TLP **17** (4): 359–364, 2017.
doi:10.1017/S1471068417000199

Preface

AGOSTINO DOVIER

Department of Mathematics, Computer Science, and Physics, CLPLAB, University of Udine 33100 Udine, Italy
(*e-mail*: agostino.dovier@uniud.it)

submitted 20 June 2017; revised 22 June 2017; accepted 22 June 2017

Magic squares, chess-like problems, cryptarithmetic puzzles, and similar classes of problems have been extensively used to challenge human reasoning capabilities. *Lo Shu* magic square can be traced back to 650 B.C., the eight-queens problem was proposed in 1848 by the chess player Max Bazzel TWO × TWO = THREE puzzle appeared in *Strand Magazine* in 1924. These puzzles are nowadays widely used in constraint programming courses. The first programming language provided with constraint modelling primitives (Sketchpad) has been proposed by the Turing award winner Ivan Sutherland in his PhD thesis (1963). Logemann and Loveland, when implementing the Davis–Putnam procedure (Davis and Putnam 1960) for testing the satisfiability of a propositional formula (SAT), devised an algorithm (Davis–Putnam–Logemann–Loveland (DPLL)) that has become the core of all SAT/and Answer Set Programming solvers (50 years later). It consists in choosing an unassigned variable, assigning it a value 0 or 1, propagating the chosen value (unit propagation), and proceeding with the alternative value, if the original assignment leads to a contradiction (backtracking). Some years later Waltz (1975) introduced the notion of domain filtering (arc-consistency-based constraint propagation). With this idea the same DPLL scheme can be used for verifying the satisfiability of a constraint satisfaction problem, where the assignment is no longer 0/1 and the unit propagation is replaced by constraint propagation. For a detailed history of these early years achievements, we refer the reader to the works by Loveland *et al.* (2017), Jaffar and Maher (1994), and Freuder and Mackworth (2006).

A major breakthrough occurred in 1986: Jaffar, Lassez, and Maher proposed the Constraint Logic Programming (CLP) scheme (Jaffar *et al.* 1986; Jaffar and Lassez 1987). The elegance of their proposal is astonishing. Given a constraint domain (e.g., finite domains (FD), infinite trees), a declarative modeling language on this domain is embedded in a declarative programming language. The authors demonstrated how a small set of conditions are sufficient to derive a sound and complete logic programming paradigm. The model theoretical results of definite clause programming (e.g., minimum model) are guaranteed and a modular extension of SLD resolution is defined. Several researchers worked on the CLP scheme presenting instances on Boolean, FD, Rationals, Reals, Sets, Multisets, Graphs, etc.—for details, we refer to the survey by Jaffar and Maher (1994). Some implementations were immediately incorporated in Prolog systems, and they are

still maintained, in particular, for CLP (FD) that is efficiently supported by most systems.

Although the CLP scheme is Turing complete, expressivity for knowledge representation (common sense reasoning, default reasoning, ...) is limited, due to the poor handling of negation. As in Prolog, the goal-oriented resolution scheme is capable of dealing with a stratified use of negation only, with additional requirements on the use of variables (e.g., safety)—but some implementations based on well-founded semantics have been proposed (e.g., XSB Prolog, Rao *et al.* 1997). Stuckey (1991) proposed to use constructive negation which is, basically, a rewriting approach to negation. It defines some sufficient conditions that the program and the domain must satisfy. Dovier *et al.* (2000) proved that those conditions are also necessary and, in fact, they strongly limit the capability of handling negation in CLP. At the end of a long series of approaches put forward by many researchers (see, e.g., the volume edited by Minker 1988 for an overview), Gelfond and Lifschitz (1988) proposed what is now broadly accepted as *the* semantics of programs with negation, namely the stable model semantics—again, save for some applications where well-founded semantics (which is computable in polynomial time from a finite program, as shown by Brass *et al.* 2001) is acceptable. If incorporated in a Turing complete framework (as CLP), this would immediately lead to undecidability, but decidability is ensured in the case of finiteness of the *grounding* of the program (and there are sufficient syntactical properties to guarantee that). Thus, a new logic programming paradigm, strongly based on model-theoretical semantics, was born: Answer Set Programming, or simply ASP (Marek and Truszczynski 1999). This is the second major breakthrough of our story. Establishing whether a stable model exists is NP-complete, or even Σ_2^P complete if programs with disjunctive heads are considered (w.r.t. the size of the grounded program—see, for instance, Eiter and Gottlob 1995). The language originates from knowledge representation desiderata, but it emerged that it is also a perfect framework for encoding intractable combinatorial problems. As a matter of fact, Niemelä (1999) shows how to encode constraint satisfaction problems (CSP) in ASP. The link between ASP and Constraint Programming has been established and the emerging ASP solvers (smodels, Simons 2000, cmodels, Lierler and Maratea 2004, and DLV, Leone *et al.* 2006) have made ASP effective.

A comparison of the two families of logic paradigms for encoding CSP, namely CLP (FD) and ASP, was made by Dovier *et al.* (2005). Although the tests are nowadays obsolete, the overall considerations still hold: As long as variable domains remain "small" (e.g., Boolean), ASP encoding is faster than CLP (FD) (even if one might appreciate the possibility of easily defining problem-driven search heuristics in Prolog, while ASP solvers are mostly used as black boxes). The situation is reversed when domains are "large," since in ASP grounding issues arise and the power of constraint propagation (in particular, with global constraints) emerges with large domains. As a programming methodology, it was also pointed out how the availability in ASP of clauses without head allows a compact encoding of universal quantification that, instead, should be dealt with recursion using lists in CLP.

In 2002, the SAT community started the SAT competition: SAT solvers are challenged using a set of benchmarks given in DIMACS

https://doi.org/10.1017/S1471068417000199 Published online by Cambridge University Press

format (van Maaren and Franco 2002). Solvers were based on DPLL core, but a major breakthrough took place with the encoding of conflict-driven clause learning in solvers—for details, we refer to the paper by Silva and Sakallah (1999). The same idea was inherited by the ASP solver Clasp (Gebser *et al.* 2009) that is currently the fastest ASP solver (according to the results of the ASP competition, organized since 2009 Calimeri *et al.* 2016). The idea can be implemented in constraint solvers as well (Stuckey 2010), and it is proved to be effective in the Constraint Programming competitions (MiniZinc Challenge, organized since 2008 Stuckey *et al.* 2014), although for this framework excellent implementations of global constraints can still be more effective. These competitions allow a continuous improvement of solver's performance. CLP can exploit them since the constraint solvers can be interfaced with the Prolog system. It must be said, however, that the constraint community is slowly abandoning CLP moving towards dedicated modeling languages, like MiniZinc.

Even if the solvers are becoming faster every day, ASP and SAT are intrinsically limited by the grounding size explosion especially for problems that require the encoding of large integer intervals. In response, several proposals emerged in both communities for adding constraints on different domains that can be dealt with mixed techniques using external solvers, leading to research directions known as satisfiability modulo theories (SMT) (Nieuwenhuis *et al.* 2006), and adding constraints to answer set programming (e.g., Mellarkod *et al.* 2008; Baselice *et al.* 2015). Although various names have been proposed in earlier proposals, we refer simply to this research direction as CASP (Constraint ASP).

Another integration of CLP and ASP is presented by Dal Palù *et al.* (2009) where CLP (FD) is used for computing stable models delaying grounding as much as possible and exploiting the constraint-based reasoning for efficiently computing deterministic consequences of a current sets of choices. Finally, the language Picat (Zhou *et al.* 2015) is capable of mixing Constraint Solving, SAT solving, and Mixed Integer Programming, exploiting an efficient tabling mechanism for storing intermediate solutions without the need of recomputing them. It has proved to be extremely efficient in solving planning problems.

The purpose of this special issue of TPLP is to investigate recent results on this emerging modern notion of *CLP* that builds on all the developments mentioned above. Among the nine submission received, seven papers have been selected after two rounds of reviews. All authors are leading researchers in the area. We thank all authors who submitted their contributions to this special issue.

The issue starts with a paper from one of the inventors of CLP:

Michael J. Maher.
Contractibility for open global constraints.

The paper is in the mainstream of constraint programming. "Open" here means that new variables can be added to the global constraint during the computation. The author provides a characterization—called contractibility—of the constraints, where filtering remains sound also when the constraint is open.

https://doi.org/10.1017/S1471068417000199 Published online by Cambridge University Press

We have then two papers on the advancements of CASP solvers:

Mutsunori Banbara, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub.
Clingcon: The next generation.

Marcello Balduccini and Yuliya Lierler.
Constraint answer set solver EZCSP and why integration schemas matter.

The following is a contribution from a leading research group of SMT:

Andrew Reynolds, Cesare Tinelli, Clark Barrett.
Constraint solving for finite model finding in SMT solvers.

Then a paper that, in the spirit of this issue, bridges the two communities:

Yuliya Lierler, Benjamin Susman.
On relation between constraint answer set programming and satisfiability modulo theories.

Subsequently, an application paper showing how CASP can be successfully used in domains written in Planning Domain Definition Language extended with continuous update expressions (namely PDDL+ domains):

Marcello Balduccini, Daniele Magazzeni, Marco Maratea, Emily LeBlanc.
CASP solutions for planning in hybrid domains.

Last, but not least, a paper that shows how using logic programming to model (and solve) Distributed Constraint Optimization Problems. The authors define an extension of ASP implementing a Distributed Pseudo-tree Optimization Procedure and they exploit the Linda infrastructure implemented in SICStus Prolog. They also show some application domains, including reasoning in a power distribution network.

Tiep Le, Tran Cao Son, Enrico Pontelli, William Yeoh.
Solving distributed constraint optimization problems using logic programming.

We hope you enjoy the reading.

Acknowledgements

We would like to thank the following first-class logic programming researchers that did a great job for our community by carefully reviewing the submitted papers: Marcello Balduccini, Roman Barták, Lukás Chrpa, Alessandro Dal Palù, Marc Denecker, Andrea Formisano, Marco Gavanelli, Martin Gebser, Willem-Jan van Hoeve, Tomi Janhuen, Tomi Juntilla, Marco Maratea, Jacopo Mauro, Pedro Meseguer, Angelo Montanari, Max Ostrowski, Enrico Pontelli, Francesco Ricca, Gianfranco Rossi, Tran Cao Son, Peter J. Stuckey, Neng-Fa Zhou, and Roie Zivan. Special thanks go to Mirek Truszczyński for the idea of this special issue and his precious help behind the scene.

References

Baselice, S., Bonatti, P. and Gelfond, M. 2015. *Proc. of the 21st International Conference on Logic Programming*, M. Gabbrielli and G. Gupta, Eds. Lecture Notes in Computer Science, vol. 3668, 52–66.

Brass, S., Dix, J., Freitag, B.and Zukowski, U. 2001. Transformation-based bottom-up computation of the well-founded model. *Theory and Practice of Logic Programming 1,* 5, 497–538.

Calimeri, F., Gebser, M., Maratea, M. and Ricca, F. 2016. Design and results of the fifth answer set programming competition. *Artificial Intelligence 231*, 151–181.

Dal Palù, A., Dovier, A., Pontelli, E. and Rossi, G. 2009. Answer set programming with constraints using lazy grounding. In *Proc. of 25th International Conference on Logic Programming, ICLP 2009*, Pasadena, CA, USA, July 14–17, 2009, P. M. Hill and D. S. Warren, Eds. Lecture Notes in Computer Science, vol. 5649. Springer, 115–129.

Davis, M. and Putnam, H. 1960. A computing procedure for quantification theory. *Journal of the ACM 7,* 3, 201–215.

Dovier, A., Formisano, A. and Pontelli, E. 2005. A comparison of CLP (FD) and ASP solutions to NP-complete problems. In *Proc. of 21st International Conference on Logic Programming, ICLP 2005*, Sitges, Spain, October 2–5, 2005, M. Gabbrielli and G. Gupta, Eds. Lecture Notes in Computer Science, vol. 3668. Springer, 67–82.

Dovier, A., Pontelli, E. and Rossi, G. 2000. A necessary condition for constructive negation in constraint logic programming. *Information Processing Letters 74,* 3–4, 147–156.

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

Freuder, E. C. and Mackworth, A. K. 2006. Constraint satisfaction: An emerging paradigm. In *Handbook of Constraint Programming*, F. Rossi, P. van Beek, and T. Walsh, Eds. Foundations of Artificial Intelligence, vol. 2. Elsevier, 13–27.

Gebser, M., Kaufmann, B. and Schaub, T. 2009. The conflict-driven answer set solver clasp: Progress report. In *Proc. of 10th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2009*, Potsdam, Germany, September 14-18, 2009, E. Erdem, F. Lin, and T. Schaub, Eds. Lecture Notes in Computer Science, vol. 5753. Springer, 509–514.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of the 5th International Conference and Symposium on Logic Programming*, Seattle, Washington, August 15–19, 1988 (2 Volumes), R. A. Kowalski and K. A. Bowen, Eds. MIT press, 1070–1080.

Jaffar, J. and Lassez, J. 1987. Constraint logic programming. In *Conference Record of the 14th Annual ACM Symposium on Principles of Programming Languages*, Munich, Germany, January 21–23, 1987. ACM, 111–119.

Jaffar, J., Lassez, J. and Maher, M. J. 1986. Logic programming language scheme. In *Logic Programming: Functions, Relations, and Equations.* Prentice-Hall, 441–467.

Jaffar, J. and Maher, M. J. 1994. Constraint logic programming: A survey. *Journal of Logic Programming 19/20*, 503–581.

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S. and Scarcello, F. 2006. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic 7,* 3, 499–562.

Lierler, Y. and Maratea, M. 2004. Cmodels-2: Sat-based answer set solver enhanced to non-tight programs. In *Proc. of 7th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2004*, Fort Lauderdale, FL, USA, January 6–8, 2004, V. Lifschitz and I. Niemelä, Eds. Lecture Notes in Computer Science, vol. 2923. Springer, 346–350.

Loveland, D., Sabharwal, A. and Selman, B. 2017. DPLL: The core of modern satisfiability solvers. In *Martin Davis on Computability, Computational Logic, and Mathematical Foundations*, E. G. Omodeo and A. Policriti, Eds. Springer, 315–335.

Marek, V. W. and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective*, K. R. Apt, W. Marek, M. Truszczyński, and D. S. Warren, Eds. Springer, 375–398 (also *CoRR* cs.LO/9809032, 1998).

Mellarkod, V. S., Gelfond, M. and Zhang, Y. 2008. Integrating answer set programming and constraint logic programming. *Annals of Mathematics and Artificial Intelligence 53,* 1–4, 251–287.

Minker, J. 1988. *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence 25,* 3–4, 241–273.

Nieuwenhuis, R., Oliveras, A. and Tinelli, C. 2006. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM 53,* 6, 937–977.

Rao P., Sagonas K., Swift T., Warren, D. S. and Freire J. 1997. XSB: A system for efficiently computing well-founded semantics. In *Proc. of Logic Programming and Nonmonotonic Reasoning—*, LNCS, vol. 1265. Springer Verlag, 431–440.

Silva, J. P. M. and Sakallah, K. A. 1999. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers 48,* 5, 506–521.

Simons, P. 2000. Extending and implementing the stable model semantics. Doctoral dissertation. Report 58, Helsinki University of Technology.

Stuckey, P. J. 1991. Constructive negation for constraint logic programming. In *Proc. of the 6th Annual Symposium on Logic in Computer Science (LICS '91)*, Amsterdam, The Netherlands, July 15–18, 1991. IEEE Computer Society, 328–339.

Stuckey, P. J. 2010. Lazy clause generation: Combining the power of SAT and CP (and MIP?) solving. In *Proc. of Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 7th International Conference, CPAIOR 2010*, Bologna, Italy, June 14–18, 2010, A. Lodi, M. Milano, and P. Toth, Eds. Lecture Notes in Computer Science, vol. 6140. Springer, 5–9.

Stuckey, P. J., Feydy, T., Schutt, A., Tack, G. and Fischer, J. 2014. The miniZinc challenge 2008–2013. *AI Magazine 35,* 2, 55–60.

Sutherland, I. E. 1963. *Sketchpad: A Man-Machine Graphical Communications System.* Technical Report 296, MIT, Lincoln Laboratory.

van Maaren, H. and Franco, J. 2002. SAT Competition. http://www.satcompetition.org/

Waltz, D. 1975. Understanding line drawings of scenes with shadows. In *The Psychology of Computer Vision*, P. H. Winston, Ed. McGraw-Hill, 19–91.

Zhou, N., Kjellerstrand, H. and Fruhman, J. 2015. *Constraint Solving and Planning with Picat*. Springer Briefs in Intelligent Systems. Springer.