The Second Answer Set Programming Competition

Marc Denecker[1], Joost Vennekens[1], Stephen Bond[1], Martin Gebser[2], and Mirosław Truszczyński[3]

[1] Department of Computer Science, Katholieke Universiteit Leuven Celestijnenlaan 200A, B-3001-Heverlee, {marcd, joost, stephen}@cs.kuleuven.be
[2] Institut für Informatik, Universität Potsdam, August-Bebel-Str. 89, D-14482 Potsdam, Germany, gebser@cs.uni-potsdam.de
[3] Department of Computer Science, University of Kentucky, Lexington, KY 40506-0046, USA, mirek@cs.uky.edu

Abstract. This paper reports on the *Second Answer Set Programming Competition*. The competitions in areas of Satisfiability checking, Pseudo-Boolean constraint solving and Quantified Boolean Formula evaluation have proven to be a strong driving force for a community to develop better performing systems. Following this experience, the Answer Set Programming competition series was set up in 2007, and ran as part of the International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). This second competition, held in conjunction with LPNMR 2009, differed from the first one in two important ways. First, while the original competition was restricted to systems designed for the *answer set programming language*, the sequel was open to systems designed for other modeling languages, as well. Consequently, among the contestants of the second competition were a CLP(FD) team and three model generation systems for (extensions of) classical logic. Second, this latest competition covered not only satisfiability problems but also optimization ones. We present and discuss the set-up and the results of the competition.

1 Introduction

In many real-life problems, we search for objects of complex nature — plans, schedules, assignments. Several research areas within computer science, operations research and mathematics are concerned with the development of systems that compute such objects from their specifications. Researchers in these areas design and study languages to describe objects of interest, as well as algorithms to extract them from these descriptions. Depending on the language and the area, such objects are called "answer sets", "valuations", "structures", "interpretations", "functions" or "arrays". Answer Set Programming (ASP), propositional Satisfiability (SAT) and Constraint Programming (CP) are arguably the three most prominent areas developing such languages and techniques.

In the context of logic, it is often the case that once constraints on objects to search for are given as formulas (rules), models of the resulting theory determine answers to the search problem — objects satisfying the constraints. For instance, each model of a theory specifying a scheduling domain typically defines a correct schedule. Thus, a model generator applied to such a theory will solve the corresponding scheduling problem. This idea of model generation as a declarative problem solving paradigm has been pioneered in the area of ASP [1–3].

ASP has three fundamental characteristics: a *modeling language* based on the syntax of logic programs, the use of the *answer set semantics* [4] to interpret programs in that language, and a *problem-solving methodology* in which a program is written so that its answer sets provide solutions. ASP has its origins in Logic Programming (LP) [5, 6], in particular in the attempts in the 1980s to develop a declarative semantics for logic programs with negation and to turn this logic into a formalism suitable for knowledge representation. Gelfond and Lifschitz sought inspiration in nonmonotonic reasoning [7, 8] and proposed to interpret logic programs as special default theories under the semantics of Reiter [8]. Based on this view, they developed the *stable model semantics* for logic programs [9], and extended it later to the *answer set semantics* for disjunctive logic programs with classical negation [4], which is the *core* ASP language today.

The area made a major leap in 1997, when the first two systems to compute answer sets of logic programs were developed: *dlv* [10] and *smodels* [11]. These systems demonstrated that effective tools for processing answer set programs are possible. Following that, in 1999, Marek and Truszczyński [1] and Niemelä [2] proposed answer set computation as a new declarative problem solving paradigm, and Lifschitz dubbed the area answer set programming [12]. It turned out that a rich class of problems could be modeled elegantly as answer set programs according to this paradigm. That became a strong driving force for the development of fast computational techniques in ASP, and for studies of practical applications where the ASP tools could be used effectively.

Experience in areas concerned with checking propositional or Pseudo-Boolean (PB) satisfiability and evaluating Quantified Boolean Formulas (QBF) shows that a programming competition gives an effective incentive to the research community to work on developing better performing systems. ASP sought to emulate that experience. The first preliminary competitions for ASP systems were held in 2002 and 2005 at two Dagstuhl meetings [13]. In 2007, the *First Answer Set Programming System Competition* [14] was organized as part of LPNMR. That competition consisted of four tracks. In three tracks, solvers were tested on prespecified answer set programs. In the SCore-v and SCore tracks, input consisted of a ground logic program, respectively with and without disjunction; in the SLparse track, the programs used *lparse*'s output language (including aggregates). In the fourth track, called *Model, Ground, Solve* (MGS), contestants encoded problems in a language of their choice. Ten teams competed. The *clasp* solver won the SCore and SLparse tracks, and the competition version of *dlv* won in SCore-v and MGS.

In the past two years, some important developments have occurred in ASP. First, as it is clear from the results of the second ASP competition (see Section 5), existing systems have improved considerably, both in available language features and in the speed of the solvers. In addition, new systems have been built, and more teams competed. Clearly, the first ASP competition has had its desired effect! Second, the ASP community has been gradually opening up to other domains. The fields of SAT [15], SAT Modulo Theories (SMT) [16], CP [17] and, in some way, also Abductive Logic Programming (ALP) [18] are in one key respect very close to ASP. Namely, the ASP declarative problem solving paradigm does not depend on the answer set programming language but applies to and, in fact, has been used with other declarative languages too. For example, Mitchell and Ternovska [19] proposed to use model expansion (a form of model generation) for (extensions of) first-order logic as a declarative problem solving paradigm for NP search problems. (Ground) abduction in ALP is similar to model generation [18], and integrations of ALP and CLP have been used for planning, scheduling and constraint solving problems [20, 21]. Moreover, workshops organized by the ASP community such as Answer Set Programming and Other Computing Paradigms (ASPOCP), held in conjunction with the International Conferences on Logic Programming in Udine in 2008 and Pasadena in 2009, and Logic and Search (LaSh), held in Seattle in 2006 and Leuven in 2008, explicitly aimed to bring together researchers from all fields that share the problem solving methodology based on model generation.

The *Second Answer Set Programming Competition*, organized in conjunction with LPNMR 2009, further fortified this trend. Having a competition that would be open not only to the ASP community but also to the communities of SAT, LP, CP, etc. was an important objective of the program committee, and a key condition for accepting the charge of organizing it put forth by the members of the Knowledge Representation and Reasoning (KRR) group at the K.U. Leuven. The underlying idea was that only such an open competition can lead to progress and result in new insights into the strengths of different technologies in the context of diverse applications. An important ramification was that the competition had to be restricted to the *Model and Solve* track, as only that mode allows teams using tools based on different languages and logics to compete. On the other hand, the scope of the competition was expanded to include not only decision problems but also optimization ones.

The first step in organizing the competition was to collect *benchmarks* — specific problems together with sets of instances. Many researchers contributed, and we gratefully acknowledge their efforts. In total, we received 38 benchmarks, nine of which were optimization problems. Most benchmarks came from ASP researchers, some came from the Constraint Logic Programming (CLP) [22] community. We split these benchmarks into four categories: the *P decision class* consisting of polynomially solvable decision problems, the *NP decision class* consisting of decision problems in NP not known to be polynomially solvable, the *global decision class* consisting of all problems in the first two groups and of one Σ_2^p-complete problem and, finally, the *optimization class*.

The competition was a success. Sixteen teams competed, with nine of them for the first time! Twelve teams used the ASP language and tools. The call for participation to other communities had limited success. Three teams, Enfragmo, IDP and amsolver, used model generators/expanders for first-order logic extended with aggregates, arithmetic and, in the case of IDP and amsolver, also with inductive definitions. However, the amsolver team competed only on eight benchmarks. One team, BPSolver-CLP(FD), led by Neng-Fa Zhou, used the CLP(FD) solver B-Prolog. There were no SAT, PB nor SMT participants, but SAT, PB or SMT solvers were used in many systems.

It is important to note that the second ASP competition had a policy of openness about benchmark solutions (programs encoding problem specifications). Several teams made their codes publicly available, allowing other teams to profit from their efforts. For example, at least eight teams used the benchmark solutions by the Potassco team available at [23].

The paper is organized as follows. In the next section, we give an outline of the competition and specify its format. In Section 3, we discuss the collection of benchmarks, the ranking system and the competition platform. In Section 4, we introduce the teams that competed. We present the results in Section 5. Finally, we close by summarizing our experience and outlining potential future improvements. For more detailed information on the results of this competition, we refer to [24].

2 Outline of the Competition

The second ASP competition had one track, referred to as *Model and Solve*, and included both decision and optimization problems.

The first step of the competition was a call for submitting benchmark problems. This call was sent to research communities of ASP, LP, SAT and CP. Benchmark authors were invited to provide an (informal) problem description, input and output predicates, a set of instances and a *checker* program to verify the correctness of solutions (see Section 3 for details).

Contributed benchmarks were then evaluated by the organizers and the program committee (from now on referred to simply as organizers). The overall benchmark pool was a mixture of many diverse decision problems and several optimization problems. There were fundamental differences between decision benchmarks, and some seemed to favor particular types of systems quite strongly. To understand better which features were useful for particular types of problems, the organizers decided to split decision benchmarks into the following sub-categories:

Decision-in-P class: Problems that can be solved in polynomial time. While they are simple, the challenge is the sheer size of the instances requiring highly optimized grounding techniques.

Decision-in-NP class: Problems that are in NP but are not known to be in P. They are the problems that require highly effective search algorithms.

Decision-global class: Problems from the previous two groups and one Σ_2^p-complete problem, the well-known Strategic Companies problem. The goal of this category is to evaluate solving systems with respect to both their grounding and search effectiveness on a broad range of problems of varying complexity.

In the competition, each team was to solve each benchmark problem for several instances. To this end, each team had to submit a script for each benchmark on which the team wanted to compete. This script was to call the solver and apply it to the benchmark encoding and a problem instance given through standard input. During the competition, this script was called repeatedly for each instance of the benchmark.

For both decision and optimization problems, a problem instance was represented as a sequence of atomic clauses (an atom followed by a period) over predicates from the input vocabulary. The output depended on the type of problem:

Decision problems: The script should write the following to standard output:

- **UNSATISFIABLE**, if the instance has no solution.

- A sequence of atomic clauses in the output vocabulary, if the instance is satisfiable. Such a sequence should represent a ”witness” to the satisfiability of the instance, that is, it should be the set of atoms over the output predicates in an answer set or model (for first-order logic formalisms) determining a solution for the instance. The format is the same as with the input except that it must not contain line breaks.
- **UNKNOWN**, if the solver decides to give up before timeout.

Optimization problems: The script should output the following:

- **UNSATISFIABLE**, in case of an unsatisfiable instance.
- A series of witnesses of the search problem, if the instance is satisfiable. The format of a witness is the same as for decision problems. Witnesses are separated by line breaks. The last (and hopefully best) witness is considered as the generated solution.
- **OPTIMUM FOUND**, if the instance is satisfiable and the solver can ascertain the optimality of the last produced witness.

Remark. The use of separate scripts for different benchmarks gives teams the freedom to fine-tune parameter settings to specific benchmarks, or even to use different solvers. During the installation phase of the competition, this raised some controversy among the participants. Our point of view is that for an open competition, there is no option but to offer this freedom. For example, a SAT team should have the freedom to develop for each benchmark a program to compute the CNF theory corresponding to an instance. However, only two teams, Potassco and BPSolver-CLP(FD) made extensive use of the facility to tune systems towards benchmarks. Potassco used various grounders and solvers, with different parameter settings. The BPSolver-CLP(FD) team used B-Prolog’s control structures to implement various labeling strategies. Other teams used the same combination of systems and parameters for all decision problems and for all optimization problems. This distinction is relevant for the interpretation of the competition results and will be taken into account in Section 5.

3 Benchmarks

The collected benchmarks constitute the result of efforts by many researchers, mostly from the ASP community; some were provided by the BPSolver-CLP(FD) team. The author(s) of a benchmark problem provided us with the following information:

- a clear non-ambiguous (informal) problem description,
- a specification of the input and output vocabulary,
- for optimization problems, an integer-valued cost function to be minimized,
- a set of instances,
- a checker program to test the correctness of witnesses and, for optimization problems, to additionally compute the cost.

Verifying unsatisfiability of an NP-hard decision problem or optimality of a witness of an NP-hard optimization problem is intractable in general (assuming the polynomial

Benchmark	Class	Contributors	#Instances
HydraulicPlanning	Decision,P	M. Gelfond, R. Morales and Y. Zhang	15
HydraulicLeaking	Decision,P	M. Gelfond, R. Morales and Y. Zhang	15
CompanyControls	Decision,P	Mario Alviano	15
GrammarBasedInformationExtraction	Decision,P	Marco Manna	29
Reachability	Decision,P	Giorgio Terracina	15
BlockedNQueens	Decision,NP	G. Namasivayam and M. Truszczyński	29
Sokoban	Decision,NP	Wolfgang Faber	29
15Puzzle	Decision,NP	L. Liu, M. Truszczyński and M. Gebser	16
HamiltonianPath	Decision,NP	L. Liu, M. Truszczyński and M. Gebser	29
SchurNumbers	Decision,NP	L. Liu, M. Truszczyński and M. Gebser	29
TravellingSalesperson	Decision,NP	L. Liu, M. Truszczyński and M. Gebser	29
WeightBoundedDominatingSet	Decision,NP	L. Liu, M. Truszczyński and M. Gebser	29
Labyrinth	Decision,NP	Martin Gebser	29
GeneralizedSlitherlink	Decision,NP	Wolfgang Faber	29
HierarchicalClustering	Decision,NP	G. Namasivayam and M. Truszczyński	12
ConnectedDominatingSet	Decision,NP	G. Namasivayam and M. Truszczyński	21
GraphPartitioning	Decision,NP	G. Namasivayam and M. Truszczyński	13
Hanoi	Decision,NP	G. Namasivayam, M. Truszczyński and G. Terracina	15
Fastfood	Decision,NP	Wolfgang Faber	29
WireRouting	Decision,NP	G. Namasivayam and M. Truszczyński	23
Sudoku	Decision,NP	Neng-Fa Zhou	10
DisjunctiveScheduling	Decision,NP	Neng-Fa Zhou	10
KnightTour	Decision,NP	Neng-Fa Zhou	10
ChannelRouting	Decision,NP	Neng-Fa Zhou	11
EdgeMatching	Decision,NP	Martin Brain	29
GraphColouring	Decision,NP	Martin Brain	29
MazeGeneration	Decision,NP	Martin Brain	29
Solitaire	Decision,NP	Martin Brain	27
StrategicCompanies	Decision,Σ_2^P	M. Alviano, M. Maratea and F. Ricca	17
GolombRuler	Optimization	Martin Brain	24
MaximalClique	Optimization	Johan Wittocx	29
15PuzzleOptimize	Optimization	L. Liu, M. Truszczyński and M. Gebser	16
TravellingSalespersonOptimize	Optimization	L. Liu, M. Truszczyński and M. Gebser	29
WeightBoundedDominatingSetOptimize	Optimization	L. Liu, M. Truszczyński and M. Gebser	29
LabyrinthOptimize	Optimization	Martin Gebser	28
SokobanOptimize	Optimization	Wolfgang Faber	29
FastfoodOptimize	Optimization	Wolfgang Faber	29
CompanyControlsOptimize	Optimization	Mario Alviano	15

Table 1. Benchmarks of the Second Answer Set Programming Competition

hierarchy does not collapse). For this reason, checkers only had to test the correctness of witnesses, and not of the answers **UNSATISFIABLE** or **OPTIMUM FOUND**.

Table 1 gives an overview of all benchmarks of the competition. For each problem, the entry specifies the author(s), the category and the number of instances used in the competition.

3.1 Detecting Errors

A team was disqualified for a benchmark as soon as an error was detected for one of the benchmark instances. As we wrote before, checker programs did not test correctness of **UNSATISFIABLE** and **OPTIMUM FOUND**. To check the correctness of those answers, we used the following incomplete strategy. An erroneous **UNSATISFIABLE** answer for an instance is reported if another solver finds a correct witness. An erroneous **OPTIMUM FOUND** answer for an instance is reported if another solver finds a strictly better solution. This method can only detect an erroneous **UNSATISFIABLE** answer if at least one solver finds a correct witness. A similar comment holds for erroneous **OPTIMUM FOUND** answers.

3.2 Ranking System

Ranking on decision problems. The ranking system for the three decision problem tracks is very similar to the one used in the first ASP competition. The primary criterion is the number of instances solved. For each problem instance of each benchmark, the teams are given a time bound of 600 seconds to solve the instance. Each participant is assigned points according to the number of instances that can be solved within this time bound. For each benchmark, a team is awarded points as follows:

- No points, if the solver makes an error on one of the instances of the benchmark.
- Otherwise, $20(S/N)$ points, where S is the number of instances solved by the team and N is the number of instances of this benchmark that were solved by at least one team.

This boils down to the weighted sum over solved instances, the weight of a solved instance from a particular benchmark being $1/N$, where N is the number of instances of this benchmark that were solved by at least one team. By defining weights in this way (normalizing with respect to the total number of instances solved within a benchmark), the method prevents benchmarks with (very) many instances from dominating the score.

Ties are resolved by comparing actual running time where, for **UNKNOWN** answers, the timeout time is taken as the running time. During the competition, system failures frequently occurred, most often due to systems running out of memory. When a system failure occurred on an instance, it was treated as an **UNKNOWN** answer and assigned the timeout time as the running time.

Ranking on optimization problems. The ranking of teams on optimization problems depends on the quality of the solution they find. We recall that each problem has a cost function mapping a solution to an integer. Solutions with minimal cost are desirable for each instance. The checker programs of the optimization problems not only check the correctness of a witness for an instance, but also compute its cost. Teams are awarded points proportionally to the distance between the cost of the returned solution and the lowest cost of any solution found by any team. Ties are resolved according to the running time. Extra points are given to systems that correctly (or not incorrectly) return the keyword **OPTIMUM FOUND**.

A team could earn 100 points per instance of a benchmark. These points were distributed as follows:

- No points, if the solver makes an error on one of the instances of the benchmark.
- 100 points, if the solver correctly outputs **UNSATISFIABLE**.
- If the solver produces a correct witness:
  - it is awarded an initial 25 points;

- in addition, it can receive up to 50 points for the quality of the solution. Let M be the lowest cost of a solution that was produced by any solver for this instance. If the cost of the solution produced by the team is Q, then the team is awarded $50(M/Q)$ points;
- in addition, 25 points are awarded if the solver correctly outputs **OPTIMUM FOUND**.

The ranking score for this category is a weighted sum of the earned points. The weight for an instance in a benchmark is $1/N$, where N is the number of solved instances from that benchmark (by any team). This is done to prevent a disproportionate effect benchmarks with many instances might otherwise have on the scores.

Global rankings. We also used two global rankings: one for all decision problems, and another one for all decision and optimization problems. The weighting of individual decision benchmarks within the global decision category is calculated in the same fashion as described above. The global ranking awards each team a score that is the average of its score for the global decision raking and its score for the optimization ranking (divided by 100). This gives equal importance to both categories of the competition.

3.3 Competition Software and Platform

The competition was run on a pool of computers of the DTAI research group of the K.U. Leuven. The system consisted of one network server through which participants could login via ssh, one database server for storing state and results, and a pool of five identical Linux machines reserved for the competition and only accessible through the network server. One of these machines was reserved for participants to install and test solutions, while the other four served for the actual testing of solvers on benchmark instances. The system was installed and maintained by the system group of the Computer Science Department of the K.U. Leuven, in particular Bart Swennen and Kris Vangeneugden.

The competition software was derived from software that had been developed for *De Vlaamse Programmeerwedstrijd*[1] (the Flemish Programming competition) by Pieter Wuille (PhD student of DTAI). Pieter helped us greatly by adapting, maintaining and running the benchmarking software for this competition.

The operation mode of the competition system can be sketched as follows. When a team submits a benchmark solution to the competition server, the latter registers it in its database. The competition software maintains identical copies of all submitted solvers and benchmarks on the four competition machines. The tests of the instances are controlled by the database server and four client processes that run on the four test machines. Each client process requests a task from the database server. The task consists of executing a submitted benchmark script on an instance. The database server distributes the tasks and maintains their status and the obtained results. When a client process is assigned a task, it executes it with limited time and memory. When the benchmark script returns a witness, the checker script is called on it. For optimization problems, the checker program is called on the last generated witness. The client process

[1] `http://www.vlaamseprogrammeerwedstrijd.be/?page=main`

collects all necessary data (time, correctness of witness, system error, timeout), sends the data to the database server and requests a new task.

The five competition machines had identical hardware and software. The installed Linux version was Kubuntu Hardy (8.04). The details of the hardware are: *Dell OptiPlex 745 (1 CPU with 2 cores: GenuineIntel Intel(R) Core(TM)2 CPU 6600 2.40GHz), 4096 MB RAM (4x1024 MB 667 MHz 1.5 ns), Disk capacity 160 GB (model ATA ST3160815AS 3.AD)*. Although these machines have two cores, the choice was made to use only one core per task and per computer. Thus, effective parallelism was impossible. Benchmark solutions were executed with a timeout of 600 seconds and a memory limit of 2.79 GB RAM.

4 Competitors

Sixteen teams registered with participants from more than fifteen universities. Each team was assigned a user account on a competition machine. Solvers and benchmark solutions were installed from 1/4/2009 till 15/5/2009 (and later). Most participants did not submit solutions for all benchmarks, often because of limitations of their systems. For instance, only four teams proposed a solution for the Strategic Companies problem (a Σ_2^p-complete problem). Some groups submitted multiple systems: Potsdam joined with two teams each using multiple systems, Helsinki (TKK) with three systems, and a team uniting the forces of researchers at the universities of Kentucky and of Texas at Tyler and at Microsoft also submitted three different systems.

The first ASP systems, *dlv* and (a direct descendant of) *smodels*, were still in the competition and scored very well. The Smodels-IE solver is an updated version of *smodels* developed at the University of Bath. A variety of languages and of solver techniques were present in the competition. As for the languages, twelve teams used different dialects of ASP, three used extensions of first-order logic, and one team used B-Prolog with CLP(FD) and a planning preprocessor.

Five teams (Potassco, DLV, Claspfolio, Smodels-IE, ASPeRiX) participated with "native" ASP solvers, the one of ASPeRiX performing grounding on the fly, without a separate grounder. Other teams used a variety of back-ends: existing or modified SAT solvers (IDP, CMODELS, SUP, Enfragmo, LP2SAT+MINISAT, sabe), SMT solvers (LP2DIFF+BCLT, LP2DIFF+YICES), a PB solver (pbmodels) and a new solver for propositional logic with weight constraints (amsolver). Eight teams used the grounder *gringo* and the benchmark solutions available at the Asparagus system [23].

The teams and their systems are summarized in Table 2. Many teams did not participate on all benchmarks. Table 3 specifies the benchmarks on which teams competed.

5 Results

This section presents the results of the *Second Answer Set Programming System Competition*. For each category of decision problems, we report the score of each team, the number of solved instances, and the total time. For the optimization category, we report the score and total time per participating team. In the rankings, distinction is

Team	Affiliation	Language	Systems
IDP	K.U. Leuven, KRR	FO(·)	*idp* (*gidl* + *minisatid*)
Potassco	U. of Potsdam	ASP	*clasp*, *claspd*, *gringo*, *clingo*, *iclingo*, *clingcon*, *bingo*
DLV	U. of Calabria	ASP	*dlv*
Claspfolio	U. of Potsdam	ASP	*gringo* + *clasp*
Smodels-IE	U. of Bath	ASP	*gringo* + *smodelsie*
ASPeRiX	U. of Angers	ASP	*asperix*
CMODELS	U. of Texas at Austin	ASP	*gringo* + *cmodels*
SUP	U. of Texas at Austin	ASP	*gringo* + *sup*
BPSolver-CLP(FD)	International B-Prolog team	CLP(FD)	*bprolog* (tabling, CLP(FD), B^{fd}_{mv})
Enfragmo	Simon Fraser U., Computational Logic Laboratory	FO(·)	*enfragmo* (grounder + SAT solver)
LP2DIFF+BCLT	Helsinki U. of Technology (TKK)	ASP	*gringo* + *smodels* + *lp2diff* + *bclt*
LP2SAT+MINISAT	Helsinki U. of Technology (TKK)	ASP	*gringo* + *smodels* + *lp2exp* + *minisat*
LP2DIFF+YICES	Helsinki U. of Technology (TKK)	ASP	*gringo* + *smodels* + *lp2diff* + *yices*
pbmodels	U. of Kentucky, U. of Texas at Tyler, Microsoft	ASP	*pbmodels* (uses *minisat+*)
sabe	U. of Kentucky, U. of Texas at Tyler, Microsoft	ASP	*sabe* (uses *minisat*)
amsolver	U. of Kentucky, U. of Texas at Tyler, Microsoft	FO(·)	*amsolver*

Table 2. Participating teams and systems

	Decision in P					Decision in NP																							Σ_2^p	Optimization								
Team	HydraulicPlanning	HydraulicLeaking	CompanyControls	GrammarBasedInformationExtraction	Reachability	BlockedNQueens	Sokoban	15Puzzle	HamiltonianPath	SchurNumbers	TravellingSalesperson	WeightBoundedDominatingSet	Labyrinth	GeneralizedSlitherlink	HierarchicalClustering	ConnectedDominatingSet	GraphPartitioning	Hanoi	Fastfood	WireRouting	Sudoku	DisjunctiveScheduling	KnightTour	ChannelRouting	EdgeMatching	GraphColouring	MazeGeneration	Solitaire	StrategicCompanies	GolombRuler	MaximalClique	15PuzzleOptimize	TravellingSalespersonOptimize	WeightBoundedDomSetOptimize	LabyrinthOptimize	SokobanOptimize	FastfoodOptimize	CompanyControlsOptimize
IDP	y	y	**n**	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	**n**	y	y	y	y	y	y	y	y	**n**
Potassco	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y
DLV	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y
Claspfolio	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	**n**	y	y	y	y	y	y	y	y	y
Smodels-IE	y	y	y	**n**	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	**n**	y	y	y	y	y	y	y	y	y
ASPeRiX	y	y	**n**	y	y	y	y	y	**n**	y	**n**	**n**	**n**	**n**	**n**	**n**	**n**	y	**n**	**n**	y	y	**n**	y	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**
CMODELS	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**
SUP	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**
BPSolver-CLP(FD)	y	y	y	y	y	y	**n**	y	y	y	y	y	**n**	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	**n**	**n**	y	y
Enfragmo	y	y	y	**n**	y	y	**n**	y	y	y	y	y	**n**	y	y	y	y	y	y	**n**	y	y	y	**n**	y	y	**n**	y	**n**	**n**	y	**n**	**n**	**n**	**n**	**n**	**n**	**n**
LP2DIFF+BCLT	y	y	y	**n**	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**
LP2SAT+MINISAT	y	y	y	**n**	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**
LP2DIFF+YICES	y	y	y	**n**	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	y	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**
pbmodels	y	y	**n**	**n**	y	y	y	y	y	y	y	y	**n**	y	y	y	y	y	y	y	y	**n**	y	y	y	y	y	y	**n**	y	**n**	**n**	**n**	**n**	**n**	y	**n**	**n**
sabe	y	y	**n**	**n**	y	y	y	y	y	y	y	y	**n**	y	y	y	y	y	y	y	y	**n**	y	y	y	y	y	y	**n**	y	**n**	y	**n**	**n**	**n**	y	**n**	**n**
amsolver	**n**	**n**	**n**	**n**	**n**	y	**n**	**n**	y	y	y	**n**	**n**	**n**	y	y	**n**	**n**	**n**	**n**	y	**n**	**n**	**n**	**n**	y	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**	**n**

Table 3. Submitted benchmark solutions per team

made between single-system and multi-system teams. The latter used multiple systems/parameter settings for different benchmarks and are marked by *. More statistics and details are available at [24].

Before actually giving the results, we would like to warn the reader to be careful in interpreting them. Below, we point out the most important issues:

- The score is a **weighted sum** of numbers of solved instances per benchmark. Solving an instance of a benchmark with a large number of instances has a smaller contribution than solving one of a benchmark with fewer instances. Thus, it is possible that one team solves more instances but has a smaller score than another team. The weights were introduced to prevent benchmarks with many instances from dominating the competition.
- Most teams did not participate on all benchmarks. The only teams that participated on all benchmarks are Potassco and DLV. When a benchmark solution was missing, a team was assigned score 0 and 600 seconds time per instance. Consequently, the rankings of teams for which benchmark solutions were missing may not give an accurate account of the quality of their systems. We refer to [24] for detailed data per benchmark.
- Fine-tuning a benchmark solution (for instance, by adding certain redundant constraints) may have a major impact on speed. Not all teams were in a position to spend the same amount of time and care on this, which complicates an objective comparison between different solvers. This factor is not significant among the teams that used the encodings available at [23]: Potassco, Claspfolio, CMODELS, SUP, Smodels-IE, LP2DIFF+BCLT, LP2SAT+MINISAT and LP2DIFF+YICES.

5.1 Decision Problems: NP

Benchmarks in this track belong to NP and are not known to be in P. These problems require that solvers search quickly through large search spaces.

The winners in this category are:

FIRST PLACE WINNER	Potassco*	
SECOND PLACE WINNER	Claspfolio	FIRST SINGLE-SYSTEM TEAM
THIRD PLACE WINNER	CMODELS	SECOND SINGLE-SYSTEM TEAM
	IDP	THIRD SINGLE-SYSTEM TEAM

The ranking of all teams is provided in Figure 1. Figure 2 gives a comprehensive graphical overview of the results of all systems. The x-axis represents the number of (solved) benchmark instances, and the y-axis represents the maximum time needed for solving one of these. To compute this plot, the instances solved by each team were ordered according to running times, and a point (x, y) in the chart expresses that the xth instance was solved in y seconds. The more to the right the curve of a team ends, the more benchmark instances were solved within the allocated time and space.

5.2 Decision Problems: P

Problems in this class are polynomially solvable. Difficulty in solving them stems from the sheer size of the instances, which grounders may not be able to handle.

The winners in this category are:

Place	Team	Score	#Solved	Time
1	Potassco*	0.97	491 / 516 = 95%	021253
2	Claspfolio	0.89	451 / 516 = 87%	049513
3	CMODELS	0.85	434 / 516 = 84%	072283
4	IDP	0.83	409 / 516 = 79%	077428
5	LP2SAT+MINISAT	0.82	430 / 516 = 83%	075883
6	SUP	0.80	405 / 516 = 78%	083749
7	DLV	0.76	391 / 516 = 75%	100496
8	LP2DIFF+BCLT	0.73	378 / 516 = 73%	108715
9	LP2DIFF+YICES	0.72	373 / 516 = 72%	096989
10	Smodels-IE	0.61	309 / 516 = 59%	137300
11	Enfragmo	0.59	291 / 516 = 56%	156298
12	BPSolver-CLP(FD)*	0.57	274 / 516 = 53%	155559
13	pbmodels	0.44	214 / 516 = 41%	201563
14	sabe	0.40	203 / 516 = 39%	215250
15	amsolver	0.12	83 / 516 = 16%	265833
16	ASPeRiX	0.12	32 / 516 = 06%	293363

Fig. 1. Decision problems in NP: Ranking

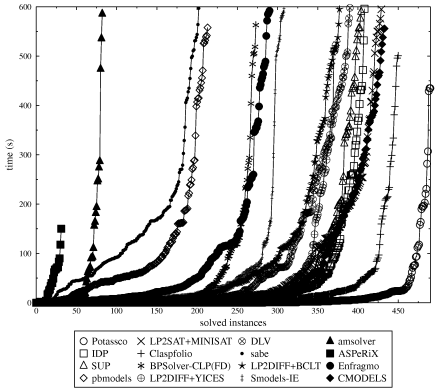

Fig. 2. Decision problems in NP: Plot

FIRST PLACE WINNER	Potassco*	
SECOND PLACE WINNER	BPSolver-CLP(FD)*	
THIRD PLACE WINNER	DLV	FIRST SINGLE-SYSTEM TEAM
	Claspfolio	SECOND SINGLE-SYST. TEAM
	Smodels-IE	THIRD SINGLE-SYSTEM TEAM

The results for all teams are presented in Figure 3 and in Figure 4.

Place	Team	Score	#Solved	Time
1	Potassco*	1.00	89 / 89 = 100%	00735
2	BPSolver-CLP(FD)*	1.00	89 / 89 = 100%	01342
3	DLV	1.00	89 / 89 = 100%	04861
4	Claspfolio	0.80	60 / 89 = 67%	17982
5	Smodels-IE	0.80	60 / 89 = 67%	18021
6	LP2SAT+MINISAT	0.80	60 / 89 = 67%	18270
7	SUP	0.80	60 / 89 = 67%	18606
8	LP2DIFF+BCLT	0.80	60 / 89 = 67%	18713
9	CMODELS	0.80	60 / 89 = 67%	19072
10	LP2DIFF+YICES	0.78	59 / 89 = 66%	18864
11	Enfragmo	0.76	57 / 89 = 64%	24157
12	ASPeRiX	0.69	66 / 89 = 74%	18051
13	IDP	0.54	41 / 89 = 46%	29594
14	sabe	0.41	31 / 89 = 34%	36426
15	pbmodels	0.38	29 / 89 = 32%	36656
16	amsolver	0.00	00 / 89 = 0%	53845

Fig. 3. Decision problems in P: Ranking

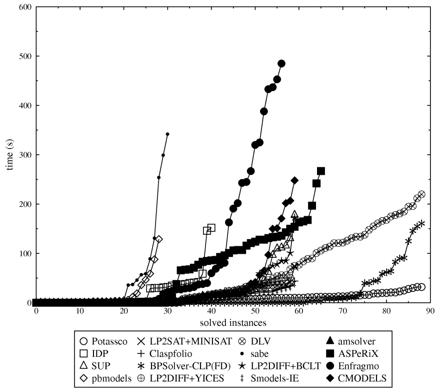

Fig. 4. Decision problems in P: Plot

5.3 Decision Problems: Global

This track consists of all previous decision problems and one Σ_2^p problem, the well-known Strategic Companies problem.

The winners in this category are:

First Place Winner	Potassco*	
Second Place Winner	Claspfolio	First Single-system Team
Third Place Winner	CMODELS	Second Single-system Team
	DLV	Third Single-system Team

The global decision problem results are provided in Figure 5 and in Figure 6.

Place	Team	Score	#Solved	Time
1	Potassco*	0.95	585 / 622 = 94%	029607
2	Claspfolio	0.84	511 / 622 = 82%	077780
3	CMODELS	0.82	498 / 622 = 80%	099721
4	DLV	0.81	497 / 622 = 79%	108448
5	LP2SAT+MINISAT	0.79	490 / 622 = 78%	104438
6	SUP	0.77	465 / 622 = 74%	112641
7	IDP	0.75	450 / 622 = 72%	117223
8	LP2DIFF+BCLT	0.72	438 / 622 = 70%	137713
9	LP2DIFF+YICES	0.70	432 / 622 = 69%	126138
10	BPSolver-CLP(FD)*	0.63	365 / 622 = 58%	165902
11	Smodels-IE	0.62	369 / 622 = 59%	165607
12	Enfragmo	0.60	348 / 622 = 55%	190741
13	pbmodels	0.42	243 / 622 = 39%	248505
14	sabe	0.39	234 / 622 = 37%	261961
15	ASPeRiX	0.21	98 / 622 = 15%	321700
16	amsolver	0.10	83 / 622 = 13%	329963

Fig. 5. Decision problems globally: Ranking

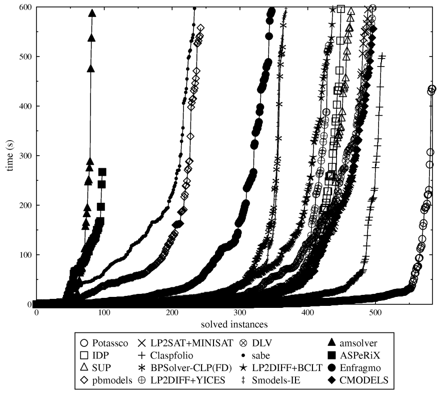

Fig. 6. Decision problems globally: Plot

5.4 Optimization Problems

The competition included nine optimization problems. Most are optimization versions of decision benchmarks, with the exception of Golomb Ruler and Maximal Clique. Only nine of the sixteen teams submitted solutions to optimization problems.

The winners in this category are:

First Place Winner	Potassco*	
Second Place Winner	Claspfolio	First Single-system Team
Third Place Winner	DLV	Second Single-system Team
	IDP	Third Single-system Team

The results for the participating teams are presented in Figure 7 and in Figure 8.

Place	Team	Score	Time
1	Potassco*	81.12	74317
2	Claspfolio	69.61	78333
3	DLV	61.04	92889
4	IDP	50.88	101081
5	Smodels-IE	49.88	103176
6	BPSolver-CLP(FD)*	35.8	113551
7	sabe	6.74	122848
8	Enfragmo	5.07	121598
9	pbmodels	1.19	135883

Fig. 7. Optimization problems: Ranking

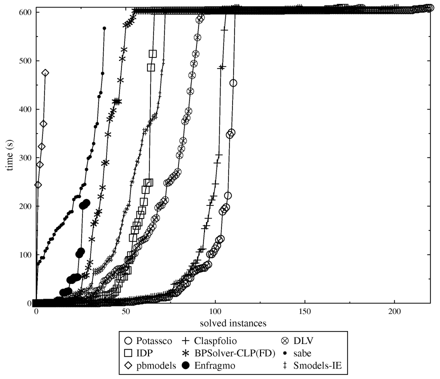

Fig. 8. Optimization problems: Plot

5.5 Decision and Optimization Problems: Global

The goal of this track is to select the systems with widest applicability. Scores are obtained as the average of the scores in the global decision and optimization categories to give decision and optimization problems the same importance.

The winners in this category are:

FIRST PLACE WINNER	Potassco*	
SECOND PLACE WINNER	Claspfolio	FIRST SINGLE-SYSTEM TEAM
THIRD PLACE WINNER	DLV	SECOND SINGLE-SYSTEM TEAM
	IDP	THIRD SINGLE-SYSTEM TEAM

The combined results for decision and optimization problems are shown in Figure 9 and in Figure 10.

5.6 Summary of Results

The Potassco team of the University of Potsdam is the clear winner of the competition. The team won in every category, and in all but the P track with a margin of around 10%. Potassco won twenty of the thirty-eight benchmarks. This is the result of a large effort for developing an excellent library of systems (*clasp*, *claspd*, *gringo*, *clingo*, *iclingo*, *clingcon*, *bingo*) and intensive work of an experienced team on benchmark solutions and parameter tuning. We congratulate the team on this success!

Potassco spent by far the most effort in fine-tuning their systems to each benchmark, and this paid off. Given that the goal of declarative problem solving is to minimize the effort of programmers, it is of equal interest to investigate the performance of teams that used a single collection of systems with uniform parameter settings.

Place	Team	Score	Time
1	Potassco*	0.88	103925
2	Claspfolio	0.77	156113
3	DLV	0.71	201338
4	IDP	0.63	218304
5	Smodels-IE	0.56	268783
6	BPSolver-CLP(FD)*	0.49	279453
7	CMODELS	0.41	237661
8	LP2SAT+MINISAT	0.39	242378
9	SUP	0.38	250581
10	LP2DIFF+BCLT	0.36	275653
11	LP2DIFF+YICES	0.35	264078
12	Enfragmo	0.32	312339
13	sabe	0.23	384810
14	pbmodels	0.21	384388
15	ASPeRiX	0.10	459640
16	amsolver	0.05	467903

Fig. 9. Global category: Ranking

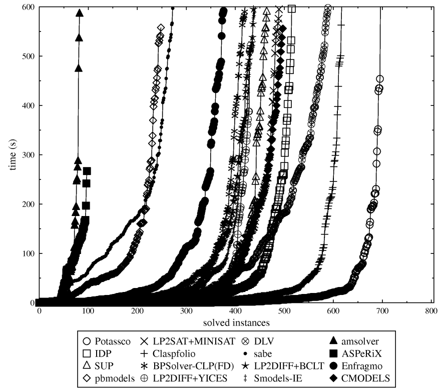

Fig. 10. Global category: Plot

In all rankings except for the P decision track, the best single-system was Claspfolio[2]. Claspfolio ran *gringo* and the *clasp* solver whose settings were chosen from instance features. On decision problems in P, the best team using a single setting for all benchmarks is DLV. BPSolver-CLP(FD) also performed excellently on P problems.

As for the winners of individual benchmarks, we already mentioned that Potassco won twenty. Five benchmarks were won by BPSolver-CLP(FD), and four by DLV. Claspfolio and Smodels-IE each won two, and IDP, Enfragmo and amsolver one each[3].

6 Discussion

The principal goals of this competition, namely, taking a snapshot of the state of the art of declarative programming paradigms and fostering future improvements, are similar to related competitions on SAT, PB, QBF, SMT, CP, etc. However, in contrast to those and the first ASP competition held in 2007, the form of this competition, aiming at openness towards alternative paradigms, was quite different. This manifests itself in the fact that participants were allowed (and actually required) to provide their individual modelings for the benchmarks used in the competition. In contrast to the other competitions mentioned, the inputs to systems run in this competition were not fixed by the organizers, except for the (arbitrarily chosen) format of problem instances. As a consequence, the results of this competition may indicate trends on the simplicity or difficulty of developing effective problem solutions using particular systems, but they cannot provide a perfect picture of the efficiency of the systems themselves.

Several SAT, PB and SMT systems were involved in the competition, as back-ends of ASP or FO($\cdot$) systems. Techniques from these areas are also applied in "native" ASP

[2] In fact, Claspfolio, just like CMODELS and SUP, used a different grounder for one benchmark. We can ignore this here because they had a zero score for this benchmark.

[3] This list does not include the multiple ex aequo winners of HydraulicLeaking and HydraulicPlanning.

solvers like those used by Potassco and Claspfolio. That there were no teams from these areas in the competition, and only one team from CLP, may have different explanations. These fields have their own, well-established competitions, while the ASP competition is relatively recent and open to them only for the first time. Another explanation is that the difficulty of modeling the benchmark problems was very high for them. We know of one SAT team that considered to participate in the competition, but gave up because of this reason. Despite the flexibility of CLP(FD) in modeling constraint problems, BPSolver-CLP(FD) had difficulties in modeling certain benchmarks and did not submit solutions for all of the planning problems. ASP and FO($\cdot$) appear to offer superior modeling facilities, which is hardly surprising given that these languages were developed for knowledge representation. On the other hand, BPSolver-CLP(FD) came in second in the P track and won on three benchmarks of the NP track. This shows that tabling as well as the constraint programming techniques featured in B-Prolog can be very useful for some kinds of problems, in particular, those in which large domains would make exhaustive grounding blow up in space. Other new entrants in the competition were the FO-based systems IDP, Enfragmo and amsolver. IDP ended fourth in the NP and global track but was less successful for P decision problems. Enfragmo and amsolver performed very well on certain benchmarks but did not compete in enough benchmarks to obtain a good ranking.

In their invited talks at LPNMR 2007, both Nicola Leone[4] and Jack Minker[5] appealed for using real-world application problems in the ASP competition. Although the benchmarks of the current contest covered a variety of different modeling or computational aspects, only a few benchmarks came from such applications. This issue was discussed with several contributors of benchmarks. The problem is not that there are no real-world applications. In fact, the contrary is shown in the application summary track of LPNMR 2009. But such real-world problems tend to be very complex, making it harder for a contributor to describe the problem in an informal yet unambiguous way and to create suitable problem instances. Moreover, the effort of modeling such problems may become too high for some contestants. This problem is inherent to a Model and Solve competition and does not occur in competitions where the formal theory is given, as in the SAT competition or the categories SCore, SCore-v and SLparse of the first ASP competition.

As motivated in Section 2, teams were free to fine-tune their solving systems towards particular benchmarks. To this end, they could use a number of instances that were available during the installation phase. Two teams effectively fine-tuned their systems in this way. BPSolver-CLP(FD) may have no option than to do so, since the programmer needs to specify the search strategy in B-Prolog. Potassco took the opportunity to test its library of systems and system parameters for controling preprocessing, heuristics and restarts[6]. In the previous section, we therefore distinguished these teams from the other single-system teams. Given that Claspfolio and Potassco used the same benchmark solutions and mostly the same technology, the competition gives a fairly accurate account of the impact of Potassco's effort on fine-tuning. Globally, Claspfolio

[4] `http://lpnmr2007.googlepages.com/nicola-lpnmr07.pdf`

[5] `http://lpnmr2007.googlepages.com/LPNMR-07.ppt`

[6] For details, see Potassco's team webpage [24].

lost 10% on Potassco and it was outperformed by Potassco in a few benchmarks. On the one hand, this shows that in the current state of the art, fine-tuning pays off and may be imperative to build hard real-world applications. On the other hand, the long term goal of declarative problem solving is to allow a programmer to focus on the declarative properties of the problem and to relieve him or her of tedious control issues. The example of Potassco and Claspfolio allows us to evaluate our current progress towards this goal. In this respect, it is encouraging to see that globally, Claspfolio lost by *only* 10%.

We would like to end with some reflections and recommendations on the competition format. We believe that an open model and solve competition like this one fosters cross-fertilization between different areas of declarative programming and gives valuable global information on the quality of modeling and solving technologies. On the other hand, it does not allow for a precise and unbiased comparison of system performance, due to the use of different encodings. To allow for more detailed and objective comparisons, separate competitions are needed in which problem encodings are given and fixed. This is the format used in the SAT competition and also in the SCore, SCore-v and SLparse tracks of the previous ASP competition. In future competitions, it would be of interest to have such tracks for both grounders and solvers. Such tracks will not be accessible for areas that do not rely on grounding and solving. A further prerequisite is the availability of common input languages. As regards a grounder competition, a common high-level fragment of ASP and FO($\cdot$) is currently lacking and should be strived for. For an unbiased comparison of solvers, some low-level language similar to DIMACS for SAT, Lparse for ASP or MNF for FO($\cdot$) would need to be selected.

A final recommendation concerns the participation of SAT teams or systems in the competition. SAT technology is heavily used in ASP and FO($\cdot$). The model and solve track would be a good occasion to compare different SAT systems in the context of typical ASP benchmarks. Participation of a SAT team can be made very easy by providing a grounder to DIMACS, benchmark solutions in the language of the grounder and a script translating SAT output into the competition format. The grounders of Enfragmo and LP2SAT+MINISAT could be used for this purpose.

Acknowledgments

A great number of people have contributed to the success of this competition. We thank the members of the program committee who helped to set up the competition format. We are indebted to all members of the ASP and CLP community that submitted benchmarks. We are extremely grateful to everyone that participated in the contest. Thank you for your enthusiasm and your encouragements.

We wish to thank the DTAI group of the K.U. Leuven for providing the pool of computers on which the competition was run. Special thanks go to Pieter Wuille for developing and managing the competition software, and to Bart Swennen and Kris Vangeneugden for setting up the computer infrastructure. We are grateful also to Hanne Vlaeminck and Maarten Mariën, who together with the first three authors formed the local organization team. The KRR group participated in the competition with the IDP

team, mostly through the efforts of Johan Wittocx who did 90% of the work. The group thanks him for this.

On behalf of the program committee and the participants, Martin Gebser and Mirosław Truszczyński would like to thank the KRR group of the K.U. Leuven for the huge efforts taken to prepare and to conduct this competition. Their excellent work was invaluable to make this competition, first, possible and, second, a great success for the ASP community in the most general sense, in view of presenting and representing a broad spectrum of alternative yet tightly bonded modeling and solving approaches.

Martin Gebser acknowledges partial funding by DFG under Grant SCHA 550/8-1.

References

1. Marek, V., Truszczyński, M.: Stable models and an alternative logic programming paradigm. In Apt, K., Marek, W., Truszczyński, M., Warren, D., eds.: The Logic Programming Paradigm: a 25-Year Perspective, Springer (1999) 375–398
2. Niemelä, I.: Logic programming with stable model semantics as a constraint programming paradigm. Annals of Mathematics and Artificial Intelligence **25**(3-4) (1999) 241–273
3. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
4. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9**(3-4) (1991) 365–385
5. Colmerauer, A., Kanoui, H., Pasero, R., Roussel, P.: Un systeme de communication homme-machine en Francais. Technical report, University of Marseille (1973)
6. Kowalski, R.: Predicate logic as a programming language. In Rosenfeld, J., ed.: Proceedings of the Congress of the International Federation for Information Processing, North Holland (1974) 569–574
7. McCarthy, J.: Circumscription — a form of nonmonotonic reasoning. Artificial Intelligence **13**(1-2) (1980) 27–39
8. Reiter, R.: A logic for default reasoning. Artificial Intelligence **13**(1-2) (1980) 81–132
9. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In Kowalski, R., Bowen, K., eds.: Proceedings of the International Conference on Logic Programming, MIT Press (1988) 1070–1080
10. Eiter, T., Leone, N., Mateis, C., Pfeifer, G., Scarcello, F.: A deductive system for nonmonotonic reasoning. In Dix, J., Furbach, U., Nerode, A., eds.: Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning, Springer (1997) 364–375
11. Niemelä, I., Simons, P.: Smodels — an implementation of the stable model and well-founded semantics for normal logic programs. In Dix, J., Furbach, U., Nerode, A., eds.: Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning, Springer (1997) 420–429
12. Lifschitz, V.: Answer Set Planning. In De Schreye, D., ed.: Proceedings of the International Conference on Logic Programming, MIT Press (1999) 23–37
13. Borchert, P., Anger, C., Schaub, T., Truszczyński, M.: Towards systematic benchmarking in answer set programming: The Dagstuhl initiative. In Lifschitz, V., Niemelä, I., eds.: Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning, Springer (2004) 3–7
14. 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.: Proceedings of the International Conference on Logic Programming and Nonmonotonic Reasoning, Springer (2007) 3–17

15. Biere, A., Heule, M., van Maaren, H., Walsh, T., eds.: Handbook of Satisfiability. IOS Press (2009)
16. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: From an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). Journal of the ACM **53**(6) (2006) 937–977
17. Rossi, F., van Beek, P., Walsh, T., eds.: Handbook of Constraint Programming. Elsevier (2006)
18. Denecker, M., Kakas, A.: Abduction in Logic Programming. In Kakas, A., Sadri, F., eds.: Computational Logic: Logic Programming and Beyond, Springer (2002) 402–436
19. Mitchell, D., Ternovska, E.: A framework for representing and solving NP search problems. In Veloso, M., Kambhampati, S., eds.: Proceedings of the National Conference on Artificial Intelligence, AAAI Press / MIT Press (2005) 430–435
20. Kakas, A., Michael, A: Air-Crew scheduling through abduction. In Imam, I., Kodratoff, Y., El-Dessouki, A., Ali, M., eds.: Proceedings of the International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Springer (1999) 600–611
21. Pelov, N., De Mot, E., Denecker, M.: Logic programming approaches for representing and solving constraint satisfaction problems: A comparison. In Parigot, M., Voronkov, A., eds.: Proceedings of the International Conference on Logic for Programming and Automated Reasoning, Springer (2000) 225–239
22. Van Hentenryck, P.: Constraint Satisfaction in Logic Programming. MIT Press (1989)
23. `http://asparagus.cs.uni-potsdam.de`
24. `http://www.cs.kuleuven.be/~dtai/events/ASP-competition`