TPLP **22** (6): 770–775, 2022. © The Author(s), 2022. Published by Cambridge University Press.
doi:10.1017/S1471068422000345 First published online 22 September 2022

Introduction to the Collection of Papers Celebrating the 20th Anniversary of TPLP

THOMAS EITER
Technische Universität Wien, Vienna, Austria
(*e-mail:* eiter@kr.tuwien.ac.at)

MICHAEL MAHER
Reasoning Research Institute, Canberra, Australia
(*e-mail:* michael.maher@reasoning.org.au)

ENRICO PONTELLI
New Mexico State University, Las Cruces, New Mexico, USA
(*e-mail:* epontell@cs.nmsu.edu)

LUC DE RAEDT
KU Leuven, Leuven, Belgium
(*e-mail:* luc.deraedt@cs.kuleuven.be)

MIROSLAW TRUSZCZYNSKI
University of Kentucky, Lexington, Kentucky, USA
(*e-mail:* mirek@cs.uky.edu)

submitted 6 July 2022; revised 18 July 2022; accepted 25 July 2022

The first issue of the journal Theory and Practice of Logic Programming, or TPLP, was published in January 2001. This issue, the last one in the present volume, and the following issue, the first one in the next volume, comprise a collection of papers commemorating and celebrating the twentieth anniversary of the journal. This celebratory collection comes with about one year delay due to the COVID-19 pandemic (but also, if we were to be entirely honest, because of a common human tendency to put things off). Whatever the true reason for the delay, the collection is finally here. We hope and expect it will prove to be a demonstration of the vitality of logic programming, and of a broad range of research directions it spawned in the past and continues to generate today.

Logic programming appeared as a scientific subarea of computer science in the early 1970s as a result of the happy confluence of research on automated theorem proving in first-order logic and the original implementation of the Prolog programming language. The presence of these two original sources of inspiration has been distinctly felt over the years. On the one hand, logic programming attracted theoreticians pursuing deeper and highly nuanced understanding of the semantics of logic programs; on the other hand, it drew in researchers whose goal was to advance the repertoire of logic programming tools by refining, perfecting, and expanding Prolog, proposing and implementing new computational paradigms for logic programming, and developing methods to build and analyze logic programs. Moreover, and it also goes back to its very origins, logic programming attracted researchers interested in applications such as natural language processing, database querying, constraint solving, planning, learning, and knowledge representation, to name but a few.

https://doi.org/10.1017/S1471068422000345 Published online by Cambridge University Press

The high level of research activity in logic programming led to the founding, in 1984, of a dedicated journal, the Journal of Logic Programming, and then, in 1986, of a professional organization, the Association for Logic Programming. The Association took on the responsibility to oversee the journal and to serve as a guardian of its high standards. It also took on the organization of annual meetings dedicated to the advancement of logic programming.

The Journal of Logic Programming was a title of the Elsevier publishing company. In the late 1990s, concerns about the growing subscription costs prompted the Association to seek alternative publishers that could guarantee access to the journal at a lower price. In 1999, an agreement was made with Cambridge University Press to take over publishing the journal under the new name of *Theory and Practice of Logic Programming* or *TPLP*, for short. The papers in the backlog of the Journal of Logic Programming when the agreement was made were published in 2000 still in the Journal of Logic Programming. The first issue of the TPLP journal appeared early in 2001 under the leadership of Jack Minker as the Editor-in-Chief and Maurice Bruynooghe as the Managing Editor, the latter moving to that position after serving as Editor-in-Chief of the Journal of Logic Programming up to the time of the transition.

Once established, the TPLP journal took over as the flagship archival publication venue for research in logic programming. As its predecessor, it is run under the oversight of the Association for Logic Programming. In particular, it is the Association that selects the Editor-in-Chief for the journal, and the Editor-in-Chief makes annual reports to the Association during its Executive Committee meetings and general Association business meetings. The model has served the journal well. It helped maintain the highest, most rigorous quality standards and the standing as the go-to publication venue for the best logic programming research. The close engagement of the Association for Logic Programming in the journal also resulted in an agreement with the publisher according to which the regular papers from the annual International Conference on Logic Programming, organized and run by the Association, are published concurrently with the conference as a special issue of the journal. On the one hand, this elevates the status of the conference regular papers and, on the other, helps the journal maintain its cutting-edge image.

The collection of papers in this issue and in the next one celebrate the TPLP journal and the role it has played in the advancement of logic programming. We are especially pleased that as a happy, even if unintended, consequence of the delays that occurred, the first one of the two issues is published in 2022, the year of the 50th anniversary of Prolog. Without Prolog there would be no logic programming. And so, in some way our anniversary volume dedicated to the TPLP journal is also an homage to Prolog. In fact, the first paper in this issue directly addresses that topic.

This celebratory collection of papers was co-edited by Thomas Eiter, Michael Maher, Enrico Pontelli, Luc De Raedt, and Miroslaw Truszczynski. The work started with a call for paper proposals issued in 2019. Twenty-two proposals were received. The authors of twelve of these proposals were invited to submit full papers. All invitations were accepted and the full papers were submitted for review. After several rounds of peer review each paper was accepted by the guest editors for publication in the special issue.

The twelve papers are devoted to a wide range of topics, where the authors survey and discuss contributions that have been made, complemented with occasional new results

https://doi.org/10.1017/S1471068422000345 Published online by Cambridge University Press

and addressing also future perspectives. Below, we present brief descriptions of all papers in the collection.

Fifty Years of Prolog and Beyond is authored by Philipp Körner, Michael Leuschel, João Barbosa, Vítor Santos Costa, Verónica Dahl, Manuel Hermenegildo, Jose Morales, Jan Wielemaker, Daniel Diaz, Salvador Abreu, and Giovanni Ciatto. It discusses Prolog systems, which have been a mainstay of the logic programming community. The paper provides an overview of the evolution of Prolog implementations and features up until the Prolog standard certified by ISO in 1995. It then analyses the current state of Prolog systems. The paper concludes that current systems – while remaining largely compliant with the standard – have diverged in features, in response to requirements of different user communities and developers interests. The authors make the case for another community effort to standardize features, in the interest of code-sharing and portability of Prolog code across systems.

While, in the previous paper, the development of Prolog is seen as occurring through the pragmatic innovation and refinement of features, Dale Miller, in *The Proof-Theoretic Foundations of Logic Programming*, looks at logic programming language design through the lens of structural proof theory, which is based on Gentzen's sequent calculus. He uses the notion of an abstract logic programming language as a framework in which to position a variety of designs of logic programming languages with features ranging from modularity and higher-order expressions to linearity (in the sense of linear logic), polarity, and focus. The last two together support both backward- and forward-chaining inference. Thus, the paper describes a contrasting vision of how logic programming languages could develop in the future.

Parallel execution of logic programming has been an exciting concept since the inception of the logic programming paradigm, thanks to its declarative nature and the minimal amount of sequential control. A team of logic programming researchers published, in 2001, a comprehensive survey exploring 20 years of research, challenges, and accomplishments in the field of parallel logic programming. In *Parallel Logic Programming: A Sequel*, Agostino Dovier, Andrea Formisano, Gopal Gupta, Manuel Hermenegildo, Enrico Pontelli, and Ricardo Rocha explore how the field has evolved over the last 20 years. The survey starts where the 2001 survey stopped, by initially reviewing the key challenges in parallel execution of Prolog, with an emphasis on the actual systems developed and currently available, and with particular focus on the use of static analysis techniques. From there, the survey moves into exploring how parallelism has been explored in the context of more recent logic programming paradigms, in particular in the context of tabled logic programming and answer set programming. The survey also explores the connections between logic programming and a diversity of parallel architectures, such as graphical processing units and cloud computing platforms.

Constrained Horn clauses (CHCs) are constraint logic programs used as a representation, rather than a means to produce computations. In *Analysis and Transformation of Constrained Horn Clauses for Program Verification* by Emanuele De Angelis, Fabio Fioravanti, John P. Gallagher, Manuel V. Hermenegildo, Alberto Pettorossi, and Maurizio Proietti, the authors explain how CHCs are used to verify properties of programs in imperative programming languages. They survey several methods that can be used to derive CHCs from a program. They also describe the transformations and analyses of

https://doi.org/10.1017/S1471068422000345 Published online by Cambridge University Press

CHCs that have proven useful in verification tasks, and discuss their combination. The paper concludes with a discussion of research challenges in this area.

In *Thirty Years of Epistemic Specifications*, Jorge Fandinno, Wolfgang Faber, and Michael Gelfond survey three decades of research on an extension of logic programming with subjective literals. These literals, constructed with the help of modal operators, are meant to express truth of objective literals in all or in some stable models of a program (more generally in some class of interpretations of the program). They yield elegant representations of the open and closed world assumptions, and offer concise and intuitive encodings of epistemic knowledge relevant to many application domains. The paper surveys the motivations for the formalism, the main semantics proposed in the literature, and their properties.

The need for reasoning about the world in the context of time has led to many temporal logics based on different approaches. In particular, an extension of propositional logic with modal operators to express that some statements hold at the next point in time, somewhere along a discrete timeline, or until another statement holds has found important applications in verification, where properties of evolving systems are formally described and assessed. In *Linear-Time Temporal Answer Set Programming*, Felicidad Aguado, Pedro Cabalar, Martín Diéguez, Gilberto Pérez, Torsten Schaub, Anna Schuhmann, and Concepción Vidal survey and further advance an extension of answer set programming based on Pearce's Equilibrium Logic to linear temporal logic (LTL). The resulting Temporal Equilibrium Logic (TEL) comes in two flavors, namely, for infinite and for finite linear traces, which are important in many applications. Notably, the rich body of foundational results on TEL is complemented with an overview of *telingo*, an implementation of a fragment of TEL and a valuable tool for designing and realizing algorithms for planning and other temporal reasoning problems.

Understanding belief revision and belief updates have long been recognized as one of the fundamental problems in artificial intelligence. The survey by João Leite and Martin Slota, *A Brief History of Updates of Answer-Set Programs*, provides a comprehensive look of a version of this problem, specific to logic programming, in which programs are updated by *programs*. The formalism proposed for this research is that of *dynamic logic programs* that allows one to model sequences of updates as they become available over time. The paper surveys two main research directions on the semantics of dynamic logic programs, one driven by syntactic considerations and the other one based on a semantic analysis. The survey concludes with a thoughtful assessment of the current understanding of the field and a list of important remaining open problems.

Closely related to belief change is forgetting, which aims at reducing the vocabulary over which a knowledge base is formulated while preserving its semantics. In the article *Forgetting in Answer Set Programming*, Ricardo Gonçalves, Matthias Knorr, and João Leite survey the complex landscape of approaches to this operation in answer set programming, which has been studied for more than 15 years. In a systematic manner they review properties that a forgetting operator should satisfy, discuss concrete operators for forgetting, and consider which properties are satisfied by which operators. The article collects many results scattered in the literature but also presents novel results that fill gaps to provide a comprehensive and detailed picture of the landscape of forgetting, which is supplemented with helpful guidance for selecting suitable forgetting operators for different applications.

https://doi.org/10.1017/S1471068422000345 Published online by Cambridge University Press

Among the many syntactic extensions developed over the years for the answer set programming paradigm, the notion of aggregates has been one of the most popular and most widely studied. In spite of a long history, the literature has often been divided on what is the most appropriate way of extending the answer set semantics to provide a meaning to aggregate constructs. In *Aggregate Semantics for Propositional Answer Set Programs*, Mario Alviano, Wolfgang Faber, and Martin Gebser provide a comprehensive survey of the long history of semantic characterizations of aggregate constructs in the context of ground logic programs under the answer set semantics. The survey provides not only a review of the proposed semantics, but it provides a formal characterization and contextualization, which is useful to appreciate strengths, weaknesses, and differences among the different semantics.

In *Constraint Answer Set Programming: Integrational and Translational (or SMT-based) Approaches*, Yuliya Lierler presents a unifying perspective on the state-of-the-art of the investigations aiming to expand answer set programming with expressions to model constraints. This line of research resulted in formalisms allowing for more natural, direct, and concise modeling of search and optimization problems and, at the same time, improved the efficiency of search for solutions. In her paper, Lierler identifies two main approaches to the design of solvers for constraint answer set programs, one in which ideas developed for search for answer sets are integrated with techniques developed in the constraint satisfaction area, and another one where constraint answer set programs are translated into satisfiability modulo theories expressions. She then goes on to present key technical design ideas used in the current generation of solvers of both types.

Logic programming has a long history of serving as an effective framework to investigate provably correct solutions to challenges in reasoning about actions and change. The term Answer Set Planning has been introduced to capture the use of answer set programming to generate plans that transform the state of the world to meet given goals. The article *Answer Set Planning: A Survey* by Tran Cao Son, Enrico Pontelli, Marcello Balduccini, and Torsten Schaub provides a comprehensive survey of how answer set programming has enabled the exploration of solutions to planning problems, from classic problems to problems that arise in highly challenging domains, such as conformant planning, conditional planning, and planning with preferences. The survey explores also answer set planning solutions in the context of multi-agent systems and in presence of resources. The survey addresses strengths and weaknesses of answer set planning in different planning domains and helps in identifying open challenges for future research.

The final contribution in this volume, titled *How to Build Your Own ASP-based System*, by Roland Kaminski, Javier Romero, Torsten Schaub, and Philipp Wanko, provides an informative overview of how the answer set programming system *clingo* can be used as a foundation for the development of special purpose systems extending answer set programming. The paper explores two approaches to this challenge. The first approach relies on the use of meta-programming, supported by *clingo*'s reification capabilities. The second approach takes advantage of *clingo*'s Python API, which allows developers to interact with the entire answer set programming workflow, from modeling, to grounding, to solving.

The first five of these papers can be found in the present issue (issue 6 of volume 22 of TPLP), the remaining seven in the next one (issue 1 of volume 23 of TPLP).

https://doi.org/10.1017/S1471068422000345 Published online by Cambridge University Press

Acknowledgements

Needless to say, the volume would have never become a reality if not for the hard work of the authors. We thank them for their commitment to the project, and all the time and effort they put in the preparation of excellent articles. Further thanks are due to many anonymous reviewers who carefully read the submissions and made extensive detailed comments and improvement suggestions, working under a tight timetable. The volume is clearly much better because of that. Lastly, we thank our publisher, Cambridge University Press for their support throughout the project.

https://doi.org/10.1017/S1471068422000345 Published online by Cambridge University Press