfond and Leone 2002; Gelfond and Lifschitz 1991; Lifschitz 1999; Marek and Truszczyński 1999) is a well-established declarative programming approach to knowledge representation and reasoning, proposed in the area of nonmonotonic reasoning and logic programming, with close relationship to other formalisms such as SAT, SAT Modulo Theories, Constraint Handling Rules, PDDL, and many others. The Answer Set Programming (ASP) Competition is a (usually) biennial event whose goal is to access the state of the art in ASP solving (see, e.g., (Alviano et al. 2013b; Dal Palù et al. 2009; Gebser et al. 2013; Gebser et al. 2012a; Giunchiglia et al. 2006; Janhunen et al. 2006; Leone et al. 2006; Lin and Zhao 2004; Liu et al. 2012; Mariën et al. 2008; Simons et al. 2002) on challenging benchmarks. This year we depart from the usual timeline, and the event takes place one year

* Also affiliated with the University of Potsdam, Germany.

after the Fourth ASP Competition¹; basically, the two main reasons are: (i) to be part of the Vienna Summer of Logic (VSL)², which is expected to be the largest event in the history of logic, and (ii) to “push” the adoption of the new language standard ASP-Core-2³: in 2013 it was not fully supported by most participants, and/or the submitters did not have enough time to integrate new language features in a completely satisfactory way.

In this paper we report about the design of the Fifth ASP Competition⁴, jointly organized by the University of Calabria (Italy), the Aalto University (Finland), and the University of Genova (Italy). The event is part of the FLoC Olympic Games⁵ at VSL 2014, and is affiliated with the 30th International Conference on Logic Programming (ICLP 2014)⁶. The basis is a re-run of the System Track of the 2013 edition, i.e., same ASP solvers, and essentially same domains. Additionally, (i) we have given the participants the opportunity to submit updated versions of the solvers, yet the call was open to new participants; (ii) we have also reconsidered problem encodings, providing new versions for almost all problems; (iii) the domains have not been classified simply by taking into account the “complexity” of the encoded problems (as in past events), but also considering the language features involved in the encoding (e.g., choice rules, aggregates, presence of queries). This was intended not only to widen participation, but also in order to be able to measure both the progress of each solver, and of the state of the art in ASP solving, and draw a more complete picture about the approaches that can efficiently solve problems with different features, while still contributing to push the new standard.

The present report is structured as follows. Section 2 illustrates the setting of the competition, while Section 3 and Section 4 present the problem domains and ASP solvers taking part in the competition, respectively. The report ends by drawing some final remarks in Section 5.

2 Format of the Fifth ASP Competition

In this section, we illustrate the settings of the competition, focusing on the differences introduced with respect to the previous editions. The competition builds on the basis of the last competition, with each participant in the last edition having the chance (but not the obligation) to submit an updated version, and the focus is on the System Track. As already discussed in Section 1, this can be seen, in a sense, as a special edition of the competition series, and the format changes accordingly, especially when looking at track structure and scoring system.

We decided first to simplify the scoring system w.r.t. the last edition (see (Alviano et al. 2013a)), and to improve it in the case of optimization problems.

As far as the language is concerned, in order to encourage new teams and research groups to join the event, we completely redesigned the tracks, which are now conceived on language features, other than on a complexity basis. It now makes sense, for a brand new system, or even a preliminary/experimental new version of an old one, to take part only in some tracks, i.e., the ones featuring the subset of the language it correctly supports. In addition, this choice should pave the way to more interesting analyses, such as measuring the progress of a solver, as well of the state of the art, while still contributing to push the standardization process forward, one of

¹ <https://www.mat.unical.it/aspcomp2013/>

² <http://vsl2014.at/>

³ <https://www.mat.unical.it/aspcomp2013/ASPStandardization/>

⁴ <https://www.mat.unical.it/aspcomp2014/>

⁵ <http://vsl2014.at/olympics/>

⁶ <http://users.ugent.be/~tschrijv/ICLP2014/>

the main goal of the competition series. Furthermore, the tracks are supposed to draw a clearer and more complete picture about what (combinations of) techniques work for a particular set of features, which, in our opinion, is more interesting, from a scientific point of view, than merely reporting the winners.

In the following, we briefly recall the previous competition format, before discussing the new one: we will illustrate categories and tracks, and present the scoring system in detail, along with a proper overview of general rules; eventually, we will provide the reader with some information about the competition infrastructure.

Previous Competition format. The Fourth ASP Competition (Alviano et al. 2013a) consisted of two different tracks, adopting the distinction between Model&Solve and System Track. Both tracks featured a selected suite of domains, chosen by means of an open *Call for Problems* stage. The System Track was conceived with the aim of fostering the language standardization, and let the competitors compare each other on fixed encodings under predefined conditions, excluding, e.g., domain-tailored evaluation heuristics and custom problem encodings. The Model&Solve Track was instead left open to any system loosely based on a declarative specification language, with no constraints on the declarative language. Indeed, the spirit of this track was to encourage the development of new declarative constructs and/or new modeling paradigms and to foster the exchange of ideas between communities in close relationships with ASP, besides the stimulation of the development of new ad-hoc solving methods, refined problem specifications and solving heuristics, on a per domain basis.

New Competition format. First of all, given what has already been discussed above, along with the fact that we aim to focus on language features, there is no Model&Solve Track in this edition. The competition will take place, then, in the spirit of the former System Track: it was open to any general-purpose solving system, provided it was able to parse the ASP-Core-2 input format. Encodings for each problem selected for the competition, along with instance data, have been chosen by the Organizing Committee (see Section 3); participant systems will be run in a uniform setting, on each problem and instance thereof (*out-of-the-shelf* setting). Furthermore, sub-tracks are not based on “complexity” of problems (as in past events), but rather take into consideration language features, as already discussed above.

Competition Categories. The competition consists of *two categories*, depending on the computational resources allocated 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. As stated by the Call for Participation, according to the availability of benchmarks, to submitted systems, and to participants feedback, 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.

Scoring system. The scoring system adopted simplifies the ones from the Third and Fourth ASP Competitions. In particular, it balances the following factors:

- Problems are always weighted equally.
- If a system outputs an incorrect answer to some instance of a problem, this should invalidate its score for the problem, even if all 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 will hence consist 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. Let M be the number of participant systems; then, the 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 solvers 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 over sums of scores, the sum of run-times is taken into account as a tie-breaker, in favor, of course, of the system whose run-time is smaller.

Verification of Answers. Each benchmark domain P is equipped with a checker program C_P that takes as input both an instance I and a corresponding witness solution A , and it is such that $C_P(A, I) = \text{true}$ in case A is a valid witness for I w.r.t. P .

Let us suppose that a system S is faulty for an instance I of a problem P ; then, there are two possible ways to detect incorrect behavior, and subsequently disqualify system S for P :

- S produces an answer A , but A is not a correct solution for I . This case is detected by checking the outcome of $C_P(A, I)$.

- S recognizes instance I as unsatisfiable, but I actually has some witness solution. In this case, it is checked whether another system S' produced a solution A' for which $C_P(A', I)$ is true.

A case of general failure (e.g., “out of memory” errors or some other abrupt system failures) does not imply disqualification on a given benchmark.

When dealing with Optimization problems, checkers produce also the cost of the (last) witness. This latter value is considered when computing scores and assessing answers of systems. Given an instance I for an Optimization problem P , in general, the cost of best witnesses found over all participants is taken as the *imperfect optimum*. When a system S marks its witness A as optimal for I :

- if no other system finds a better witness for I , A is pragmatically assumed to be optimal;
- if the cost of A turns out to be different from the *imperfect optimum* for I , S is disqualified on P .

Software and Hardware settings. The competition is run on a Debian Linux server (64bit kernel), featuring Intel Xeon X5365 Processors with 8MB of cache and 16GB of RAM. Time and memory for each run are limited to 10 minutes and 6GB, respectively. Participants can exploit up to 8 cores in the **MP** category, whereas the execution is constrained to 1 core in the **SP** category. The execution environment is composed of a number of scripts, and performance is measured using the *pyrunlim* tool⁷.

3 Benchmark Suite

The benchmark domains in this edition of the ASP Competition largely coincide with the ones from 2013. Although ASP-Core-2 encodings had already been made available one year ago, most participants then lacked preparation time and could not submit appropriate systems. In fact, half of the systems in 2013 were run on “equivalent” encoding reformulations in legacy formats. In the meantime, however, the ASP-Core-2 compliant grounder GRINGO-4 became available, furnishing an off-the-shelf front-end for solvers operating at the propositional level.

As described in Section 2, the benchmarks in the Fifth ASP Competition are categorized into tracks based on the language features utilized by encodings. Table 1 provides a respective overview, grouping benchmark domains in terms of language features in the ASP-Core-2 encodings from 2013. That is, the 2013 encodings for *Labyrinth* and *Stable Marriage* belong to the Basic Decision track, and the “D” entries in the second column indicate that both domains deal with Decision problems. The Advanced Decision track includes the sixteen 2013 encodings for the domains in rows from *Bottle Filling* to *Weighted-Sequence Problem*. Among them, the *Reachability* domain aims at Query answering, as indicated by “Q” in the second column. The following four rows marked with “O” provide the domains in the Optimization track. Finally, the last four rows give the encodings in the Unrestricted track, where *Abstract Dialectical Frameworks* is an Optimization problem and *Strategic Companies* deals with Query answering.

The third column of Table 1 indicates particular language features of the encodings from the Fourth ASP Competition. While merely normal rules and comparison operators, considered as “basic” features, are used for *Stable Marriage*, the Basic Decision encoding for *Labyrinth*

⁷ <https://github.com/alviano/python/>

Table 1. Benchmark Suite of the Fifth ASP Competition

Domain	P	2013 Encoding	2014 Encoding
<i>Labyrinth</i>	D	basic, non-tight	basic, non-tight
<i>Stable Marriage</i>	D	basic	basic
<i>Bottle Filling</i>	D	aggr	aggr, choice
<i>Graceful Graphs</i>	D	choice [#]	choice [#]
<i>Graph Colouring</i> [*]	D	disj	basic
<i>Hanoi Tower</i> [*]	D	disj	basic
<i>Incremental Scheduling</i>	D	aggr, choice [#]	aggr, choice [#]
<i>Knight Tour with Holes</i> [*]	D	disj, non-tight	basic, non-tight
<i>Nomystery</i>	D	aggr, choice [#]	choice [#]
<i>Partner Units</i>	D	aggr, disj, non-tight	aggr, choice
<i>Permutation Pattern Matching</i>	D	choice [#]	choice
<i>Qualitative Spatial Reasoning</i>	D	choice [#] , disj	disj
<i>Reachability</i>	Q	non-tight	n/a
<i>Ricochet Robots</i>	D	choice [#]	aggr, choice [#]
<i>Sokoban</i>	D	aggr, choice [#]	choice [#]
<i>Solitaire</i>	D	choice [#]	aggr, choice [#]
<i>Visit-all</i> [*]	D	aggr, choice [#]	basic
<i>Weighted-Sequence Problem</i>	D	choice [#]	aggr, choice
<i>Connected Still Life</i>	O	aggr, choice [#] , non-tight	aggr, choice, non-tight
<i>Crossing Minimization</i>	O	disj	aggr, choice
<i>Maximal Clique</i>	O	disj	basic
<i>Valves Location</i>	O	aggr, choice [#] , non-tight	aggr, choice [#] , non-tight
<i>Abstract Dialectical Frameworks</i>	O	aggr, disj, level, non-tight	aggr, disj, level, non-tight
<i>Complex Optimization</i>	D	choice, disj, non-tight	choice, disj, non-tight
<i>Minimal Diagnosis</i>	D	disj, non-tight	disj, non-tight
<i>Strategic Companies</i>	Q	disj, non-tight	n/a

induces “non-tight” ground instances with positive recursion among atoms (Fages 1994; Erdem and Lifschitz 2003). The use of aggregates like `#count`, `#sum`, `#max`, and `#min` (Faber et al. 2008; Calimeri et al. 2013), e.g., in the Advanced Decision encoding for *Bottle Filling*, is indicated by an “aggr” entry. Moreover, “disj” denotes proper disjunctions in rule heads (Gelfond and Lifschitz 1991; Eiter and Gottlob 1995), as utilized in the 2013 encoding for *Graph Colouring*. Choice rules (Simons et al. 2002; Calimeri et al. 2013) are indicated by a “choice” entry, where the superscript “[#]” stands for non-trivial lower and/or upper bounds on the number of chosen atoms, e.g., used in the Advanced Decision encoding for *Graceful Graphs*. Unlike that, the choice rules for *Complex Optimization* in the Unrestricted track are unbounded, and thus “[#]” is omitted in its row. Finally, *Abstract Dialectical Frameworks* is the only Optimization problem in the Fifth ASP Competition for which more than one “level” (Simons et al. 2002; Leone et al. 2006) is used in the encoding.

Compared to the Fourth ASP Competition, we decided to drop the *Chemical Classification* domain, whose large encoding imposed primarily a grounding bottleneck. On the other hand, we reintroduced the application-oriented *Partner Units* domain, reusing encodings and instances submitted to the Third ASP Competition. In order to furnish a novel benchmark collection for this year, we also devised new encoding variants utilizing the language features indicated in the fourth column of Table 1. The 2014 encodings for the domains marked with “^{*}” omit advanced language features of their 2013 counterparts, so that the Basic Decision track on new encodings

comprises six benchmark domains. By evaluating the participant systems on previous as well as new encodings, we hope to gain insights regarding the impact of encodings on system performance. In fact, for all domains but *Reachability* and *Strategic Companies* aiming at Query answering, we could make substantial modifications to previously available encodings.

The instances to run in the Fifth ASP Competition have been randomly selected from the suites submitted in 2013 (or 2011 for *Partner Units*), using the concatenation of winning numbers from the EuroMillions lottery of Tuesday, 22nd April 2014, as random seed. In this way, twenty instances were picked per domain in order to assess the participant systems both on the encodings from 2013 as well as their new variants.

4 Participants

In this section, we briefly present all participants; we refer the reader to the official competition website (Calimeri et al. 2014) for further details.

The competition featured 16 systems coming from three teams:

- The Aalto team from the Aalto University submitted nine solvers, working by means of translations (Bomanson and Janhunen 2013; Gebser et al. 2014; Liu et al. 2012; Nguyen et al. 2011). Three systems, LP2SAT3+GLUCOSE, LP2SAT3+LINGELING, and LP2SAT3+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 part is expressed in terms of bit-vector logic or acyclicity checking, respectively, supported by the back-end SMT solvers of the LP2BV2+BOOLECTOR and LP2GRAPH systems. While the aforementioned systems do not support optimization and participate in the Basic and Advanced Decision tracks (#1 and #2) only, LP2MAXSAT+CLASP, LP2MIP2, and LP2MIP2-MT, running CLASP as a Max-SAT solver or the Mixed Integer Programming solver CPLEX as back-ends, respectively, compete in the Optimization track (#3) as well. Finally, LP2NORMAL2+CLASP normalizes aggregates (of up to certain size) and uses CLASP as back-end ASP solver; LP2NORMAL2+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 (*Reachability* and *Strategic Companies*). The systems LP2SAT3+PLINGELING-MT and LP2MIP2-MT exploit multi-threading and run in the **MP** category, while the other, sequential systems participate in the **SP** category.
- The Potassco team from the University of Potsdam submitted the sequential system CLASP (Gebser et al. 2012a) in the **SP** category and the multi-threaded CLASP-MT system (Gebser et al. 2012b) in the **MP** category. Both systems utilize GRINGO-4 for grounding and CLASP, a native ASP solver for (extended) disjunctive logic programs based on conflict-driven learning, as back-end. The systems participate in all tracks, except for the Query problems.
- The Wasp team from the University of Calabria submitted five incarnations of WASP (Alviano et al. 2013b; Alviano et al. 2014), a native ASP solver built upon techniques originally introduced in SAT, yet extended and combined with techniques specifically designed for solving disjunctive logic programs, in the **SP** category. Unlike WASP-1, utilizing a prototype version of DLV (to cope with the ASP-Core-2 language) for grounding, WASP-2 relies on GRINGO-4 and further differs from WASP-1 in the implementation of support inferences and program simplifications. Moreover, WASP-1.5 is a hybrid system combining WASP-1 and WASP-2, basically switching between them depending on whether a logic program is

HCF or subject to a Query. While WASP-1 and WASP-1.5 compete in all domains and tracks, WASP-2 does not participate in the Unrestricted track (#4). Finally, the WASP-WMSU1-ONLY-WEAK and WASP-WPM1-ONLY-WEAK systems are specifically designed for solving optimization problems and thus participate in the Optimization track (#3) only.

In summary, similarly to past competitions, we can identify two main ASP solving approaches:

- “native” systems, which exploit techniques purposely conceived/adapted for dealing with logic programs under the stable models semantics, and
- “translation-based” systems, which (roughly) at some stage of the evaluation process produce an intermediate specification in some different formalism, which is then fed to a corresponding solver.

The solvers submitted by the Potassco and Wasp teams as well as the LP2NORMAL2+CLASP system by the Aalto team rely on the first approach, while the remaining systems by the Aalto team are of the second kind.

It is worth mentioning that, in order to assess the improvements in system implementation, we also ran a selection of systems that were submitted in the Fourth ASP Competition. In particular, we considered one system per team selected according to the following criteria: (*i*) it features an updated version in this year’s edition, (*ii*) it is compliant with ASP-Core-2, and (*iii*) it performed best in the Fourth ASP Competition among the systems submitted by the same team.

5 Conclusions

The Fifth ASP Competition is jointly organized by the University of Calabria (Italy), the Aalto University (Finland), and the University of Genova (Italy), and is affiliated with the 30th International Conference on Logic Programming (ICLP 2014). The main goals of the Fifth ASP Competition are to measure the advances of the state of the art in ASP solving and to push the adoption of the ASP-Core-2 standard format. In this paper, the design of the fifth edition of the ASP Competition was presented, along with an overview of the participants.

The results will be announced in Vienna at ICLP 2014, which is part of the Federated Logic Conference. Participants will be awarded in a ceremony organized by the FLoC Olympic Games at the Vienna Summer of Logic, taking place on Monday, 21th July 2014.

References

ALVIANO, M., CALIMERI, F., CHARWAT, G., DAO-TRAN, M., DODARO, C., IANNI, G., KRENNWALLNER, T., KRONECKER, M., OETSCH, J., PFANDLER, A., PÜHRER, J., REDL, C., RICCA, F., SCHNEIDER, P., SCHWENGERER, M., SPENDIER, L., WALLNER, J., AND XIAO, G. 2013a. The fourth answer set programming competition: Preliminary report. In *Proceedings of the 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR’13)*. Springer-Verlag, 42–53.

ALVIANO, M., DODARO, C., FABER, W., LEONE, N., AND RICCA, F. 2013b. Wasp: A native ASP solver based on constraint learning. In *Proceedings of the 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR’13)*. Springer-Verlag, 54–66.

ALVIANO, M., DODARO, C., AND RICCA, F. 2014. Preliminary report on WASP 2.0. In *Proceedings of the 15th International Workshop on Non-Monotonic Reasoning (NMR’14)*.

BARAL, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.

BOMANSON, J. AND JANHUNEN, T. 2013. Normalizing cardinality rules using merging and sorting constructions. In *Proceedings of the 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR’13)*. Springer-Verlag, 187–199.

CALIMERI, F., FABER, W., GEBSER, M., IANNI, G., KAMINSKI, R., KRENNWALLNER, T., LEONE, N., RICCA, F., AND SCHAUB, T. 2013. ASP-Core-2: 4th ASP competition official input language format. <https://www.mat.unical.it/aspcomp2013/files/ASP-CORE-2.01c.pdf>.

CALIMERI, F., GEBSER, M., MARATEA, M., AND RICCA, F. 2014. The fifth answer set programming competition homepage. <https://www.mat.unical.it/aspcomp2014/>.

DAL PALÙ, A., DOVIER, A., PONTELLI, E., AND ROSSI, G. 2009. GASP: Answer set programming with lazy grounding. *Fundamenta Informaticae* 96, 3, 297–322.

EITER, T., FABER, W., LEONE, N., AND PFEIFER, G. 2000. Declarative problem-solving using the DLV system. In *Logic-Based Artificial Intelligence*. Kluwer Academic Publishers, 79–103.

EITER, T. AND GOTTLÖB, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence* 15, 3-4, 289–323.

EITER, T., GOTTLÖB, G., AND MANNILA, H. 1997. Disjunctive Datalog. *ACM Transactions on Database Systems* 22, 3, 364–418.

EITER, T., IANNI, G., AND KRENNWALLNER, T. 2009. Answer set programming: A primer. In *Tutorial Lectures of the 5th International Summer School on Semantic Technologies for Information Systems (Reasoning Web)*. Springer-Verlag, 40–110.

ERDEM, E. AND LIFSHITZ, V. 2003. Tight logic programs. *Theory and Practice of Logic Programming* 3, 4-5, 499–518.

FABER, W., PFEIFER, G., LEONE, N., DELL'ARMI, T., AND IELPA, G. 2008. Design and implementation of aggregate functions in the DLV system. *Theory and Practice of Logic Programming* 8, 5-6, 545–580.

FAGES, F. 1994. Consistency of Clark's completion and existence of stable models. *Journal of Methods of Logic in Computer Science* 1, 1, 51–60.

GEBSER, M., JANHUNEN, T., AND RINTANEN, J. 2014. Answer set programming as SAT modulo acyclicity. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*.

GEBSER, M., KAUFMANN, B., AND SCHAUB, T. 2012a. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence* 187–188, 52–89.

GEBSER, M., KAUFMANN, B., AND SCHAUB, T. 2012b. Multi-threaded ASP solving with clasp. *Theory and Practice of Logic Programming* 12, 4-5, 525–545.

GEBSER, M., KAUFMANN, B., AND SCHAUB, T. 2013. Advanced conflict-driven disjunctive answer set solving. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI'13)*. IJCAI/AAAI Press, 912–918.

GELFOND, M. AND LEONE, N. 2002. Logic programming and knowledge representation – the A-Prolog perspective. *Artificial Intelligence* 138, 1-2, 3–38.

GELFOND, M. AND LIFSHITZ, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9, 365–385.

GIUNCHIGLIA, E., LIERLER, Y., AND MARATEA, M. 2006. Answer set programming based on propositional satisfiability. *Journal of Automated Reasoning* 36, 4, 345–377.

JANHUNEN, T., NIEMELÄ, I., SEIPEL, D., SIMONS, P., AND YOU, J. 2006. Unfolding partiality and disjunctions in stable model semantics. *ACM Transactions on Computational Logic* 7, 1, 1–37.

LEONE, N., PFEIFER, G., FABER, W., EITER, T., GOTTLÖB, G., PERRI, S., AND SCARCELLO, F. 2006. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic* 7, 3, 499–562.

LIFSHITZ, V. 1999. Answer set planning. In *Proceedings of the 16th International Conference on Logic Programming (ICLP'99)*. MIT Press, 23–37.

LIN, F. AND ZHAO, Y. 2004. ASSAT: Computing answer sets of a logic program by SAT solvers. *Artificial Intelligence* 157, 1-2, 115–137.

LIU, G., JANHUNEN, T., AND NIEMELÄ, I. 2012. Answer set programming via mixed integer programming. In *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR'12)*. AAAI Press, 32–42.

MAREK, V. AND TRUSZCZYŃSKI, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm – A 25-Year Perspective*. Springer-Verlag, 375–398.

MARIËN, M., WITTOCX, J., DENECKER, M., AND BRUYNNOOGHE, M. 2008. SAT(ID): Satisfiability of propositional logic extended with inductive definitions. In *Proceedings of the 11th International Conference on Theory and Applications of Satisfiability Testing (SAT'08)*. Springer-Verlag, 211–224.

NGUYEN, M., JANHUNEN, T., AND NIEMELÄ, I. 2011. Translating answer-set programs into bit-vector logic. In *Proceedings of the 19th International Conference on Applications of Declarative Programming and Knowledge Management (INAP'11) and the 25th Workshop on Logic Programming (WLP'11)*. Springer-Verlag, 95–113.

SIMONS, P., NIEMELÄ, I., AND SOININEN, T. 2002. Extending and implementing the stable model semantics. *Artificial Intelligence* 138, 1-2, 181–234.