Published: 19th September 2020
DOI: 10.4204/EPTCS.325
ISSN: 2075-2180

EPTCS 325

Proceedings 36th International Conference on
Logic Programming
(Technical Communications)

UNICAL, Rende (CS), Italy, 18-24th September 2020

Edited by: Francesco Ricca, Alessandra Russo, Sergio Greco, Nicola Leone, Alexander Artikis, Gerhard Friedrich, Paul Fodor, Angelika Kimmig, Francesca Lisi, Marco Maratea, Alessandra Mileo and Fabrizio Riguzzi

Preface

This volume contains the Technical Communications and the Doctoral Consortium papers of the 36th International Conference on Logic Programming (ICLP 2020), held virtually in Rende (CS), Italy, from September 20th to 25th, 2020. This is the first time that ICLP is run remotely together with all co-located events due to the COVID pandemic.

Since the first conference held in Marseille in 1982, ICLP has been the premier international event for presenting research in logic programming. The scope of the conference covers all areas of logic programming including, but not restricted to:

Foundations:
: Semantics, Formalisms, Answer-Set Programming, Non-monotonic Reasoning, Knowledge Representation.

Declarative Programming:
: Inference engines, Analysis, Type and mode inference, Partial evaluation, Abstract interpretation, Transformation, Validation, Verification, Debugging, Profiling, Testing, Logic-based domain-specific languages, constraint handling rules.

Related Paradigms and Synergies:
: Inductive and Co-inductive Logic Programming, Constraint Logic Programming, Interaction with SAT, SMT and CSP solvers, Logic programming techniques for type inference and theorem proving, Argumentation, Probabilistic Logic Programming, Relations to object-oriented and Functional programming, Description logics, Neural-Symbolic Machine Learning, Hybrid Deep Learning and Symbolic Reasoning.

Implementation:

Concurrency and distribution, Objects, Coordination, Mobility, Virtual machines, Compilation, Higher Order, Type systems, Modules, Constraint handling rules, Meta-programming, Foreign interfaces, User interfaces.

Applications:

Databases, Big Data, Data Integration and Federation, Software Engineering, Natural Language Processing, Web and Semantic Web, Agents, Artificial Intelligence, Bioinformatics, Education, Computational life sciences, Education, Cybersecurity, and Robotics.

Besides the main track, ICLP 2020 included the following additional tracks and special sessions:

Applications Track: This track invites submissions of papers on emerging and deployed applications of LP, describing all aspects of the development, deployment, and evaluation of logic programming systems to solve real-world problems, including interesting case studies and benchmarks, and discussing lessons learned.

Sister Conferences and Journal Presentation Track: This track provides a forum to discuss important results related to logic programming that appeared recently (from January 2018 onwards) in selective journals and conferences but have not been previously presented at ICLP.

Research Challenges in Logic Programming Track: This track invites submissions of papers describing research challenges that an individual researcher or a research group is currently attacking. The goal of the track is to promote discussions, exchange of ideas, and possibly stimulate new collaborations. Papers submitted to this track do not go through the usual review and will not be published in the proceedings they will be distributed at the conference as a technical report.

Special Session. Women in Logic Programming: This track aims to increase the visibility and impact of women in LP. It invites submissions of papers in all areas of logic programming co-authored by at least one woman and includes invited talks and presentations by women in logic programming.

The organizers of ICLP 2020 were:

General Chairs

- Sergio Greco, University of Calabria, Italy
- Nicola Leone, University of Calabria, Italy

Program Chairs

- Francesco Ricca, University of Calabria, Italy
- Alessandra Russo, Imperial College London, UK

Organizing Chairs

- Marco Calautti, University of Calabria, Italy
- Carmine Dodaro, University of Calabria, Italy

Publicity Chair

- Laura Pandolfo, Universitá degli Studi di Sassari, Italy

Applications Track Chairs

- Alexander Artikis, University of Piraeus & NCSR Demokritos, Greece
- Angelika Kimmig, Cardiff University, UK

Research Challenges in Logic Programming Track Chairs

- Gerhard Friedrich, Universität Klagenfurt, Austria
- Fabrizio Riguzzi, Universitá di Ferrara, Italy

Sister Conferences and Journal Presentation Track Chairs

- Paul Fodor, Stony Brook University, USA
- Marco Maratea, University of Genova, Italy

Women in Logic Programming Special Session Chairs

- Francesca Alessandra Lisi, University of Bari Aldo Moro, Italy
- Alessandra Mileo, INSIGHT Centre for Data Analytics, Dublin City University, Irland

Workshops Chair

- Martin Gebser, University of Potsdam, Germany

Doctoral Consortium Chairs

- Daniela Inclezan, Miami University, USA
- Bart Bogaerts, Vrije Universiteit Brussel, Belgium

Programming Competition Chairs

- Jessica Zangari, University of Calabria, Italy
- Markus Hecher, Vienna University of Technology, Austria

Three kinds of submissions were accepted:

- Technical papers for technically sound, innovative ideas that can advance the state of logic programming;
- Application papers that impact interesting application domains;
- System and tool papers which emphasize novelty, practicality, usability, and availability of the systems and tools described.

ICLP implemented the hybrid publication model used in all recent editions of the conference, with journal papers and Technical Communications (TCs), following a decision made in 2010 by the Association for Logic Programming. Papers of the highest quality were selected to be published as rapid publications in this special issue of TPLP. The TCs comprise papers which the Program Committee (PC) judged of good quality but not yet of the standard required to be accepted and published in TPLP as well as extended abstracts from the different tracks and dissertation project descriptions stemming from the Doctoral Program (DP) held with ICLP.

We have received 88 submissions of abstracts, of which 77 resulted in paper submissions, distributed as follows: ICLP main track (53), Applications track (7 full papers and 1 short papers), Sister Conferences and Journal Presentation track (6), Women in Logic Programming session (5 full papers and 3 short papers) and Research Challenge Track (2 short papers). The Program Chairs organized the refereeing process, which was undertaken by the PC with the support of external reviewers. Each technical paper was reviewed by at least three referees who provided detailed written evaluations. This yielded submissions short-listed as candidates for rapid communication. The authors of these papers revised their submissions in light of the reviewers suggestions, and all these papers were subject to a second round of reviewing. Of these candidates papers, 27 were accepted as rapid communications, to appear in the special issue. In addition, the PC recommended 40 papers to be accepted as technical communications, to appear at Electronic Proceedings in Theoretical Computer Science (EPTCS) either as full papers or extended abstracts, of which 38 were also presented at the conference (2 were withdrawn).

We are deeply indebted to the Program Committee members and external reviewers, as the conference would not have been possible without their dedicated, enthusiastic and outstanding work. The Program Committee members of ICLP 2020 were:

Mario Alviano Michael Gelfond Ricardo Rocha

Nicos Angelopoulos
Marcello Balduccini
Mutsunori Banbara
Chitta Baral
Roman Barták
Christoph Benzmüller
Alex Brik
François Bry
Pedro Cabalar
Francesco Calimeri
Manuel Carro
Angelos Charalambidis
Michael Codish
Stefania Costantini
Marc Denecker
Martín Diéguez
Carmine Dodaro
Agostino Dovier
Thomas Eiter
Wolfgang Faber
Thom Fruehwirth
Marco Gavanelli
Martin Gebser
Laura Giordano
Gopal Gupta
Michael Hanus
Manuel V. Hermenegildo
Katsumi Inoue
Tomi Janhunen
Jianmin Ji
Nikos Katzouris
Michael Kifer
Zeynep Kiziltan
Ekaterina Komendantskaya
Evelina Lamma
Michael Leuschel
Vladimir Lifschitz
Francesca Alessandra Lisi
Yanhong A. Liu
Marco Maratea
Viviana Mascardi
Yunsong Meng
Emilia Oikarinen
Magdalena Ortiz
Simona Perri
Enrico Pontelli
Chiaki Sakama
Torsten Schaub
Konstantin Schekotihin
Tom Schrijvers
Guillermo R. Simari
Tran Cao Son
Mohan Sridharan
Theresa Swift
Paul Tarau
Hans Tompits
Francesca Toni
Irina Trubitsyna
Mirek Truszczynski
German Vidal
Alicia Villanueva
David Warren
Jan Wielemaker
Stefan Woltran
Jia-Huai You
Shiqi Zhang
Neng-Fa Zhou

The Program Committee members of the Applications track were:

Nicos Angelopoulos
Chitta Baral
Alex Brik
François Bry
Francesco Calimeri
Angelos Charalambidis
Martín Diéguez
Martin Gebser
Gopal Gupta
Jianmin Ji
Nikos Katzouris
Zeynep Kiziltan
Marco Maratea
Yunsong Meng
Konstantin Schekotihin
Tom Schrijvers
Mohan Sridharan
Paul Tarau
David Warren
Jan Wielemaker
Shiqi Zhang
Neng-Fa Zhou

The external reviewers were:

Gianvincenzo Alfano
Elena Bellodi
Pierre Carbonnelle
Matthew Castellana
Wolfgang Dvořák
Serdar Erbatur
Francesco Fabiano
Jorge Fandinno
Andrea Formisano
David Fuenmayor
Tobias Geibinger
Rafael Kiesel
Vladimir Komendantskiy
François Laferriere
Pedro Lopez-Garcia
Julio Mariño
Elena Mastria
Djordje Markovich
Simon Marynissen
Jose F. Morales
Johannes Oetsch
Francesco Parisi
Sascha Rechenberger
Javier Romero
Elmer Salazar
Mantas Simkus
Takehide Soh
Fabio Tardivo
Yi Tong
David Tuckey
Sarat Chandra Varanasi

The 16th Doctoral Consortium (DC) on Logic Programming was held in conjunction with ICLP 2020. It attracts Ph.D. students in the area of Logic Programming Languages from different backgrounds (e.g. theoretical, implementation, application) and encourages a constructive and fruitful advising. Topics included: theoretical foundations of logic and constraint logic programming, sequential and parallel implementation technologies, static and dynamic analysis, abstract interpretation, compilation technology, verification, logic-based paradigms (e.g., answer set programming, concurrent logic programming, inductive logic programming) and innovative applications of logic programming. This year the Doctoral Consortium accepted 4 papers in the areas described above. We warmly thank all student authors, supervisors, referees, co-chairs, members of the program committee and the organizing team that made the Doctoral Consortium greatly successful.

The DC Program Committee members were:

Carmine Dodaro	Martin Gebser
Jorge Fandinno	Jose F. Moralesr
Fabio Fioravanti	Zeynep G. Saribatur
Paul Fodor	Frank Valencia

We would also like to express our gratitude to the full ICLP 2020 organization committee. Our gratitude must be extended to Torsten Schaub as current President and Thomas Eiter as incoming President of the Association of Logic Programming (ALP), to Marco Gavanelli in the role of conference-coordinator for ALP, to all the members of the ALP Executive Committee and to Mirek Truszczynski, Editor-in-Chief of TPLP. Also, to the staff at Cambridge University Press for their assistance. We would also like to thank Rob van Glabbeek, Editor-in-Chief of EPTCS, for helping the Program Chairs with their prompt support. Finally, we wish to thank each author of every submitted paper, since their efforts keep the conference alive and the participants to ICLP for bringing and sharing their ideas and latest developments. This is particularly appreciated in this 36th edition of the conference which has run during the unprecedented time of the COVID pandemic.

Francesco Ricca, Alessandra Russo, Sergio Greco, Nicola Leone, Alexander Artikis, Angelika Kimmig,
Gerhard Friedrich, Fabrizio Riguzzi, Paul Fodor, Marco Maratea, Francesca Alessandra Lisi, Alessandra Mileo (Eds.)

Applications of Answer Set Programming where Theory meets Practice

Esra Erdem (*Sabanci University, Turkey*)

We have been investigating applications of Answer Set Programming (ASP) in various domains, ranging from historical linguistics and bioinformatics to economics and robotics. In these applications, theory meets practice around challenging computational problems, and they all start a journey towards benefiting science and life. ASP plays an important role in this journey, sometimes as a declarative programming paradigm to solve hard combinatorial search problems (e.g., phylogeny reconstruction for Indo-European languages, multi-agent path finding in autonomous warehouses, matching problems in economics), and sometimes as a knowledge representation paradigm to allow deep reasoning and inferences over incomplete heterogeneous knowledge and beliefs of agents (e.g., hybrid planning for robotic manipulation, diagnostic reasoning in cognitive factories, explanation generation for biomedical queries). In this talk, we will share our experiences from such different applications of ASP, and discuss its role and usefulness from different perspectives.

---

2. Damiano Azzolini, Fabrizio Riguzzi, Evelina Lamma, Elena Bellodi & Riccardo Zese (2018): *Modeling Bitcoin Protocols with Probabilistic Logic Programming.* In Elena Bellodi & Tom Schrijvers, editors: Proceedings of the 5th International Workshop on Probabilistic Logic Programming, PLP 2018, co-located with the 28th International Conference on Inductive Logic Programming (ILP 2018), Ferrara, Italy, September 1, 2018,CEUR Workshop Proceedings2219, CEUR-WS.org, pp. 49–61. Available at http://ceur-ws.org/Vol-2219/paper6.pdf.
3. Satoshi Nakamoto (2008): *Bitcoin: A peer-to-peer electronic cash system.*
4. Joseph Poon & Thaddeus Dryja (2016): *The bitcoin lightning network: Scalable off-chain instant payments.*
5. Fabrizio Riguzzi (2018): *Foundations of Probabilistic Logic Programming.* River Publishers, Gistrup, Denmark.
6. Elias Rohrer, Julian Malliaris & Florian Tschorsch (2019): *Discharged Payment Channels: Quantifying theLightning Network's Resilience to Topology-Based Attacks.* In: 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), IEEE, pp. 347–356, doi:10.1109/EuroSPW.2019.00045.
7. István András Seres, László Gulyás, Dániel A Nagy & Péter Burcsi (2020): *Topological analysis of bitcoin's lightning network.* In: Mathematical Research for Blockchain Economy, Springer, pp. 1–12.

Accountable Protocols in Abductive Logic Programming (Extended Abstract)

Marco Gavanelli (*University of Ferrara*)
Marco Alberti (*University of Ferrara*)
Evelina Lamma (*University of Ferrara*)

Abstract

Finding the responsible of an unpleasant situation is often difficult, especially in artificial agent societies.

*S*CIFF is a language to define formal rules and protocols in agent societies, and an abductive proof-procedure for compliance checking. However, how to identify the responsible for a violation is not always clear.

In this work, a definition of accountability for artificial societies is formalized in *S*CIFF. Two tools are provided for the designer of interaction protocols: a guideline, in terms of syntactic features that ensure accountability of the protocol, and an algorithm (implemented in a software tool) to investigate if, for a given protocol, non-accountability issues could arise.

1 Introduction

The current economic world is strongly based on large corporations, that have been able to provide large economic benefits, such as cheaper prices for everyday goods and better employment rates, but that also represented large problems in case of failure. Every person can list problems in his/her own country in which a large firm misbehaved, e.g., polluting the environment, or by failing in a disastrous way causing huge losses for small investors. In many cases, the firm itself cannot be punished for its misbehavior, because a company cannot be sent to jail. One hopes that the culprit of the misbehavior (e.g., the CEO) is punished, but in many cases the complex behavior of an organization depends on the policies established by previous members (that might even be dead), by common practices, or by the fact that many individuals contributed to the disaster each for an inconceivably small amount.

In the literature of moral responsibility, there exist different notions of responsibility. Van de Poel et al. [3] distinguish five moral meanings of responsibility: accountability, blameworthiness, liability, obligation and virtue.

> The ascription of responsibility-as-accountability has the following implication: *i* is responsible-as-accountable for φ implies that *i* should account for (the occurrence of) φ, in particular for *i*'s role in doing, or bringing about φ, or for *i*'s role in failing to prevent φ from happening, where *i* is some agent, and φ an action or a state-of-affairs.

and

> Accountability implies blameworthiness unless the accountable agent can show that a reasonable excuse applies that frees her from blameworthiness. So holding an agent *i* accountable shifts the burden of proof for showing that *i* is not blameworthy to the agent *i*: the agent is now to show - by giving an account - that she is not blameworthy.

In this work, we focus on the *S*CIFF system [1], a complete system for defining and checking the compliance of agents to interaction protocols. It includes a language to define agent interaction protocols and to relate a current state of affairs with one or more expected behaviors of the agents, formalized as a set of expectations. The language was designed to leave freedom to agents, not overconstraining them to follow statically pre-defined paths, but, instead, to assert explicitly the obligatory actions and those that are forbidden, while leaving everything not explicitly stated as a possible action that an agent can perform if it is convenient. An abductive proof-procedure accepts asynchronous events and reasons about them through the protocol definition, generates the expected behavior of the agents, and checks if the actual behavior matches with the expectations.

*S*CIFF lacks a concept of responsibility, because expectations, unlike commitments, are not characterized by a debtor, due to different language design objectives: while *S*CIFF is able to detect violations of the protocols, it is not always clear which agent is responsible for the wrong state of affairs.

In this work, we address the problem by adopting the accountability version of responsibility. Accountability in the proposed setting stands for the possibility to account for the wrong state of affairs of an agent that is the one that performed (or did not perform) the action in its expected behavior. The agent might then, by reasoning on the protocol and the state of affairs (again, using the *S*CIFF proof procedure), be able to *account for* its own behavior. The agent might be able to find an explanation in which its expected behavior is fulfilled, and in such a case it cannot be held responsible for the violation. In some cases, this might happen because another agent is actually responsible for the violation, but in other cases it might be due to a wrong design of the interaction protocol.

For this reason, we define formally a notion of *accountability* of the interaction protocol. The idea is that a protocol is accountable if it allows to identify the agent (or agents) responsible for each possible violation. If the interactions in an organization or society are ruled by an accountable protocol, then, for each possible undesirable state of affairs, one or more agents will be unambiguously held responsible.

The formal definition allows us to provide guidelines and tools for the developer of interaction protocols. The guidelines are syntactic conditions that ensure a-priori the accountability of a protocol. We identify a syntactic characterization of a fairly large class of protocols and prove that protocols in such class are accountable. Various protocols taken from the list of *S*CIFF applications belong to this class.

However, even protocols that do not belong to the identified class may be accountable. For existing protocols, the user might not want to completely re-design the protocol, and in this case a tool that checks the protocol for accountability might be more suitable. We propose a tool to detect if a protocol has non-accountability issues. If there exists such an issue, the tool also provides a counterexample, i.e., a course of events with protocol violation, but for which no agent can be held responsible. We tested, through such tool, protocols modeled in the past with *S*CIFF, and we were

able to identify non-accountability of two protocols that were completely reasonable for the task for which they were designed. Thanks to the tool and the provided counterexample, it was then easy for the designer to fix the protocol.

References

1. Marco Alberti, Federico Chesani, Marco Gavanelli, Evelina Lamma, Paola Mello & Paolo Torroni (2008): *Verifiable Agent Interaction in Abductive Logic Programming: the SCIFF Framework*. *ACM Transactions on Computational Logic* 9(4), pp. 29:1-29:43, doi:10.1145/1380572.1380578.
2. Marco Gavanelli, Marco Alberti & Evelina Lamma (2018): *Accountable Protocols in Abductive Logic Programming*. *ACM Transactions on Internet Technology* 18(4), pp. 46:1-46:20, doi:10.1145/3107936.
3. Ibo Van de Poel, Lambèr Royakkers & Sjoerd D. Zwart (2015): *Moral Responsibility and the Problem of Many Hands*. Routledge Studies in Ethics and Moral Theory, Routledge, doi:10.4324/9781315734217.

Footnotes:

[1]The full version of this extended abstract can be found in [2].

Sampling-Based SAT/ASP Multi-Model Optimization as a Framework for Probabilistic Inference (Extended Abstract)

Matthias Nickles (*School of Computer Science, National University of Ireland, Galway*)

Email: matthias.nickles@nuigalway.ie

This extended abstract reports an earlier work [8] which introduced multi-model optimization through SAT witness or answer set sampling where the sampling process is controlled by a user-provided differentiable loss function over the multiset of sampled models. Probabilistic reasoning tasks are the primary use cases, including deduction-style probabilistic inference and hypothesis weight learning. Technically, our approach enhances a CDCL-based SAT and ASP solving algorithm to differentiable satisfiability solving (respectively differentiable answer set programming), by using a form of Gradient Descent as branching literal decision approach, and optionally a cost backtracking mechanism. Sampling of models using these methods minimizes a task-specific, user-provided multi-model loss function while adhering to given logical background knowledge (background knowledge being either a Boolean formula in CNF or a logic program under stable model semantics). Features of the framework include its relative simplicity and high degree of expressiveness, since arbitrary differentiable cost functions and background knowledge can be provided.

Keywords: *Boolean Satisfiability Problem*, *Probabilistic Logic Programming*, *Answer Set Programming*, *Differentiable Satisfiability*, *Weighted Sampling*, *PSAT*, *Optimization*

With this extended abstract, we report and summarize an earlier publication [8] which introduced an approach to finding an optimal multiset of satisfying Boolean variable assignments or answer sets by means of loss function gradient-steered sampling. The sampling process minimizes a user-defined objective function using a new optimization method which we call *Differentiable Satisfiability Solving* respectively *Differentiable Answer Set Programming* [7] (∂SAT/∂ASP).