Appendix B of the Introduction to the 31st International Conference on Logic Programming Special Issue: Abstracts of Accepted Contribution at the 11th ICLP Doctoral Consortium

submitted 23 May 2015; accepted 16 June 2015; revised 17 July 2015

Dynamic Programming on Tree Decompositions using Binary Decision Diagrams: Research Summary

Günther Charwat

TU Wien, Institute of Information Systems
Favoritenstraße 9, 1040 Wien, Austria
(*e-mail:* gcharwat@dbai.tuwien.ac.at)

Dynamic programming (DP) on tree decompositions is a well studied approach for solving hard problems efficiently. State-of-the-art implementations usually rely on tables for storing information, and algorithms specify how the tuples are manipulated during traversal of the decomposition. However, a major bottleneck of such table-based algorithms is relatively high memory consumption. The goal of the doctoral thesis herein discussed is to mitigate performance and memory shortcomings of such algorithms. The idea is to replace tables with an efficient data structure that no longer requires to enumerate intermediate results explicitly during the computation. To this end, Binary Decision Diagrams (BDDs) and related concepts are studied with respect to their applicability in this setting. Besides native support for efficient storage, from a conceptual point of view BDDs give rise to an alternative approach of how DP algorithms are specified. Instead of tuple-based manipulation operations, the algorithms are specified on a logical level, where sets of models can be conjointly updated. The goal of the thesis is to provide a general tool-set for problems that can be solved efficiently via DP on tree decompositions.

KEYWORDS: Logic, Dynamic Programming, Fixed Parameter Tractability, Binary Decision Diagram, Algorithm Design

Formal Methods for Answer Set Programming

Amelia Harrison

Department of Computer Science
The University of Texas at Austin
2317 Speedway, 2.302, Austin, Texas 78712, USA
Internal Mail Code: D9500
(*e-mail:* ameliaj@cs.utexas.edu)

Answer set programming is a logic programming paradigm that has increased in popularity over the past decade and found applications in a wide variety of fields. Even so, manuals written by the designers of answer set solvers usually described the semantics of the input languages of their systems using examples and informal comments that appeal to the users' intuition, without references to any precise semantics. We describe a precise semantics for the input language of the grounder GRINGO, which serves as the front end for several answer set solvers. The semantics represents GRINGO rules as infinitary propositional formulas. We develop methods for using this semantics to prove properties of GRINGO programs, such as verifying program correctness.

KEYWORDS: Infinitary formulas, Semantics of Aggregates, Stable Models.

4

Handling Uncertainty in Answer Set Programming

Yi Wang

School of Computing, Informatics, and Decision Systems Engineering
Arizona State University, Tempe, USA
(*e-mail:* ywang485@asu.edu)

Answer Set Programming (ASP) is a powerful declarative computing paradigm that is especially suitable for modeling commonsense reasoning problems. However, the crisp nature of the underlying semantics, the stable model semantics, makes it difficult to handle reasoning domains involving probability and inconsistency. To address this issue, we present an extension of logic programs under the stable model semantics, where rules are associated with weights. Under our semantics, probabilistic commonsense domains where inconsistency might be involved can be represented in an intuitive and elaboration tolerant way. Our semantics extends MLN and logic programming under stable model semantics. We have shown that probabilistic action domains and Pearl's probabilisitic causal models can be represented, and various existing probabilistic logic programming frameworks can be embedded in our language. Future work includes further investigating the property of this language, devising algorithms for inference and learning in our language, and exploring various possible extensions of our language.

KEYWORDS: Stable Model Semantics, Answer Set Programming, Markov Logic Network, Probabilistic Logic Programming

Higher Order Support in Logic Specification Languages for Data Mining Applications

Matthias van der Hallen

KU Leuven
Department of Computer Science
Celestijnenlaan 200a
3001 Leuven, Belgium
(*e-mail:* matthias.vanderhallen@kuleuven.be)

In this paper, we introduce our work on our doctorate with title "Higher Order Support in Logic Specification Languages for Data Mining Applications". Current logic specification languages, such as FO($\cdot$) provide an intuitive way for defining the knowledge within a problem domain.

Extended support for data representation is lacking however, and we want to introduce structured recursive types and generic types, together with a first class citizen approach to predicates. These additions correspond to higher order concepts.

We provide a background of the current techniques that might be of interest when implementing these higher order abstractions, such as lazy grounding and oracles. We sketch the eventual goal of our research, and give an overview of the current state and research questions that are being considered.

KEYWORDS: Logic Specification, Higher Order, Grounding, Lifted Reasoning

Extended Abstract : Transforming Delimited Control: Achieving Faster Effect Handlers

Amr Hany Saleh

KU Leuven
Department of Computer Science
Celestijnenlaan 200a
3001 Leuven, Belgium
(*e-mail:* ah.saleh@cs.kuleuven.be)

Schrijvers et al. (2013) have introduced support for delimited control in Prolog. It enables the definition of new high-level language features at the program level (e.g., in libraries). Effect handlers (Plotkin and Pretnar 2009) are an attractive application of delimited control. They are an elegant way to add many kinds of side-effectful operations to a language in a compositional fashion.

Problem Statement. Handlers can be combined in two different ways. The first is composing them, which is one of the main attractions of effect handlers. However, this modularity comes at the cost of considerably reduced runtime performance. The second is to develop monolithic handlers such that one handler handles all the effects a program needs. Monolithic handlers are generally faster than modular ones. However, the cost of monolithiy is losing the modularity. Therefore, how can we combine both, modularity and performance.

Our Approach. We are trying to systematically derive the monolithic definition of handlers from the modular ones. This way the programmer can write his programs in terms of the modular handlers, but the Prolog system can actually run the corresponding monolithic handler. Our main technique for the systematic derivation is static program transformation using folding/unfolding framework of Pettorossi and Proietti. We complement it with transformation rules (which we proved to be sound) that capture the semantics of delimited control.

Preliminary Results and Open Issues Our experimental evaluation indicates that merged handlers are twice as fast on average. Our main issue is automating the transformation and finding a technique to handle recursive handlers' operations.

KEYWORDS: Delimited Control, Algebraic Effect Handlers, Program Transformation, Folding / Unfolding

Advances in Analyzing Coroutines by Abstract Conjunctive Partial Deduction

Vincent Nys

KU Leuven
Department of Computer Science
Celestijnenlaan 200a
3001 Leuven, Belgium
(*e-mail:* `vincent.nys@kuleuven.be`)

The current work describes a technique for the analysis of coroutining in Logic Programs. The technique provides greater insight into the execution of Logic Programs and yields a transformation from programs with nonstandard execution rules to programs with the standard execution rule of Prolog. The transformation, in turn, allows the use of analysis and optimization techniques developed for the standard execution rule.

A technique known as Compiling Control, or CC for short, was used to study these issues 30 years ago, but it lacked the tools to formalize a complete procedure. Abstract Conjunctive Partial Deduction, introduced by Leuschel, provides an appropriate setting to redefine Compiling Control for simple examples. For more elaborate examples, we extend Leuschel's framework with a new operator.

Preliminary experiments with the new operator show that a wide range of programs can be analyzed. We have performed a complete manual analysis and transformation for five examples of increasing complexity, as well as a complete analysis for a sixth example. We have also developed an automated program analyzer, which performs the analysis phase with a small amount of user interaction. The analyzer is capable of duplicating the manual analysis for all the examples. We conjecture that the extension to Abstract Conjunctive Partial Deduction allows for the analysis of any standard logic program with an instantiation-based selection rule.

The technique outlined here requires no information which is not known to the author of a Logic Program. It can be largely automated, allowing the user to write declarative programs which can be further optimized with techniques for standard Logic Programs.

KEYWORDS: Coroutines, Compiling Control, Abstract Conjunctive Partial Deduction

8

Opinions and Beliefs as constraint system operators

Salim PERCHY and Frank VALENCIA

Comète LIX - École Polytechnique de Paris, France
(*e-mail:* yamil-salim.perchy@inria.fr,frank.valencia@gmail.com)

The growing presence of digital distributed systems in social life is exemplified by many particular instances, including opinion forums, social networks, dating sites and photo sharing portals. The increased usage in the last decade of these systems brings various risks and behaviors, inherent from the social interaction therein. An epistemic aspect is singled out as a common feature shared between these systems and the behaviors carried within them. Designing, constructing and verifying formalisms to represent information that is epistemic in nature can help develop a sound theory to analyze the scenarios mentioned before and at the same time bridge the concepts involved to a logical and mathematical domain.

Regarding this, a specific concept of declarative and logic programming, that of a constraint system, deals with information represented by constraints (a constraint c could be a logical proposition partially describing a bigger system, e.g. $temperature > 20$). Constraint systems capable of incorporating the concept of spatiality such as user-spaces or message walls already exist (i.e. $[c]_i$, could read as "data/belief/constraint c belongs to agent i"). However, the movement of information between spaces is still not designed nor included in said constraint systems. Some process algebras do possess a concept of space mobility, notwithstanding, it is from an operational point of view, specifying only its behavior. Therefore it remains to mathematically define it along with all its properties.

The proposed project intends to provide constraint systems with an algebraic operator that correspond to moving information in-between spaces as to mimic the mobility of data of distributed systems such as posting opinions/lies to other spaces or publicly disclosing data (i.e. $\uparrow_i c$ reads as "extruding data/belief/constraint c from the space of agent i). Also, this extrusion operator should have a direct relationship with the spatiality operator, meaning that it should be modeled in constraint systems that also posses the concept of space (i.e. $[c \sqcup \uparrow_i d]_i = [c]_i \sqcup d$ reads "information d is extruded from the space of agent i", it can be alternatively interpreted as agent i posting an opinion d).

The authors developed a constraint system implementing the concepts of space and extrusion. The interaction between these conceptos account for mobility with no side effects, it is modeled as extrusion being the right inverse of space (i.e. $[\uparrow_i c]_i = c$ for any agent i). Additionally, given an already defined concept of space in a constraint system, the authors described different constructive ways of defining its extrusion and their mathematical properties. As a practical example, by means of a constraint system with space and extrusion, the authors gave semantic meaning

to a logic with modalities of belief B_i and utterance U_i where $B_i U_i \phi \Leftrightarrow \phi$ for any formula ϕ of the logic.