TLP **14** (4–5): 401–414, 2014. © Cambridge University Press 2014
doi:10.1017/S1471068414000581

Introduction to the 30th International Conference on Logic Programming Special Issue

MICHAEL LEUSCHEL
University of Düsseldorf, Germany
(*e-mail:* leuschel@cs.uni-duesseldorf.de)

TOM SCHRIJVERS
Ghent University, Belgium
(*e-mail:* tom.schrijvers@ugent.be)

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

The 30th edition of the International Conference of Logic Programming took place in Vienna in July 2014 at the Vienna Summer of Logic – the largest scientific conference in the history of logic. Following the initiative in 2010 taken by the Association for Logic Programming and Cambridge University Press, the full papers accepted for the International Conference on Logic Programming again appear as a special issue of Theory and Practice of Logic Programming (TPLP) – the *30th International Conference on Logic Programming Special Issue*. Papers describing original, previously unpublished research and not simultaneously submitted for publication elsewhere were solicited in all areas of logic programming including but not restricted to: Theory: Semantic Foundations, Formalisms, Non- monotonic Reasoning, Knowledge Representation; Implementation: Compilation, Memory Management, Virtual Machines, Parallelism; Environments: Program Analysis, Transformation, Validation, Verification, Debugging, Profiling, Testing; Language Issues: Concurrency, Objects, Coordination, Mobility, Higher Order, Types, Modes, Assertions, Programming Techniques; Related Paradigms: Abductive Logic Programming, Inductive Logic Programming, Constraint Logic Programming, Answer-Set Programming; Applications: Databases, Data Integration and Federation, Software Engineering, Natural Language Processing, Web and Semantic Web, Agents, Artificial Intelligence, Bioinformatics.

There were four broad categories for submissions: (1) technical papers for describing technically sound, innovative ideas that can advance the state of the art of logic programming; (2) application papers, where the emphasis is on their impact on the application domain; (3) system and tool papers, where the emphasis is on the novelty, practicality, usability and general availability of the systems and tools described; and (4) technical communications, aimed at describing recent developments, new projects, and other materials that are not ready for main

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press

publication as standard papers. The length limit for full papers was set at 12 pages plus bibliography and for technical communications at 10 pages total. The papers appearing in this issue are classed as "TPLP rapid publications".

In response to the call for papers we received 80 abstract submissions of which 66 were submitted as full papers. The program chairs acting as guest editors organized the refereeing process with the help of the program committee and external reviewers. Each paper was reviewed by at least three anonymous referees who provided full written evaluations. Certain papers were conditionally accepted and were subject to a second round of reviewing. Finally, 25 papers were accepted as regular papers and 22 papers were accepted as technical communications. The latter appear as online supplement on the TPLP web page. Together, the journal special issue and the online supplement of short technical communications constitute the *proceedings* of ICLP14.

The list of the 25 accepted full papers appearing in this special issue follows. The abstracts of the technical communications can be found in Appendix A.

Dynamic Consistency Checking in Goal-Directed Answer Set Programming
Kyle Marple and Gopal Gupta.

Tabling, Rational Terms, and Coinduction Finally Together!
Theofrastos Mantadelis, Ricardo Rocha and Paulo Moura.

Efficient Computation of the Well-Founded Semantics over Big Data
Ilias Tachmazidis, Grigoris Antoniou and Wolfgang Faber.

The P-Box CDF-Intervals: A Reliable Constraint Reasoning with Quantifiable Information
Aya Saad, Thom Fruehwirth and Carmen Gervet.

Simulating Dynamic Systems Using Linear Time Calculus Theories
Bart Bogaerts, Joachim Jansen, Bruynooghe Maurice, Broes De Cat, Joost Vennekens and Marc Denecker.

A Linear Logic Programming Language for Concurrent Programming over Graph Structures
Flavio Cruz, Seth Goldstein, Frank Pfenning and Ricardo Rocha.

SUNNY: a Lazy Portfolio Approach for Constraint Solving
Roberto Amadini, Maurizio Gabbrielli and Jacopo Mauro.

Exchanging Conflict Resolution in an Adaptable Implementation of ACT-R
Daniel Gall and Thom Frühwirth.

Pengines: Web Logic Programming Made Easy
Torbjörn Lager and Jan Wielemaker.

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press

Incremental Tabling for Knowledge Representation and Reasoning
Terrance Swift.

claspfolio 2: Advances in Algorithm Selection for Answer Set Programming
Holger Hoos, Marius Lindauer and Torsten Schaub.

Vicious Circle Principle and Logic Programs with Aggregates
Michael Gelfond and Yuanlin Zhang.

Causal Graph Justifications of Logic Programs
Pedro Cabalar, Jorge Fandiño and Michael Fink.

On Termination, Confluence and Consistent CHR-based Type Inference
Gregory J. Duck, Rémy Haemmerlé and Martin Sulzmann.

Contextual Abductive Reasoning with Side-Effects
Luìs Moniz Pereira, Emmanuelle-Anna Dietz and Steffen Hölldobler.

A Proof Theoretic Study of Soft Concurrent Constraint Programming
Elaine Pimentel, Carlos Olarte and Vivek Nigam.

A Measure of Arbitrariness in Abductive Explanations
Luciano Caroprese, Mirek Truszczynski, Ester Zumpano and Irina Trubitsyna.

Lifted Variable Elimination for Probabilistic Logic Programming
Elena Bellodi, Evelina Lamma, Fabrizio Riguzzi, Vítor Santos Costa and Riccardo Zese.

Using Tabled Logic Programming to Solve the Petrobras Planning Problem
Roman Bartak and Neng-Fa Zhou.

On Cascade Products of Answer Set Programs
Christian Antic.

Minimum Model Semantics for Extensional Higher-order Logic Programming with Negation
Angelos Charalambidis, Zoltan Esik and Panos Rondogiannis.

Resource Usage Analysis of Logic Programs via Abstract Interpretation Using Sized Types
Alejandro Serrano, Pedro Lopez-Garcia and Manuel V. Hermenegildo.

Anytime Computation of Cautious Consequences in Answer Set Programming
Mario Alviano, Carmine Dodaro and Francesco Ricca.

A Module System for Domain-Specific Languages
Ethan K. Jackson.

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press

Abstract Diagnosis for tccp using a Linear Temporal Logic
Marco Comini, Laura Titolo and Alicia Villanueva.

As in previous years, four awards are handed out at the conference.

The program committee hands out awards to the two papers that they consider the *best* among the accepted papers. This year the *best paper* award goes to:

A Linear Logic Programming Language for Concurrent Programming over Graph Structures
Flavio Cruz, Seth Goldstein, Frank Pfenning and Ricardo Rocha.

The *best student paper* award is for:

On Cascade Products of Answer Set Programs
Christian Antic.

The program chairs also hand out *test of time* awards to the most influential papers published 10 and 20 years ago at the ICLP and ILPS conferences. In the 20 year category, the following paper from ILPS'1994 was selected to have withstood the test of time and to have had the largest impact:

CLP(Intervals) Revisited
Frédéric Benhamou, David A. McAllester, Pascal Van Hentenryck.

In the 10 year category, the following influental paper from ICLP'2004 was selected:

The Refined Operational Semantics of Constraint Handling Rules.
Gregory J. Duck, Peter J. Stuckey, Maria J. Garcìa de la Banda, Christian Holzbaur.

In conclusion, we would like to thank the members of the Program Committee and the external referees for their enthusiasm, hard work, and promptness, despite the higher load of the two rounds of refereeing plus the copy editing phase. The PC members were:

Elvira Albert, Sergio Antoy, Marcello Balduccini, Francois Bry, Mats Carlsson, Iliano Cervesato, Kaustuv Chaudhuri, Michael Codish, Danny De Schreye, Marc Denecker, Esra Erdem, Samir Genaim, Gopal Gupta, Michael Hanus, Remy Haemmerle, Ethan Jackson, Gerda Janssen, Michael Kifer, Andy King, Guenter Kniesel, Yanhong Annie Liu, Michael Maher, Rainer Manthey, Fred Mesnard, Jose Morales, Alberto Pettorossi, Gianfranco Rossi, Vitor Santos Costa, Peter Schachte, Torsten Schaub, Hirohisa Seki, Peter Stuckey, Paul Tarau, Michael Thielscher, Hans Tompits, Francesca Toni, German Vidal, Jan Wielemaker, Stefan Woltran, and Neng-Fa Zhou. The external referees were: Benjamin Andres, Gerald Berger, Bart Bogaerts, Martin Brain, Maurice Bruynooghe, Joana Côrte-Real, Rafael Caballero, Francesco Calimeri, Tran Cao Son, Aziem Chawdhary, Robert Craven, Emanuele De Angelis, Broes De Cat, Bart Demoen, Jo Devriendt, Alessandra Di Pierro, Agostino Dovier, Gregory Duck, Wolfgang Faber, Fabio Fioravanti, Julien Fischer, Graeme Gange, Martin Gebser, Gregory Gelfond, Miguel Gomez-Zamalloa,

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press

Hai-Feng Guo, Raúl Gutiérrez, Fahmida Hamid, Fredrik Heintz, Jacob Howe, Jinbo Huang, Daniela Inclezan, Nobuhiro Inuzuka, Joachim Jansen, Arne König, Roland Kaminski, Phuong Lan Nguyen, Markus Lepper, Jiefei Ma, Theofrastos Mantadelis, Marco Maratea, Kyle Marple, Enrique Martin-Martin, Barton Massey, Wannes Meert, Philipp Obermeier, Johannes Oetsch, Carlos Olarte, Max Ostrowski, Jörg Pührer, Sergey Paramonov, Etienne Payet, David Pearce, Maurizio Proietti, Patrick Prosser, Ricardo Rocha, Kostis Sagonas, Claudia Schulz, Dietmar Seipel, Mantas Simkus, Harald Sondergaard, Guido Tack, Sven Thiele, Laura Titolo, Baltasar Trancón Widemann, Pieter Van Hertum, Alicia Villanueva, and Neil Yorke-Smith.

Moreover, we are grateful to the invited speakers: Andrey Rybalchenko for his talk on "(Quantified) Horn Constraint Solving for Program Verification and Synthesis", and Neng-Fa Zhou for "Combinatorial Search with Picat". Manuel Carro (General Chair) and Haifeng Guo (Workshop Chair) have greatly contributed to the organization of ICLP 2014 and its affiliated events. Martin Gebser and Jael Kriener have our thanks for organizing the 10th edition of the ICLP Doctoral Consortium; the list of student papers accepted at the doctoral consortium can be found in Appendix B. We would also like to thank the organisers of the Vienna Summer of Logic for their support and for organising such a big event with such efficiency.

Finally, we would like to express our thanks and great appreciation to Ilkka Niemelä, editor in chief of Theory and Practice of Logic Programming, David Tranah from Cambridge University Press, and all the members of the ALP Executive Committee for their continued support for this initiative, which provides a new model of computer science publishing that is already being adopted by other computing research communities.

Michael Leuschel and Tom Schrijvers
Program Committee Chairs and Guest Editors

Supplementary material

The Technical Communications for this conference are available as supplementary material. To view supplementary material for this article, please visit http://dx.doi.org/10.1017/S1471068414000581.

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press

Appendix A Abstracts of Technical Communications

The 22 technical communication papers can be found in the online supplement of the special issue. Their abstracts are listed below.

On Strong and Default Negation in Answer-Set Program Updates
Martin Slota, Martin Balaz and Joao Leite.

Existing semantics for answer-set program updates fall into two categories: either they consider only *strong negation* in heads of rules, or they primarily rely on *default negation* in heads of rules and optionally provide support for strong negation by means of a syntactic transformation. In this paper we pinpoint the limitations of both these approaches and argue that both types of negation should be first-class citizens in the context of updates. We identify principles that plausibly constrain their interaction but are not simultaneously satisfied by any existing rule update semantics. Then we extend one of the most advanced semantics with direct support for strong negation and show that it satisfies the outlined principles as well as a variety of other desirable properties.

FO(C): A Knowledge Representation Language of Causality
Bart Bogaerts, Joost Vennekens, Marc Denecker and Jan Van den Bussche.

Cause-effect relations are an important part of human knowledge. In real life, humans often reason about complex causes linked to complex effects. By comparison, existing formalisms for representing knowledge about causal relations are quite limited in the kind of specifications of causes and effects they allow. In this paper, we present the new language C-Log, which offers a significantly more expressive representation of effects, including such features as the creation of new objects. We show how C-Log integrates with first-order logic, resulting in the language FO(C). We also compare FO(C) with several related languages and paradigms, including inductive definitions, disjunctive logic programming, business rules and extensions of Datalog.

A Well-Founded Semantics for FOL-Programs
Yi Bi, Jia-Huai You and Zhiyong Feng.

An FOL-program consists of a background theory in a decidable fragment of first-order logic and a collection of rules possibly containing first-order formulas. The formalism stems from recent approaches to tight integrations of ASP with description logics. In this paper, we define a well-founded semantics for FOL-programs based on a new notion of unfounded sets on consistent as well as inconsistent sets of literals, and study its properties. The semantics is defined for all FOL-programs, including those where it is necessary to represent inconsistencies explicitly. The semantics supports a form of combined reasoning by rules under closed world as well as open world assumptions, and it is a generalization of the standard well-founded semantics for normal logic programs. We also show that the well-founded semantics defined here approximates the well-supported answer set semantics for normal DL programs.

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press

Logic and Constraint Logic Programming for Distributed Constraint Optimization
Tiep Le, Enrico Pontelli, Tran Cao Son and William Yeoh.

The field of Distributed Constraint Optimization Problems (DCOPs) has gained momentum, thanks to its suitability in capturing complex problems (e.g., multi-agent coordination and resource allocation problems) that are naturally distributed and cannot be realistically addressed in a centralized manner. The state-of-the-art in solving DCOPs relies on the use of ad-hoc infrastructures and ad-hoc constraint solving procedures. This paper investigates an infrastructure for solving DCOPs that is completely built on logic programming technologies. In particular, the paper explores the use of a general constraint solver (a constraint logic programming system in this context) to handle the agent-level constraint solving. The preliminary experiments show that logic programming provides benefits over a state-of-the-art DCOP system, in terms of performance and scalability, opening the doors to the use of more advanced technology (e.g., search strategies, complex constraints) for solving DCOPs.

Properties of Stable Model Semantics Extensions
Mário Abrantes and Luís Moniz Pereira.

The stable model (SM) semantics lacks the properties of existence, relevance and cumulativity. If we prospectively consider the class of conservative extensions of SM semantics (i.e., semantics that for each normal logic program P retrieve a superset of the set of stable models of P), one may wander how do the semantics of this class behave in what concerns the aforementioned properties. That is the type of issue dealt with in this paper. We define a large class of conservative extensions of the SM semantics, dubbed affix stable model semantics, ASM, and study the above referred properties into two non-disjoint subfamilies of the class ASM, here dubbed ASMh and ASMm. From this study a number of results stem which facilitate the assessment of semantics in the class ASMh U ASMm with respect to the properties of existence, relevance and cumulativity, whilst unveiling relations among these properties. As a result of the approach taken in our work, light is shed on the characterization of the SM semantics, as we show that the properties of (lack of) existence and (lack of) cautious monotony are equivalent, which opposes statements on this issue that may be found in the literature; we also characterize the relevance failure of SM semantics in a more clear way than usually stated in the literature.

ESmodels: An Epistemic Specification Solver
Zhizheng Zhang.

ESmodels is designed and implemented as an experiment platform to investigate the semantics, language, related reasoning algorithms, and possible applications of epistemic specifications. We first give the epistemic specification language of ESmodels and its semantics. The language employs only one modal operator K but we prove that it is able to represent luxuriant modal operators by presenting transformation rules. Then, we describe basic algorithms and optimiza- tion approaches used in

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press

ESmodels. After that, we discuss possible applications of ESmodels in conformant planning and constraint satisfaction. Finally, we conclude with perspectives.

Clingo = ASP + Control: Preliminary Report
Martin Gebser, Roland Kaminski, Benjamin Kaufmann and Torsten Schaub.

We present the new ASP system clingo 4. Unlike its predecessors, being mere monolithic combinations of the grounder gringo with the solver clasp, the new clingo 4 series offers high-level constructs for realizing complex reasoning processes. Among others, such processes feature advanced forms of search, as in optimization or theory solving, or even interact with an environment, as in robotics or query-answering. Common to them is that the problem specification evolves during the reasoning process, either because data or constraints are added, deleted, or replaced. In fact, clingo 4 carries out such complex reasoning within a single integrated ASP grounding and solving process. This avoids redundancies in relaunching grounder and solver programs and benefits from the solver's learning capacities. clingo 4 accomplishes this by complementing ASP's declarative input language by control capacities expressed via the embedded scripting languages lua and python. On the declarative side, clingo 4 offers a new directive that allows for structuring logic programs into named and parameterizable subprograms. The grounding and integration of these subprograms into the solving process is completely modular and fully controllable from the procedural side, viz. the scripting languages. By strictly separating logic and control programs, clingo 4 also abolishes the need for dedicated systems for incremental and reactive reasoning, like iclingo and oclingo, respectively, and its flexibility goes well beyond the advanced yet still rigid solving processes of the latter.

Grounding Bound Founded Answer Set Programs
Rehan Abdul Aziz, Geoffrey Chu and Peter J. Stuckey.

Bound Founded Answer Set Programming (BFASP) is an extension of Answer Set Programming (ASP) that extends stable model semantics to numeric variables. While the theory of BFASP is defined on ground rules, in practice BFASP programs are written as complex non-ground expressions. Flattening of BFASP is a technique used to simplify arbitrary expressions of the language to a small and well defined set of primitive expressions. In this paper, we first show how we can flatten arbitrary BFASP rule expressions, to give equivalent BFASP programs. Next, we extend the bottom-up grounding technique and magic set transformation used by ASP to BFASP programs. Our implementation shows that for BFASP problems, these techniques can significantly reduce the ground program size, and improve subsequent solving.

Coinductive Logic Programming: Eager vs Lazy
Jónathan Heras, Ekaterina Komendantskaya and Martin Schmidt.

CoAlgebraic Logic Programming (CoALP) is a dialect of Logic programming designed to work with coinductive definitions of infinite objects. Its main goal

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press

is to introduce guarded lazy corecursion akin to functional theorem provers into logic programming. In this paper, we compare the lazy evaluation of coinductive programs in CoALP with the loop-analysis and eager execution in Coinductive Logic Programming (CoLP).

Towards an Efficient Prolog System by Code Introspection
George Souza Oliveira and Anderson Faustino Da Silva.

Several Prolog interpreters are based on the Warren Abstract Machine (WAM), an elegant model to compile Prolog programs. In order to improve the performance several strategies have been proposed, such as: optimize the selection of clauses, specialize the unification, global analysis, native code generation and tabling. This paper proposes a different strategy to implement an efficient Prolog System, the creation of specialized emulators on the fly. The proposed strategy was implemented and evaluated at YAP Prolog System, and the experimental evaluation showed interesting results.

Analysis and Transformation Tools for Constrained Horn Clause Verification
John P. Gallagher and Bishoksan Kafle.

Several techniques and tools have been developed for verification of properties expressed as Horn clauses with constraints over a background theory (CHC). Current CHC verification tools implement intricate algorithms and are often limited to certain subclasses of CHC problems. Our aim in this work is to investigate the use of a combination of off-the-shelf techniques from the literature in analysis and transformation of Constraint Logic Programs (CLPs) to solve challenging CHC verification problems. We find that many problems can be solved using a combination of tools based on well-known techniques from abstract interpretation, semantics-preserving transformations, program specialisation and query-answer transformations. This gives insights into the design of automatic, more general CHC verification tools based on a library of components.

Towards Assertion-based Debugging of Higher-Order (C)LP Programs
Nataliia Stulova, Jose F. Morales and Manuel V. Hermenegildo.

Higher-order constructs extend the expressiveness of first-order (Constraint) Logic Programming ((C)LP) both syntactically and semantically. At the same time assertions have been in use for some time in (C)LP systems helping programmers detect errors and validate programs. However, these assertion-based extensions to (C)LP have not been integrated well with higher order to date. Our work contributes to filling this gap by extending the assertion-based approach to error detection and program validation to the higher-order context, within (C)LP. It is based on an extension of properties and assertions as used in (C)LP in order to be able to fully describe arguments that are predicates.

A Simple and Efficient Lock-Free Hash Trie Design for Concurrent Tabling
Miguel Areias and Ricardo Rocha.

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press

A critical component in the implementation of a concurrent tabling system is the design of the table space. One of the most successful proposals for representing tables is based on a *two-level trie data structure*, where one trie level stores the tabled subgoal calls and the other stores the computed answers. In this work, we present a simple and efficient *lock-free design* where both levels of the tries can be shared among threads in a concurrent environment. To implement lock-freedom we took advantage of the *CAS* atomic instruction that nowadays can be widely found on many common architectures. *CAS* reduces the granularity of the synchronization when threads access concurrent areas, but still suffers from low-level problems such as false sharing or cache memory side-effects. In order to be as effective as possible in the concurrent search and insert operations over the table space data structures, we based our design on a *hash trie data structure* in such a way that it minimizes potential low-level synchronization problems by dispersing as much as possible the concurrent areas. Experimental results in the Yap Prolog system show that our new lock-free hash trie design can effectively reduce the execution time and scale better than previous designs.

Transaction Logic with (Complex) Events
Ana Sofia Gomes and Jose Julio Alferes.

This work deals with the problem of combining reactive features, such as the ability to respond to events and define complex events, with the execution of transactions over general Knowledge Bases (KBs).

With this as goal, we build on Transaction Logic (TR), a logic precisely designed to model and execute transactions in KBs defined by arbitrary logic theories. In it, transactions are written in a logic-programming style, by combining primitive update operations over a general KB, with the usual logic programming connectives and some additional connectives e.g. to express sequence of actions. While TR is a natural choice to deal with transactions, it remains the question whether TR can be used to express complex events, but also to deal simultaneously with the detection of complex events and the execution of transactions. In this paper we show that the former is possible while the latter is not. For that, we start by illustrating how TR can express complex events, and in particular, how SNOOP event expressions can be translated in the logic. Afterwards, we show why TR fails to deal with the two issues together, and to solve the intended problem propose Transaction Logic with Events, its syntax, model theory and executional semantics. The achieved solution is a non-monotonic extension of TR, which guarantees that every complex event detected in a transaction is necessarily responded.

Towards an ASP-Based Architecture for Autonomous UAVs in Dynamic Environments
Marcello Balduccini, William Regli and Duc Nguyen.

Traditional AI reasoning techniques have been used successfully in many domains, including logistics, scheduling and game playing. This paper is part of a project aimed at investigating how such techniques can be extended to coordinate teams of unmanned aerial vehicles (UAVs) in dynamic environments. Specifically

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press

challenging are real-world environments where UAVs and other network-enabled devices must communicate to coordinate—and communication actions are neither reliable nor free. Such network-centric environments are common in military, public safety and commercial applications, yet most research (even multi-agent planning) usually takes communications among distributed agents as a given. We address this challenge by developing an agent architecture and reasoning algorithms based on Answer Set Programming (ASP). ASP has been chosen for this task because it enables high flexibility of representation, both of knowledge and of reasoning tasks. Although ASP has been used successfully in a number of applications, and ASP-based architectures have been studied for about a decade, to the best of our knowledge this is the first practical application of a complete ASP-based agent architecture. It is also the first practical application of ASP involving a combination of centralized reasoning, decentralized reasoning, execution monitoring, and reasoning about network communications. This work has been empirically validated using a distributed network-centric software evaluation testbed and the results provide guidance to designers in how to understand and control intelligent systems that operate in these environments.

Adaptive MCMC-Based Inference in Probabilistic Logic Programs
Arun Nampally and C. R. Ramakrishnan.

Probabilistic Logic Programming (PLP) languages enable programmers to specify systems that combine logical models with statistical knowledge. The inference problem, to determine the probability of query answers in PLP, is intractable in general. In this paper, we present a technique for approximate inference of conditional probabilities for PLP queries. It is an adaptive Markov Chain Monte Carlo (MCMC) technique, where the proposal distribution is modified as the Markov Chain is explored. In particular, the distribution is modified to increase the likelihood that a proposed sample is consistent with evidence. In our context, each sample is uniquely characterized by the outcomes of a set of random variables. Inspired by reinforcement learning, our technique propagates positive rewards to random variable/outcome pairs used in a consistent sample. The cumulative rewards of each outcome of a random variable is used to derive a new "adapted distribution" for the variable. For a query with "Markovian evaluation structure", we show that the adapted proposal distribution converges to the query's conditional probability distribution. We empirically evaluate the effectiveness of the adaptive sampling methods for queries with and without Markovian evaluation structure.

Propagation Properties of Min-closed CSPs
G. Narboni.

Min-closed constraints are numerical relationships characterised by a simple property. Yet, with finite-domain variables, min-closed systems give rise to a polynomial class of Constraint Satisfaction Problems. Propagation alone checks them for satisfiability. Solving is therefore search-free. Can this result be generalized from a discrete to a continuous (or mixed) setting? In this paper, we investigate the use of interval solvers for handling constraints with real variables. We show

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press

that the completeness result observed in the discrete case gracefully degrades into a 'close approximation' property in the continuous case. When switching from finite to infinite domains, the pruning power of propagation remains intact in the sense that it provides a box enclosure whose lower bound cannot be further updated (even by domain splitting). Applications of this analysis to scheduling, rule-based reasoning and scientific simulation are briefly mentionned.

Multi-criteria optimal planning for Energy policies in CLP
Marco Gavanelli, Michela Milano, Stefano Bragaglia, Federico Chesani, Elisa Marengo and Paolo Cagnoli.

In the policy making process a number of disparate and diverse issues such as economic development, environmental aspects, as well as the social acceptance of the policy, need to be considered. A single person might not have all the required expertises, and decision support systems featuring optimization components can help to assess policies.

Leveraging on previous work on Strategic Environmental Assessment, we developed a fully-fledged system that is able to provide optimal plans with respect to a given objective, to perform multi-objective optimization and provide sets of Pareto optimal plans, and to visually compare them. Each plan is environmentally assessed and its footprint is evaluated. The heart of the system is an application developed in a popular Constraint Logic Programming system on the Reals sort. It has been equipped with a web service module that can be queried through standard interfaces, and an intuitive graphic user interface.

Joint Tabling of Logic Program Abductions and Updates
Ari Saptawijaya and Luís Moniz Pereira.

Abductive logic programs offer a formalism to declaratively represent and reason about problems in a variety of areas: diagnosis, decision making, hypothetical reasoning, etc. On the other hand, logic program updates allow us to express knowledge changes, be they internal (or self) and external (or world) changes. Abductive logic programs and logic program updates thus naturally coexist in problems that are susceptible to hypothetical reasoning about change. Taking this as a motivation, in this paper we integrate abductive logic programs and logic program updates by jointly exploiting tabling features of logic programming. The integration is based on and benefits from the two implementation techniques we separately devised previously, viz., tabled abduction and incremental tabling for query-driven propagation of logic program updates. A prototype of the integrated system is implemented in XSB Prolog.

Customisable Handling of Java References in Prolog Programs
Sergio Castro, Kim Mens and Paulo Moura.

Integration techniques for combining programs written in distinct language paradigms facilitate the implementation of specialised modules in the best language for their task. In the case of Java-Prolog integration, a known problem is the proper representation of references to Java objects on the Prolog side. To solve it adequately,

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press

multiple dimensions should be considered, including reference representation, opacity of the representation, identity preservation, reference life span, and scope of the inter-language conversion policies. This paper presents an approach that addresses all these dimensions, generalising and building on existing representation patterns of foreign references in Prolog, and taking inspiration from similar inter-language representation techniques found in other domains. Our approach maximises portability by making few assumptions about the Prolog engine interacting with Java (e.g., embedded or executed as an external process). We validate our work by extending JPC, an open-source integration library, with features supporting our approach. Our JPC library is currently compatible with three different open source Prolog engines (SWI, YAP and XSB) by means of drivers.

Interclausal Logic Variables
Paul Tarau and Fahmida Hamid.

Unification of logic variables instantly connects present and future observations of their value, independently of their location in the data areas of the runtime system. The paper extends this property to "interclausal logic variables", an easy to implement Prolog extension that supports instant global information exchanges without dynamic database updates. We illustrate their usefulness with two of algorithms, *graph coloring* and *minimum spanning tree*. Implementations of interclausal variables as source-level transformations and as abstract machine adaptations are given. To address the need for globally visible chained transitions of logic variables we describe a DCG-based program transformation that extends the functionality of interclausal variables.

A Framework for Bottom-Up Simulation of SLD-Resolution
Stefan Brass.

This paper introduces a framework for the bottom-up simulation of SLD-resolution based on partial evaluation. The main idea is to use database facts to represent a set of SLD goals. For deductive databases it is natural to assume that the rules defining derived predicates are known at "compile time", whereas the database predicates are known only later at runtime. The framework is inspired by the author's own SLDMagic method, and a variant of Earley deduction recently introduced by Heike Stephan and the author. However, it opens a much broader perspective.

Appendix B Doctoral Consortium

The 10th edition of the ICLP Doctoral Consortium took place alongside the main conference. The 8 accepted student papers listed below can be found in the online supplement of the special edition.

Bound Founded Answer Set Programming
Rehan Abdul Aziz.

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press

Model Revision Inference for Extensions of First Order Logic
Joachim Jansen.

Logic Programming as Scripting Language for Bots in Computer Games – Research Overview
Grzegorz Jaśkiewicz.

The Impact of Disjunction on Reasoning under Existential Rules: Research Summary
Michael Morak.

CDF-Intervals: A Reliable Framework to Reason about Data with Uncertainty
Aya Saad.

Visualization of Constraint Handling Rules
Nada Sharaf.

Application of Methods for Syntax Analysis of Context-Free Languages to Query Evaluation of Logic Programs
Heike Stephan.

Reasoning with Probabilistic Logics
Riccardo Zese.

https://doi.org/10.1017/S1471068414000581 Published online by Cambridge University Press