Recent Research Results in the group Declarative Languages and Artificial Intelligence

Jan Ramon, Maurice Bruynooghe and Danny De Schreye
Department Computer Science
Katholieke Universiteit Leuven

Abstract

To obtain a somewhat complete picture of the research going on in the group, we show the publications of the last 15 months, together with (sometimes) shortened) abstracts. For the work in Inductive Logic Programming which is less central to BENELOG, we have left out the abstracts.

1 Logic Programming

- M. Denecker, D. De Schreye, SLDNFA: an abductive procedure for normal abductive programs, in Journal of Logic Programming, ISSN 0743-1066, Vol. 34 ,feb 98.

  Abstract:
  We present SLDNFA, an extension of SLDNF-resolution for abductive reasoning on abductive logic programs. SLDNFA solves the floundering abduction problem: non-ground abductive atoms can be selected. SLDNFA provides also a partial solution for the floundering negation problem. Different abductive answers can be derived from an SLDNFA-refutation; these answers provide different compromises between generality and comprehensibility. Two extensions of SLDNFA are proposed which satisfy stronger completeness results. The soundness of SLDNFA and its extensions is proven. Their completeness for minimal solutions with respect to implication, cardinality and set inclusion is investigated. The formalisation of SLDNFA presented here does not rely on skolemisation of abductive atoms.

- S. Verbaeten, M. Denecker, D. De Schreye, Compositionality of normal open logic programs, in Proceedings of the 1997 International Logic Programming Symposium, Port Jefferson, New York.

- S. Verbaeten, M. Denecker, D. De Schreye, Compositionality of Normal Open Logic Programs, to appear in the Journal of Logic Programming.

Abstract:
Compositionality of programs is an important concern in knowledge representation and software development. In the context of Logic Programming, up till now, the issue has mostly been studied for *definite* programs only.

Here, we study compositionality in the context of *normal open logic programming*. This is a logic for knowledge representation in the context of uncertainty and incomplete knowledge on concepts and on problem domain, in which the compositionality issue turns up very naturally. The semantics of the logic is a generalisation (allowing non-Herbrand interpretations) of the well-founded semantics.

- S. Verbaeten, The closing operator: from partial to complete knowledge, in Proceedings of the European Conference on Artificial Intelligence, Brighton, 1998, young researcher paper.

  Abstract:
  The representation of knowledge in the logic OLP-FOL is split in two parts: writing definitions for known concepts and writing constraints, expressing partial knowledge on other concepts. An OLP-FOL theory T consists hence of two parts, $T = (T_d, T_c)$: the definition part T_d, which is a normal logic program with the extended well-founded model semantics (defined predicates are predicates occuring in the head of a clause of T_d), and the FOL (first order logic) part T_c, which is a set of FOL axioms, representing constraints on open (i.e. not defined) predicates.

  Here, we investigate the relationship between these two parts of an OLP-FOL theory in more detail.

- K. Van Belleghem, M. Denecker, D. De Schreye, A strong correspondence between description logics and open logic programming, in Proceedings of ICLP'97.

  Abstract:
  This paper formally investigates the relationship between Open Logic Programming (OLP) and Description Logics (DL). A description logic is designed to represent two different forms of knowledge. A T-Box represents definitional knowledge, i.e. definitions for a set of concepts. An A-Box represents assertional knowledge about specific domain objects. OLP is a declarative, terminological interpretation of the formalism of Abductive Logic Programming. In this interpretation, an abductive logic program is considered to consist of a T-Box providing definitions for non-abducible predicates and an A-Box providing assertional knowledge in the form of first order logic axioms.

- K. Van Belleghem, M. Denecker, D. Theseider Dupre, Ramifications in an event-based language, 1997, in Proceedings of Proceedings of the Ninth Dutch Conference on Artificial Intelligence (NAIC'97), Antwerpen.

Abstract:
In the last couple of years, several high-level languages have been proposed for modeling actions and change, following the example of Gelfond and Lifschitz's "A" language and tackling increasingly complex issues. In this paper we present a narrative-based language "ER" with a linear time structure, which is designed to deal correctly with simultaneous actions as well as with general fluent and change dependencies, in particular ramifications. We argue that a combination of state constraints, causal laws and action preconditions is necessary to correctly represent all such dependencies. In particular, we introduce causal laws stating that a change is triggered by a change in truth value of a complex fluent formula; we show such laws to be useful for a compact representation of ramifications, in particular those of simultaneous actions. A semantics for such complex causal laws is provided based on the principle of inductive definitions. Moreover ER is able to deal in a very flexible way with complete and incomplete knowledge on action occurrences, action ordering and the initial state of the world.

- M. Leuschel, D. De Schreye, Constraint partial deduction and the preservation of characteristic trees, New Generation Computing, 16 (1998), 283-342.

  Abstract:
  In this paper we exhibit the advantages of characteristic trees over purely syntactic measures as a means for abstraction in partial deduction. If characteristic trees can be preserved upon generalisation, then we obtain an almost perfect abstraction operator.

  To do this, we extend the expressivity and precision of the Lloyd and Shepherdson partial deduction framework by integrating constraints. We provide formal correctness results for the so obtained framework of constrained partial deduction. Within this framework, we introduce the mentioned abstraction operator, based on pruning constraints.

- D. De Schreye, R. Glueck, J. Joergenson, M. Leuschel, B.Martens, M. Sörenson, Conjuctive partial deduction: foundations, control, algorithms and experiments, Journal of Logic Programming, to appear.

  We first present a framework for conjunctive partial deduction, extending the Lloyd-Sheperdson framework by considering conjunctions of atoms (instead of individual atoms) for specialisation and renaming. Correctness results are given for the framework with respect to computed answer semantisc, least herbrand model semantics and finite failure semantics. We describe a basic algorithm for conjunctive partial deduction and refine it into a concrete algorithm for which we prove termination. We discuss renamings for removing redundant arguments. We discuss experiments conducted with a fully automatic implementation of the technique.

- Michael Leuschel and Danny De Schreye, Creating Specialised Integrity

Checks Through Partial Evaluation of Meta-Interpreters. The Journal of Logic Programming 36, 1998, pages 149-193.

Abstract:
Integrity constraints are useful for the specification of deductive databases, as well as for inductive and abductive logic programs. Verifying integrity constraints upon updates is a major efficiency bottleneck and specialised methods have been developed to speedup this task. They can however still incur a considerable overhead. In this paper we propose a solution to this problem by using partial evaluation to pre-compile the integrity checking for certain update patterns. The idea being, that a lot of the integrity checking can already be performed given an update pattern without knowing the actual, concrete update.

- W. Vanhoof and B. Martens, To Parse or Not To Parse, Proceedings of the Seventh International Workshop on Logic Program Synthesis and Transformation (LOPSTR '97), "Lecture Notes in Computer Science Vol. 1463, Leuven, Belgium.

  Abstract:
  In this paper, we reconsider the problem of specialising the vanilla meta interpreter through fully automatic and completely general partial deduction techniques. In particular, we study how the homeomorphic embedding relation guides specialisation of the interpreter. We focus on the so-called *parsing* problem, i.e. removing all parsing overhead from the program, and demonstrate that further refinements in the control of general partial deduction are necessary to properly deal with it.

- W. Vanhoof and B. Martens and D. De Schreye and K. De Vlaminck, Specialising The Other Way Around, Proceedings of the Joint International Conference and Symposium on Logic Programming, 1998, Manchester, United Kingdom, MIT-Press.

  In this paper, we present a program transformation based on bottom up evaluation of logic programs. We explain that using this technique, programs can be "specialised" w.r.t. a set of unit clauses instead of a query. Moreover, examples indicate that excellent specialisation can be obtained when this bottom up transformation is combined with a more traditional top down approach, resulting in conceptually cleaner techniques requiring a less complicated control than one overall approach.

- W. Vanhoof and D. De Schreye and B. Martens, A Framework for Bottom Up Specialisation of Logic Programs, 1998, Proceedings of the Joint International Symposia PLILP/ALP 1998, Lecture Notes In Computer Science, Vol. 1490.

  Abstract:
  In this paper, we develop a solid theoretical foundation for a bottom up program transformation, capable of specialising a logic program with respect to a set of unit clauses. Extending a well-known operator, originally

introduced for composing logic programs, we define a bottom up partial deduction operator and prove correctness of the transformation with respect to the S-semantics. We also show how, within this framework, a concrete control strategy can be designed.

- M. Leuschel, B. Martens and D. De Schreye, 1998, Some Achievements and Prospects in Partial Deduction, ACM Computing Surveys, Symposium on Partial Evaluation, To appear.

- K.Sagonas and M.Leuschel, 1998, Extending Partial Deduction to Tabled Execution: Some Results and Open Issues, ACM Computing Surveys, Symposium on Partial Evaluation, to appear.

- M. Leuschel, On the Power of Homeomorphic Embedding for Online Termination, Static Analysis. Proceedings of SAS'98, Pisa, Italy, September, 1998, LNCS 1503, pages 230–245, Springer-Verlag.

  Abstract:
  Recently well-quasi orders in general, and homeomorphic embedding in particular, have gained popularity to ensure the termination of program analysis, specialisation and transformation techniques. In this paper we investigate and clarify for the first time, both intuitively and formally, the advantages of such an approach over one using well-founded orders. Notably we show that the homeomorphic embedding relation is strictly more powerful than a large class of involved well-founded approaches. We, however, also illustrate that the homeomorphic embedding relation suffers from several inadequacies which are unsatisfactory in contexts, such as logic or functional & logic programming, where logical variables arise. We therefore also present new, extended homeomorphic embedding relations to remedy this problem.

- Michael Leuschel, A Formal Comparison of Well-founded and Well-quasi Orders for Online Termination, Logic Program Synthesis and Transformation. Pre-Proceedings of LOPSTR'98, Manchester, UK, 1998.

- M. Leuschel, Advanced techniques for logic program specialisation, in Artificial Intelligence Communications, Vol. 10.

- Michael Leuschel, Bern Martens, Danny De Schreye: Controlling Generalization amd Polyvariance in Partial Deduction of Normal Logic Programs. TOPLAS 20(1): 208-258.

  Abstract:
  Given a program and some input data, partial deduction computes a specialized program handling any remaining input more efficiently. However, controlling the process well is a rather difficult problem. In this article, we elaborate global control for partial deduction: for which atoms, among possibly infinitely many, should specialized relations be produced, meanwhile guaranteeing correctness as well as termination? Our work is based

on two ingredients. First, we use the concept of a characteristic tree, encapsulating specialization behavior rather than syntactic structure, to guide generalization and polyvariance, and we show how this can be done in a correct and elegant way. Second, we structure combinations of atoms and associated characteristic trees in global trees registering "causal" relationships among such pairs.

- M. Leuschel, B. Martens and K. Sagonas, Preserving Termination of Tabled Logic Programs While Unfolding, Proceedings of the International Workshop on Logic Program Synthesis and Transformation (LOPSTR'97), Leuven, Belgium.

  Abstract:
  We provide a first investigation of the specialisation and transformation of tabled logic programs through unfolding. We show that - surprisingly - unfolding, even determinate, can worsen the termination behaviour in the context of tabling. We therefore establish two criteria which ensure that such mishaps are avoided. We also briefly discuss the influence of some other transformation techniques on the termination and efficiency of tabled logic programs.

- S. Decorte, D. De Schreye, M. Leuschel, B.Martens and K. Sagonas, Termination Analysis for Tabled Logic Programming, Proceedings of the International Workshop on Logic Program Synthesis and Transformation, 1997, Leuven, Belgium.

  Abstract:
  This work has been motivated by an initial study on how to adapt advanced program specialisation techniques, originally developed for standard logic programming, to the context of tabled logic programming. In a companion paper, we describe how left-propagation of bindings in a program executed under SLG-resolution using a fixed left-to-right selection rule (as SLG is usually implemented in practical systems such as XSB) can seriously endanger the termination characteristics of that program.

- S. Decorte, D. De Schreye, Demand-driven and constraint-based automatic termination analysis for logic programs, in Proceedings of the 14th International Conference on Logic Programming, ICLP97, 78-92, 1997.

  Abstract:
  Current norm-based automatic termination analysis techniques for logic programs can be split up into different components: inference of mode or type information, derivation of models, generation of well-founded orders and verification of the termination conditions themselves. Although providing high precision results, these techniques suffer from an efficiency point of view as several of these analyses are often performed through abstract interpretation.

  In this paper we present a new termination analysis which integrates the various components and produces a set of constraints that, when solvable,

identifies successful termination proofs. The proposed method is both efficient and precise. The use of constraint sets enables the propagation of information over all different phases while the need for multiple analyses is considerably reduced.

- S. Decorte, D. De Schreye, Termination analysis: some practical properties of the norm and level mapping space, in Proceedings of the Joint International Conference and Symposium on Logic Programming, JICSLP98, 1998, 235-249.

  Abstract:
  The notions of norm and level mapping play an important role in the context of termination analysis. The goal of this paper to provide some structure on the norm and level mapping space, to analyse that structure and to obtain some new insights in termination analysis.

- R. Prasad, K. Sagonas, T. Swift, D. Warren, J. Freire, XSB: A system for efficiently computing WFS, in Proceedings of Proceedings of the Fourth International Conference on Logic Programming and Non-Monotonic Reasoning, Dagstuhl Castle, Germany, series: Lecture Notes of Artificial Intelligence, Vol. 1265, 1997.

- G. Janssens and K. Sagonas, "On the Use of Tabling for Abstract Interpretation: An experiment with Abstract Equation Systems (Extended Abstract)", in proceed ings of 1st Workshop on Tabulation in Parsing and Deduction (TAPD'98), April 2-3, 1998, Paris, pp. 118-126

  Abstract:
  Abstract interpretation is a widely used method for the static analysis of programs. This paper reports on our recent experiments with an alternative approach to implementing program analyses based on abstract interpretation. Instead of using a special-purpose system for abstract interpretation (such as `PLAI`, `GAIA`, or `AMAI`), we use a logic programming system which supports tabling (namely XSB). To obtain the results of the analysis, we directly transform the program to be analysed into an tabled logic program such that its execution by XSB coincides with the abstract interpretation of the original program.

- M. Bruynooghe, G. Janssens, A. Kagedal, Live-structure analysis for logic programming languages with declarations 01/01/1997 in Proceedings of the Fourteenth International Conference on Logic Programming (ICLP'97), Leuven, Belgium.

  Abstract:
  The advent of logic languages with type and mode declarations makes it worthwhile to analyse whether the live-structure analysis of Mulkers et al. can be elaborated in an efficient method for reclaiming memory. This paper develops a backward use analysis and makes several proposals which have potential for reducing the cost of the alias analysis.

- M. Leuschel, Program Specialisation and Abstract Interpretation Reconciled, Proceedings of the Joint International Conference and Symposium on Logic Programming JICSLP'98, MIT Press, Manchester, UK, 1998, pages 220–234.

  Abstract:
  We clarify the relationship between abstract interpretation and program specialisation in the context of logic programming. We present a generic top-down abstract specialisation framework, along with a generic correctness result, into which a lot of the existing specialisation techniques can be cast. The framework also shows how these techniques can be further improved by moving to more refined abstract domains. It, however, also highlights inherent limitations shared by all these approaches. In order to overcome them, and to fully unify program specialisation with abstract interpretation, we also develop a generic combined bottom-up/top-down framework, which allows specialisation and analysis outside the reach of existing techniques.

- M. Bruynooghe, M. Leuschel, K. Sagonas, A polyvariant binding-time analysis for off-line partial deduction 01/01/1998 in Proceedings of the European Symposium on Programming (ESOP'98), Part of ETAPS'98, Lecture Notes of Computer Sciences, Vol. 1381, Lisboa, Portugal.

  Abstract:
  We study the notion of binding-time analysis for logic programs. We formalise the unfolding aspect of an on-line partial deduction system as a Prolog program. Using abstract interpretation, we collect information about the run-time behaviour of the program. We use this information to make the control decisions about the unfolding at analysis time and to turn the on-line system into an off-line system. We report on some initial experiments.

- D. de Waal, M. Denecker, M. Bruynooghe, M. Thielscher, The generation of pre-interpretations for detecting unsolvable planning problems 01/01/1997 in Proceedings of the Workshop on Model-Based Automated Reasoning (15th International Joint Conference on Artificial Intelligence)

  Abstract:
  A general, problematic aspect of deductive planning, namely, detecting unsolvable planning problems is investigated. The work is based on a STRIPS-like approach to reasoning about actions and change using a logic programming paradigm. A model-based analysis is proposed as a promising and successful way to detect such problems.

- M. Bruynooghe, H. Vandecasteele, D.A. de Waal, M. Denecker, Detecting Unsolvable Queries for Definite Logic Programs in Principles of Declarative Programming, Proc. PLILP/ALP'98, Pisa, sept. 1998, Eds. Catuscia Palamidessi, Hugh Glaser, Karl Meinke, LNCS 1490, Springer Verlag, pp 118– 133. ISBN 3-540-65012-1

Abstract:
In logic programming, almost no work has been done so far on proving that certain queries cannot succeed. Work in this direction could be useful for queries which seem to be non-terminating. Such queries are not exceptional, e.g. in planning problems. The paper develops some methods, based on abduction, goal-directedness, tabulation, and constraint techniques, for proving failure of queries for definite logic programs. It also reports some experiments with various tools.

- Henk Vandecasteele, Bart Demoen en Joachim Van Der Auwera: The Use of Mercury for the Implementation of a Finite Domain Solver. In: Nova Science special volume on Parallelism and Implementation Technology for (Constraint) Logic Programming Languages, to appear.

  Mercury is a recent phenomenon in the field of logic programming: it is faster than other logic language implementations (e.g. Prolog) and better suited for the development of large applications because of its compile-time error-detection capabilities. The concepts and technology used in the implementation of Mercury are rather new and not yet evaluated thoroughly outside its implementors group. This paper reports on an evaluation of Mercury, by describing the porting from the medium-sized constraint solving tool ROPE written originally in Prolog, to Mercury.

2 Inductive Logic Programming

- H. Blockeel and L. De Raedt, Top-down Induction of First Order Logical Decision Trees, 1998, Artificial Intelligence, Vol. 101, Nr. 1-2, pages 285-297.

- S. Džeroski, S. Schulze-Kremer, K. R. Heidtke, K. Siems, D. Wettschereck, H. Blockeel, Diterpene Structure Elucidation from 13C NMR Spectra with Inductive Logic Programming, 1998, Applied Artificial Intelligence, Vol 12., Nr. 5, pages 363-384.

- H. Blockeel and L. De Raedt, IsIdd: An Interactive System for Inductive Database Design, 1998, Applied Artificial Intelligence, Vol. 12, Nr. 5, pages 385-421.

- H. Blockeel, L. De Raedt, N. Jacobs and B. Demoen, Scaling Up Inductive Logic Programming by Learning From Interpretations, 1998, Data Mining and Knowledge Discovery, To appear.

- L. De Raedt, Logical settings for concept-learning, in Artificial Intelligence, ISSN 0004-3702, Vol. 95.

- L. De Raedt and L. Dehaspe. Clausal discovery, Machine Learning , 26:99-146, 1997.

- L. De Raedt, H. Blockeel, L. Dehaspe, and W. Van Laer. Three companions for first order data mining, In N. Lavrac and S. Dzeroski, eds, Inductive Logic Programming for Knowledge Discovery in Databases, Lecture Notes in Artificial Intelligence, to appear, Springer-Verlag, 1998.
- Kurt Driessens, Nico Jacobs, Nathalie Cossement, Patrick Monsieurs, Luc De Raedt, Inductive Verification and Validation of the KULRoT RoboCup Team, Proceedings of the second RoboCup workshop, 1998.
- Nico Jacobs, Kurt Driessens, Luc De Raedt, Using ILP-systems for Verification and Validation of Multi-Agent Systems, Proceedings of the 8th International Conference on Inductive Logic Programming, 1998.
- S. Roberts, N. Jacobs, W. Van Laer, S. Muggleton, and J. Broughton, A Comparision of ILP and Propositional Systems on Propositional Traffic Data, Proceedings of the 8th International Conference on Inductive Logic Programming, 1998.
- Nico Jacobs, Kurt Driessens, Luc De Raedt, Inductive Verification and Validation of Multi-Agent Systems, Verification and Validation Workshop, Knowledge Representation '98.
- S. Dzeroski, N. Jacobs, M. Molina, C. Moure, ILP experiments in detecting traffic problems, 1998, in Proceedings of the 10th European Conference on Machine Learning, series: Lectures Notes in Artificial Intelligence.
- W. Van Laer, L. De Raedt, S. Dzeroski, On multi-class problems and discretization in inductive logic programming, 1997, in Proceedings of the 10th International Symposium on Methodologies for Intelligent Systems (ISMIS97), Lecture Notes in Artificial Intelligence, Vol. 1325
- H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In Proceedings of the 15th International Conference on Machine Learning, pages 55-63, 1998. http://www.cs.kuleuven.ac.be/ ml/PS/ML98-56.ps.
- J. Ramon and M. Bruynooghe. A framework for defining distances between first-order logic objects. In Proceedings of the 8th International Conference on Inductive Logic Programming, Lecture Notes in Artificial Intelligence, pages 271-280. Springer-Verlag, 1998.
- J. Ramon, M. Bruynooghe, and W. Van Laer. Distance measures between atoms. In Proceedings of the CompulogNet Area Meeting on 'Computational Logic and Machine Learning', page 35-41, 1998.
- L. Dehaspe, H. Toivonen, and R. D. King. Finding frequent substructures in chemical compounds, In Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD-98).

- S. Džeroski, L. De Raedt and H. Blockeel, Relational Reinforcement Learning, In Proceedings of the 15th International Conference on Machine Learning, 1998.
- H. Blockeel, L. De Raedt, Top-down induction of logical decision trees, in Proceedings of Proceedings of the Ninth Dutch Conference on Artificial Intelligence (NAIC'97), Antwerpen
- L. Dehaspe, L. De Raedt, Wanted: association rules over multiple relations, in Proceedings of Proceedings of the Ninth Dutch Conference on Artificial Intelligence (NAIC'97), Antwerpen.
- L. De Raedt, L. Dehaspe, Learning from satisfiability, in Proceedings of the Ninth Dutch Conference on Artificial Intelligence (NAIC'97), Antwerpen.