The Second Answer Set Programming Competition

Marc Denecker Joost Vennekens Stephen Bond
Martin Gebser Mirosław Truszczyński

Katholieke Universiteit Leuven University of Potsdam University of Kentucky

17 Sept 2009

The Second Answer Set Programming Competition

Marc Denecker Joost Vennekens Stephen Bond
Martin Gebser Mirosław Truszczyński

Katholieke Universiteit Leuven University of Potsdam University of Kentucky

17 Sept 2009

University of Angers
CUNY Brooklyn College
Helsinki University of Technology
Katholieke Universiteit Leuven
University of Potsdam
University of Texas at Austin

University of Bath
University of Calabria
University of Kentucky
Microsoft Corporation
Simon Fraser University
University of Texas at Tyler

Outline

Answer Set Programming

Answer set programming is an emerging programming/problem solving paradigm. The fundamental underlying idea is to describe a problem declaratively in such a way that models of the description provide solutions to problems. One particular instance of this paradigm are logic programs under stable model semantics (respectively answer set semantics if an extended class of logic programs is used).

(G. Brewka, I. Niemelä, T. Schaub, M. Truszczyński; 2002)

Answer Set Programming

Answer set programming is an emerging programming/problem solving paradigm. The fundamental underlying idea is to describe a problem declaratively in such a way that models of the description provide solutions to problems. One particular instance of this paradigm are logic programs under stable model semantics (respectively answer set semantics if an extended class of logic programs is used).

(G. Brewka, I. Niemelä, T. Schaub, M. Truszczyński; 2002)

What has ASP to offer?

Constraint Solving Paradigms

- In many constraint problems, we search for complex objects that satisfy certain properties
  - schedules, assignments, plans, diagnoses, etc.

Constraint Solving Paradigms

- In many constraint problems, we search for complex objects that satisfy certain properties
  - schedules, assignments, plans, diagnoses, etc.
- Different constraint programming paradigms, different ways of representing these complex objects

Constraint Solving Paradigms

- In many constraint problems, we search for complex objects that satisfy certain properties
  - schedules, assignments, plans, diagnoses, etc.
- Different constraint programming paradigms, different ways of representing these complex objects
  - In CLP, SAT: by sets of constraint variables:
    - Programs generate constraint variables, store them in datastructures (lists, trees, . . .) and generate the constraints over them.

Constraint Solving Paradigms

- In many constraint problems, we search for complex objects that satisfy certain properties
  - schedules, assignments, plans, diagnoses, etc.

- Different constraint programming paradigms, different ways of representing these complex objects
  - In CLP, SAT: by sets of constraint variables:
    - Programs generate constraint variables, store them in datastructures (lists, trees, . . .) and generate the constraints over them.
  - In the ASP-computational paradigm : by a structure, an answer set
    - Properties of structures expressed by logical formulas

Constraint Solving Paradigms

- In many constraint problems, we search for complex objects that satisfy certain properties
  - schedules, assignments, plans, diagnoses, etc.
- Different constraint programming paradigms, different ways of representing these complex objects
  - In CLP, SAT: by sets of constraint variables:
    - Programs generate constraint variables, store them in datastructures (lists, trees, . . .) and generate the constraints over them.
  - In the ASP-computational paradigm : by a structure, an answer set
    - Properties of structures expressed by logical formulas
  - This idea was pioneered using Answer Set Programming formalisms (smodels, dlv) but is also possible for other KR-languages

Using KR-logics for Answer Set Programming

- Knowledge representation logics are designed for representing knowledge about the world
  - The world is a very complex object
  - Formally represented as a structure

Using KR-logics for Answer Set Programming

- Knowledge representation logics are designed for representing knowledge about the world
  - The world is a very complex object
  - Formally represented as a structure
- KR-languages offer a clear modeling advantage compared to CP, CLP, CSP, SAT, but there is a implementation disadvantage
- ASP tries to close the gap with SAT, CLP

Using KR-logics for Answer Set Programming

- Knowledge representation logics are designed for representing knowledge about the world
  - The world is a very complex object
  - Formally represented as a structure
- KR-languages offer a clear modeling advantage compared to CP, CLP, CSP, SAT, but there is a implementation disadvantage
- ASP tries to close the gap with SAT, CLP

$\Rightarrow$ ASP-Programming Competition !!!

(First) Competitions

1994	Prolog Programming Competition
1996	CADE ATP System Competition
2002	SAT Competition/Race
2005	CSP Solver Competition
	PB Evaluation
	SMT Competition
2006	QBF Solver Evaluation
2007	ASP System Competition

(First) Competitions

1994	Prolog Programming Competition
1996	CADE ATP System Competition
2002	SAT Competition/Race
2005	CSP Solver Competition PB Evaluation SMT Competition
2006	QBF Solver Evaluation
2007	ASP System Competition

Many communities established competitions to evaluate modeling skills, systems and tools!

A Success Story
from SAT 2009

Application/Industrial category of SAT 2007 competition

- Won by the Rsat solver

A Success Story
from SAT 2009

Application/Industrial category of SAT 2007 competition

- Won by the Rsat solver
  - Congratulations to the champion!

Application/Industrial category of SAT 2009 competition

- Rsat 2007 version (entered for comparison) came in . . .

A Success Story
from SAT 2009

Application/Industrial category of SAT 2007 competition

- Won by the Rsat solver
  - Congratulations to the champion!

Application/Industrial category of SAT 2009 competition

- Rsat 2007 version (entered for comparison) came in . . . 13th
  - Congratulations to 12 new solvers beating the former champion!

Outline

The 2nd ASP-competition

- Organized by the KRR-group of the K.U.Leuven, Belgium
  - Modelgeneration using FO($\cdot$)
  - Inside the ASP-paradigm, outside ASP-language
- Invitation by steering committee of ASP-competition

The 2nd ASP-competition

- Organized by the KRR-group of the K.U.Leuven, Belgium
  - Modelgeneration using FO($\cdot$)
  - Inside the ASP-paradigm, outside ASP-language
- Invitation by steering committee of ASP-competition
  - Several invitations actually :-)

The 2nd ASP-competition

- Organized by the KRR-group of the K.U.Leuven, Belgium
  - Modelgeneration using FO($\cdot$)
  - Inside the ASP-paradigm, outside ASP-language
- Invitation by steering committee of ASP-competition
  - Several invitations actually :-)
- We accepted on some conditions — ($\neq$ 1st ASP-competition)
  - Model and Solve competition only
  - Open to all constraint programming paradigms
  - Decision problems and optimisation problems

Opening up ASP: a trend

- Answer Set Programming and Other Computing Paradigms (ASPOCP) (2008,2009)
- Logic and Search (LaSh) (2006,2008)

  To bring together researchers from all fields that share the problem solving methodology based on model generation

Chronology of the competition

- Collection of Benchmarks (December 2008 - March 2009)
- Participants registered and installed solutions (April - May 2009)
- Competition was run (June)

Outline

Benchmarks: discussions developed

- Industrial size benchmarks
  - Nicola Leone and Jack Minker at LPNMR07 had made an urgent appeal to submit real application problems for this ASP competition
  - This was in the call for benchmarks
  - Not much response
  - To be discussed . . .

Benchmarks: discussions developed

- Industrial size benchmarks
  - Nicola Leone and Jack Minker at LPNMR07 had made an urgent appeal to submit real application problems for this ASP competition
  - This was in the call for benchmarks
  - Not much response
  - To be discussed ...
- Objections against P and Σ_2^P benchmarks
  - Some feel that the competition should focus on NP problems
  - Many smaller teams do not have systems for handling large P problems and do not have the expressivity for Σ_2^P problems.
  - To be discussed ...

The compromise

- Philosophy of this competition:
  - Getting as much information out of the competition as possible

The compromise

- Philosophy of this competition:
  - Getting as much information out of the competition as possible
- Measuring different kind of qualities:
  - NP-problems : inherent speed of the solver
  - P, NP, Σ_2^p-problems : broad applicability

The compromise

- Philosophy of this competition:
  - Getting as much information out of the competition as possible
- Measuring different kind of qualities:
  - NP-problems : inherent speed of the solver
  - P, NP, Σ_2^p-problems : broad applicability
- We opted for:
  - Allowing all types of benchmarks
  - Splitting up in different categories

Categories

Global

Decision

- Problems in P (5)
- Problems in NP (23) (not known to be in P)
- Σ_2^P-complete (Strategic Companies)

Optimization (9)

Decision Problems

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

Optimization Problems

Benchmark	Contributors	#Instances
GolombRuler	Martin Brain	24
MaximalClique	Johan Wittocx	29
15PuzzleOptimize	L. Liu, M. Truszczyński and M. Gebser	16
TravellingSalespersonOptimize	L. Liu, M. Truszczyński and M. Gebser	29
WeightBoundedDominatingSetOptimize	L. Liu, M. Truszczyński and M. Gebser	29
LabyrinthOptimize	Martin Gebser	28
SokobanOptimize	Wolfgang Faber	29
FastfoodOptimize	Wolfgang Faber	29
CompanyControlsOptimize	Mario Alviano	15

Many thanks

to all contributors!

Outline

Sixteen Teams

Team	Affiliation	Lang.	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*

Participants

16 teams - 9 new ones

Participants

16 teams - 9 new ones

Modeling Languages

- ASP (dialects): 12 teams
- FO(·): 3 teams
- CLP(FD): 1 team

Participants

16 teams - 9 new ones

Modeling Languages

- ASP (dialects): 12 teams
- FO(·): 3 teams
- CLP(FD): 1 team

Solving Systems

- "native" ASP solvers: 5 teams (*asperix* grounding on-the-fly)
- SAT solvers: 6 teams
- SMT solvers: 2 teams
- PB solvers: 1 team
- *bprolog*: 1 team

Discussions developed

- In an open competition, rules should be literal:
  - Allowing a SAT team to write a separate C++ program turning instances into CNF
  - Allowing a CLP-solver to specify labeling strategy

Discussions developed

- In an open competition, rules should be literal:
  - Allowing a SAT team to write a separate C++ program turning instances into CNF
  - Allowing a CLP-solver to specify labeling strategy
- But such rules also
  - allow fine-tuning of a parameters of an ASP-system
  - allow different ASP systems in different benchmarks
  - allow C++ solutions (originally)

Discussions developed

- In an open competition, rules should be literal:
  - Allowing a SAT team to write a separate C++ program turning instances into CNF
  - Allowing a CLP-solver to specify labeling strategy
- But such rules also
  - allow fine-tuning of a parameters of an ASP-system
  - allow different ASP systems in different benchmarks
  - allow C++ solutions (originally)
- When it became clear that only two teams used this liberty this turned into a heated debate :-)
  - Potassco used different grounders, solvers and runtime parameters
  - BPSolver-CLP(FD) varied labeling strategies
- Decision to split the competition in two: (a Salomons judgement!?)
  - Single-system teams
  - Multi-system teams (Marked with a *)

Discussions developed

- In an open competition, rules should be literal:
  - Allowing a SAT team to write a separate C++ program turning instances into CNF
  - Allowing a CLP-solver to specify labeling strategy
- But such rules also
  - allow fine-tuning of a parameters of an ASP-system
  - allow different ASP systems in different benchmarks
  - allow C++ solutions (originally)
- When it became clear that only two teams used this liberty this turned into a heated debate :-)
  - Potassco used different grounders, solvers and runtime parameters
  - BPSolver-CLP(FD) varied labeling strategies
- Decision to split the competition in two: (a Salomons judgement!?)
  - Single-system teams
  - Multi-system teams (Marked with a *)

Many thanks

to all competitors!

Input and Output

Input Instances

- Atomic clauses over input predicates (facts)

Output Decision Problems

- **UNSATISFIABLE**,
- Atomic clauses over output predicates (witness) or
- **UNKNOWN**

Output Optimization Problems

- **UNSATISFIABLE** or
- Sequence of witnesses and, possibly, **OPTIMUM FOUND**

Scoring

Decision Problems

- Calculate a *score* per *team* over all benchmark problems P
- $score_{team} = \sum_{\text{Problem } P} \frac{S^P_{team}}{S^P_{some}}$
  - S^P_{team}: Number of instances of P solved by *team*
  - $S^P_{team} := 0$ if solution of *team* gave some wrong answer on P
  - S^P_{some}: Number of instances of P solved by *some* team

Scoring

Decision Problems

- Calculate a *score* per *team* over all benchmark problems P
- $score_{team} = \sum_{\text{Problem } P} \frac{S^P_{team}}{S^P_{some}}$
  - S^P_{team}: Number of instances of P solved by *team*
  - $S^P_{team} := 0$ if solution of *team* gave some wrong answer on P
  - S^P_{some}: Number of instances of P solved by *some* team

  ➥ Same weight for each problem P

- The higher $score_{team}$ the better
- Runtime used as tie-breaker

Scoring

Optimization Problems

- Calculate a quality Q^B_{team} for each benchmark B and each *team*
  - $Q^B_{team} := 1$ if answer **UNSATISFIABLE** given, otherwise
  - $Q^B_{team} := 0.25$ if some witness found
  - $Q^B_{team} := Q^B_{team} + 0.5\frac{M^B_{all}}{M^B_{team}}$ where
    - M^B_{team}: Objective function value for last witness found by *team*
    - M^B_{all}: Objective function value for best last witness of *all* teams
  - $Q^B_{team} := Q^B_{team} + 0.25$ if answer **OPTIMUM FOUND** given

Scoring

Optimization Problems

- Calculate a quality Q^B_{team} for each benchmark B and each *team*
  - $Q^B_{team} := 1$ if answer **UNSATISFIABLE** given, otherwise
  - $Q^B_{team} := 0.25$ if some witness found
  - $Q^B_{team} := Q^B_{team} + 0.5\frac{M^B_{all}}{M^B_{team}}$ where
    - M^B_{team}: Objective function value for last witness found by *team*
    - M^B_{all}: Objective function value for best last witness of *all* teams
  - $Q^B_{team} := Q^B_{team} + 0.25$ if answer **OPTIMUM FOUND** given
- $score_{team} = \sum_{\text{Problem } P} \frac{\sum_{\text{Instance } B \text{ of } P} Q^B_{team}}{S^P_{some}}$
  - $Q^B_{team} := 0$ if solution of *team* gave some wrong answer on P
  - S^P_{some}: Number of instances of P solved by *some* team

Scoring

Optimization Problems

- Calculate a quality Q^B_{team} for each benchmark B and each *team*
  - $Q^B_{team} := 1$ if answer **UNSATISFIABLE** given, otherwise
  - $Q^B_{team} := 0.25$ if some witness found
  - $Q^B_{team} := Q^B_{team} + 0.5 \frac{M^B_{all}}{M^B_{team}}$ where
    - M^B_{team}: Objective function value for last witness found by *team*
    - M^B_{all}: Objective function value for best last witness of *all* teams
  - $Q^B_{team} := Q^B_{team} + 0.25$ if answer **OPTIMUM FOUND** given
- $score_{team} = \sum_{\text{Problem } P} \frac{\sum_{\text{Instance } B \text{ of } P} Q^B_{team}}{S^P_{some}}$
  - $Q^B_{team} := 0$ if solution of *team* gave some wrong answer on P
  - S^P_{some}: Number of instances of P solved by *some* team
- Rest of scoring similar to decision problems

Platform

- Cluster of five identical Linux machines
  - One machine accessible to participants
  - Four machines running benchmarks
  - Identical copies of submitted solutions and benchmarks on all machines
- Resources per run
  - 600 seconds
  - 2.79 GB RAM
  - One core (no effective parallelism)

Platform

- Cluster of five identical Linux machines
  - One machine accessible to participants
  - Four machines running benchmarks
  - Identical copies of submitted solutions and benchmarks on all machines
- Resources per run
  - 600 seconds
  - 2.79 GB RAM
  - One core (no effective parallelism)
- Teams submitted solutions per benchmark problem
  - Installation phase to test solutions on sample benchmarks
- Checker scripts did only polynomial tasks (verifying correctness of solutions)
  - **UNSATISFIABLE** and **OPTIMUM FOUND** checked by comparison with other answers

Outline

Decision: P

favors grounding/query-answering

Team	Score	#Solved	Time
Potassco*	1.00	089 / 089 = 100%	000735
BPSolver-CLP(FD)*	1.00	089 / 089 = 100%	001342
DLV	1.00	089 / 089 = 100%	004861
Claspfolio	0.80	060 / 089 = 67%	017982
Smodels-IE	0.80	060 / 089 = 67%	018021
LP2SAT+MINISAT	0.80	060 / 089 = 67%	018270
SUP	0.80	060 / 089 = 67%	018606
LP2DIFF+BCLT	0.80	060 / 089 = 67%	018713
CMODELS	0.80	060 / 089 = 67%	019072
LP2DIFF+YICES	0.78	059 / 089 = 66%	018864
Enfragmo	0.76	057 / 089 = 64%	024157
ASPeRiX	0.69	066 / 089 = 74%	018051
IDP	0.54	041 / 089 = 46%	029594
sabe	0.41	031 / 089 = 34%	036426
pbmodels	0.38	029 / 089 = 32%	036656
amsolver	0.00	000 / 089 = 0%	053845

Decision: NP

favors search

Team	Score	#Solved	Time
Potassco*	0.97	491 / 516 = 95%	021253
Claspfolio	0.89	451 / 516 = 87%	049513
CMODELS	0.85	434 / 516 = 84%	072283
IDP	0.83	409 / 516 = 79%	077428
LP2SAT+MINISAT	0.82	430 / 516 = 83%	075883
SUP	0.80	405 / 516 = 78%	083749
DLV	0.76	391 / 516 = 75%	100496
LP2DIFF+BCLT	0.73	378 / 516 = 73%	108715
LP2DIFF+YICES	0.72	373 / 516 = 72%	096989
Smodels-IE	0.61	309 / 516 = 59%	137300
Enfragmo	0.59	291 / 516 = 56%	156298
BPSolver-CLP(FD)*	0.57	274 / 516 = 53%	155559
pbmodels	0.44	214 / 516 = 41%	201563
sabe	0.40	203 / 516 = 39%	215250
amsolver	0.12	083 / 516 = 16%	265833
ASPeRiX	0.12	032 / 516 = 06%	293363

Decision: Global

favors grounding/query-answering and search

Team	Score	#Solved	Time
Potassco*	0.95	585 / 622 = 94%	029607
Claspfolio	0.84	511 / 622 = 82%	077780
CMODELS	0.82	498 / 622 = 80%	099721
DLV	0.81	497 / 622 = 79%	108448
LP2SAT+MINISAT	0.79	490 / 622 = 78%	104438
SUP	0.77	465 / 622 = 74%	112641
IDP	0.75	450 / 622 = 72%	117223
LP2DIFF+BCLT	0.72	438 / 622 = 70%	137713
LP2DIFF+YICES	0.70	432 / 622 = 69%	126138
BPSolver-CLP(FD)*	0.63	365 / 622 = 58%	165902
Smodels-IE	0.62	369 / 622 = 59%	165607
Enfragmo	0.60	348 / 622 = 55%	190741
pbmodels	0.42	243 / 622 = 39%	248505
sabe	0.39	234 / 622 = 37%	261961
ASPeRiX	0.21	098 / 622 = 15%	321700
amsolver	0.10	083 / 622 = 13%	329963

Optimization

favors search

Team	Score	Time
Potassco*	0.81	074317
Claspfolio	0.69	078333
DLV	0.61	092889
IDP	0.50	101081
Smodels-IE	0.49	103176
BPSolver-CLP(FD)*	0.35	113551
sabe	0.06	122848
Enfragmo	0.05	121598
pbmodels	0.01	135883

Global

favors broad applicability

Team	Score	Time
Potassco*	0.88	103925
Claspfolio	0.77	156113
DLV	0.71	201338
IDP	0.63	218304
Smodels-IE	0.56	268783
BPSolver-CLP(FD)*	0.49	279453
CMODELS	0.41	237661
LP2SAT+MINISAT	0.39	242378
SUP	0.38	250581
LP2DIFF+BCLT	0.36	275653
LP2DIFF+YICES	0.35	264078
Enfragmo	0.32	312339
sabe	0.23	384810
pbmodels	0.21	384388
ASPeRiX	0.10	459640
amsolver	0.05	467903

And the winner is . . .

Congratulations to the developers of Potsdam!

Outline

Warning: Interpreting the results

- An Model and Solve competition yields only vague information about system efficiency
- The modeling plays a hugh role!
  - Many teams did not submit solutions for all benchmarks.
  - Not all groups spent the same amount of time and care in the modeling
  - (However, this does not play for a group of 8 teams)

- The large groups put great effort in modeling:
  - Potsdam (Potassco)
  - Calabria (dlv)

- Potsdam published its modelings at Asparagus
  - Eight teams used gringo and could use the Potsdam solutions!!
  - Potassco, Claspfolio, CMODELS, SUP, Smodels-IE, LP2DIFF+BCLT, LP2SAT+MINISAT and LP2DIFF+YICES.

Thanks to

the Potsdam group

for making their solutions available, and for the many other ways that they have supported the organisation of the competition!!

For the future

- The only way to avoid the impact of modeling is a track where the theory is given
- Requires a common language
  - Propositional level?
  - Predicate level

Discussion: CLP-competitor

- Neng-Fa Zhou submitted quite a few typical CLP-benchmarks that should be challenging for the ASP-solvers
- BPSolver-CLP(FD) won five benchmarks
- On some benchmarks it was superior
  - Disjunctive scheduling 60× faster than Potassco
  - On some of its other benchmarks, it lost (sudoku)

- ASP-solvers resisted quite well to the challenge
- To be discussed (impact of instances?)

Gain of fine-tuning

Potascco versus Claspfolio

- A difference of 10% more solved instances
- Larger difference if time is taken into account

A hope giving result for the development of general uniform ASP solvers based. To be followed up.

Industrial benchmarks

- Impossible in a model and solve competition
  - Too complex for modeling, too many ambiguities
- Only possible in a track with a given theory.
- Limitation on complexity of benchmarks?

An open competition

- Invitations, especially to CP and SAT
- Poor response
  - Three teams performing modelgeneration for FO($\cdot$)
    - Enfragmo (SFU)
    - Amsolver (U.Kentucky)
    - IDP (K.U.Leuven)
  - Several CP-teams considered participation, only one participated
  - At least one SAT team considered but gave up.
  - No Abductive logic programming team
- Reason: at least partially due to the difficulty of the modeling (personal communication)!

Participation of SAT

- But several SAT and SMT solvers were used in ASP systems
- With a little bit more support, SAT teams could easily participate

To be discussed ...

Outline

Conclusions

- This year we had an open modeling and solving competition
  - Out of 16 teams, 9 competed for the first time!
  - Three teams used FO(·) and one CLP(FD) as (alternative) languages
  - SAT and SMT solvers used, but no team modeled in their languages

  ↪ KR-languages appear to be particularly well-suited for modeling (the contrary would have been most upsetting!)

1

Conclusions

- This year we had an open modeling and solving competition
  - Out of 16 teams, 9 competed for the first time!
  - Three teams used FO(·) and one CLP(FD) as (alternative) languages
  - SAT and SMT solvers used, but no team modeled in their languages

  ↪ KR-languages appear to be particularly well-suited for modeling (the contrary would have been most upsetting!)
- Teams used essentially different
  - Problem modelings and
  - Solving systems

  ↪ Results indicate trends on the ease of developing effective solutions, but not more

1

Conclusions

- This year we had an open modeling and solving competition
  - Out of 16 teams, 9 competed for the first time!
  - Three teams used FO(·) and one CLP(FD) as (alternative) languages
  - SAT and SMT solvers used, but no team modeled in their languages

  ➥ KR-languages appear to be particularly well-suited for modeling (the contrary would have been most upsetting!)
- Teams used essentially different
  - Problem modelings and
  - Solving systems

  ➥ Results indicate trends on the ease of developing effective solutions, but not more
- Much more efforts on developing benchmarks/solutions[1] than in 2007
  - Many new benchmark problems, in particular, for optimization
  - Teams developed individual solutions for their solving systems

  ➥ Problems and solutions provide showcase for declarative programming, but real application benchmarks were still missing

[1] Find out more at: `www.cs.kuleuven.be/~dtai/ASP-competition`

My personal thanks to

- **Joost Vennekens, Stephen Bond, Pieter Wuille for organisation stuff!**
- **Johan Wittocx for running in the competition!**
  - Two KRR people left — unexpectedly
  - Others had to take over

And last but not least

Thank you all so much for your support and patience!!

Questions

- Industrial benchmarks?
- Complexity of benchmarks?
- How can we attract people to make use of declarative programming?
- Single-system versus Multi-system teams? Fine-tuning?
- Can we develop a uniform language — propositional , predicate?
- How can we attract neighboring communities to participate?
- What are your questions?