# Correct Reasoning about Logic Programs

**Jael Kriener**

**School of Computing, University of Kent, CT2 7NF, UK**

## Abstract

In this PhD project, we present an approach to the problem of determinacy inference in logic programs with cut, which treats cut uniformly and contextually. The overall aim is to develop a theoretical analysis, abstract it to a suitable domain and prove both the concrete analysis and the abstraction correct in a formal theorem prover (Coq). A crucial advantage of this approach, besides the guarantee of correctness, is the possibility of automatically extracting an implementation of the analysis.

**1998 ACM Subject Classification** D.1.6 Logic Programming, D.2.4 Software/Program Verification, F.3.1Specifying and Verifying and Reasoning about Programs, F.3.2 Semantics of Programming Languages

**Keywords and phrases** Prolog, cut, determinacy inference, abstract interpretation, denotational semantics, automated theorem proving, Coq

**Digital Object Identifier** 10.4230/LIPIcs.ICLP.2011.280

## 1 Introduction and problem description

The focus of my work lies on provably correct static analysis of logic programs, in particular on determinacy inference for logic programs containing *cut*, and on verified implementations thereof.

For reasons which hardly need spelling out here, the question of whether a goal is deterministic or not is central in logic programming. In practice, logic programmers often use in-build control mechanisms, like the *cut* in Prolog, to ensure determinacy of certain goals. Yet, todate methods for inferring determinacy conditions on goals have not addressed this close connection between the *cut*s in a program and the determinacy of its goals. One of the problems I address in my PhD work is to apply well-known techniques in program analysis (abstract interpretation) to develop and prove correct a method for determinacy inference that takes account of this.

Furthermore, I would like to build on the work of Cachera, Pichardie and others ([3], [4], etc.), who observe that "[i]n spite of the nice mathematical theory of program analysis and the solid algorithmic techniques available one problematic issue persists, *viz.*, the *gap* between the analysis that is proved correct on paper and the analyser that actually runs on the machine" [3]. Thanks to advances in theorem proving, and in particular to the Coq system, this gap can be bridged. Coq provides a framework in which the development of a well-defined determinacy inferrence, its correctness proof with respect to an underlying program semantics and its implementation can be part of one integrated process. I would like to use Coq to prove correct the determinacy analysis discussed above and to obtain a verified implementation of it.

Technical Communications of the 27th International Conference on Logic Programming (ICLP'11).
Editors: John P. Gallagher, Michael Gelfond; pp. 280–283
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 2 Background and overview of the existing literature

### Logic Programming

[11], [13], [15] are standard works on logic programming and as such have been most helpful so far and will in all likelihood continue to be so.

### Determinacy Inference in Logic Programs

[5] present a method for inferring determinacy information from a program by adding constraints to the clauses of a predicate which allow the inference of mutual exclusion conditions between these clauses rather than determinacy conditions for a whole predicate. [14] presents a method for determinacy analysis, based on a partial evaluation technique for full Prolog which detects whether there are none, one or more than one ways a goal can succeed. [10] present a top-down framework for abstract interpretation of Prolog which is based on sequences of substitutions and can be instantiated to derive an analysis equivalent to that of [14]. Finally, my own supervisor has worked on the problem of determinacy inference before (see [6], [7], [12]). These works form the starting point of my own research.

### Coq and Automated Abstract Interpretaion

I have spent less time aquiring background in on Coq and Automated Abstract Interpretation. [1] is a standard reference for Coq and should provide a good starting point. David Cachera, David Pichardie and others have published on abstract interpretation in Coq (see [3], [4]).

## 3 Goal of the research

My goal is to develop a determinacy analysis for logic programs including *cut*, that is as tight as possible, to prove it correct using a framework for automated theorem proving and to obtain a verified implementation of the same.

## 4 Current status of the research

During the first six months of my PhD work I have focused on the following:

- Acquiring the necessary background and techniques in program analysis, in particular in abstract interpretation.
- Researching previous work on determinacy in logic programs and on formal semantics for logic programs including *cut*.
- Applying the understanding of these two areas to develop and prove correct a determinacy inference for logic programs including *cut*.

This work has led to a paper submitted to ICLP 2011 (see Section 5 below). There are some issues arising from this work, that will need to be addressed, before I can move to the next stage in my overall project (see Section 6 below).

## 5 Preliminary results accomplished

In collaboration with my supervisor Dr Andy King, I have written a paper presenting and manually proving correct a method for inferring determinacy conditions for Prolog with *cut*

which has been accepted to ICLP 2011 and for publication in a special issue of the journal "Theory and Practice of Logic Programming" [9, 8].

In the process of implementing this method, I found one difficulty in computational elimination of existential quantifiers in constraint systems. In addressing this problem I have developed, in collaboration with Jörg Brauer and my supervisor, a method for reducing existential quantifier elimination to incremental SAT, a paper on which has just been accepted for publication in the conference Computed Aided Verification (see [2]).

## 6 Open issues and expected achievements

In the short term, I am addressing a further difficulty arising from the manual implementation of the determinacy inference, other than the issue of existential quantifier elimination mentioned in the last section, namely how to efficiently compute a mutual exclusion condition for two sets of constraints. I am reasonably confident that this problem can be addressed in a similar fashion by reformulating it as an incremental SAT problem.

In the long term, to achieve the goals outlined in the previous sections, I plan to reformulate and implement the determinacy inferrence mentioned above and its underlying semantics for Prolog with *cut* in the Coq system and mechanize my manual correctness prove. For this, I will need to gain considerable background in automated theorem proving.

## References


1 Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. Springer Verlag, 2004.

2 J. Brauer, A. King, and J. Kriener. Existential Quantification as Incremental SAT. In Ganesh Gopalakrishnan and Shaz Qadeer, editors, *Twenty-third International Conference on Computer Aided Verification*, Lecture Notes in Computer Science. Springer-Verlag, July 2011. To appear.

3 David Cachera, Thomas P. Jensen, David Pichardie, and Vlad Rusu. Extracting a Data Flow Analyser in Constructive Logic. *Theor. Comput. Sci.*, 342(1):56–78, 2005.

4 David Cachera and David Pichardie. A Certified Denotational Abstract Interpreter. In Matt Kaufmann and Lawrence C. Paulson, editors, *ITP*, volume 6172 of *Lecture Notes in Computer Science*, pages 9–24. Springer, 2010.

5 S. Dawson, C. R. Ramakrishnan, I. V. Ramakrishnan, and R. C. Sekar. Extracting Determinacy in Logic Programs. In *Proceedings of the Tenth International Conference on Logic Programming*, pages 424–438. MIT Press, 1993.

6 S. Genaim and A. King. Inferring Non-Suspension Conditions for Logic Programs with Dynamic Scheduling. *ACM Transactions on Computational Logic*, 9(3), November 2008.

7 A. King, L. Lu, and S. Genaim. Detecting Determinacy in Prolog Programs. In *Proceedings of the Twenty-second International Conference on Logic Programming*, volume 4079 of *Lecture Notes in Computer Science*, pages 132–147. Springer, 2006.

8 Jael Kriener and Andy King. Appendix for RedAlert: Determinacy Inference for Prolog. Technical Report 1-11, School of Computing, University of Kent, CT2 7NF, UK, 2011. Available from: `http://www.cs.kent.ac.uk/pubs/2011/3107`.

9 Jael Kriener and Andy King. RedAlert: Determinacy Inference for Prolog. *Theory and Practice of Logic Programming*, July 2011. To appear.

10 B. Le Charlier, S. Rossi, and P. Van Hentenryck. An Abstract Interpretation Framework which Accurately Handles Prolog Search-Rule and the Cut. In *Symposium on Logic Programming*, pages 157–171. MIT Press, 1994.

**11** John W. Lloyd. *Foundations of Logic Programming, 2nd Edition*. Springer, 1987.

**12** L. Lu and A. King. Determinacy Inference for Logic Programs. In Shmuel Sagiv, editor, *Fourteenth European Symposium on Programming*, volume 3444 of *Lecture Notes in Computer Science*, pages 108–123. Springer, 2005.

**13** Richard A. O'Keefe. *The Craft of Prolog*. MIT Press, Cambridge, MA, USA, 1990.

**14** D. Sahlin. Determinacy Analysis for Full Prolog. In *Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, pages 23–30. ACM, 1991.

**15** Leon Sterling and Ehud Y. Shapiro. *The Art of Prolog - Advanced Programming Techniques, 2nd Ed*. MIT Press, 1994.