# Some Achievements and Prospects in Partial Deduction

MICHAEL LEUSCHEL, BERN MARTENS and DANNY DE SCHREYE
Katholieke Universiteit Leuven

## 1. PARTIAL DEDUCTION

In the context of logic programming the full input to a program $P$ consists of a goal $G$ and evaluation corresponds to constructing a complete SLDNF-tree for $P \cup \{G\}$. Partial evaluation in such a setting is usually referred to as *partial deduction*. Its general idea is to construct a finite set of atoms $\mathcal{A}$ and associated finite, but possibly *incomplete* SLDNF-trees. The derivation steps in these SLDNF-trees comprise the computation performed beforehand by the *partial deducer*. Clauses of the specialised program are composed of root and leaves of these trees: one specialised clause per branch.

## 2. CONTROL OF AUTOMATIC PARTIAL DEDUCTION

Most work on partial deduction is situated within the *on-line* tradition: All the control decisions are performed fully automatically *during* the actual specialisation phase. The work presented here is no exception (see e.g. [Jørgensen and Leuschel 1996] for an off-line approach).

In partial deduction one usually distinguishes two levels of control:

- the *global control*, determining the set $\mathcal{A}$, i.e. deciding *which* atoms are partially deduced, and
- the *local control*, guiding construction of the finite SLDNF-trees for each individual atom in $\mathcal{A}$ and thus determining *what* the definitions for the partially deduced atoms look like.

The control of partial deduction should ensure termination, adequate specialisa-

Michael Leuschel is Post-doctoral Fellow of the Fund for Scientific Research — Flanders Belgium. Bern Martens is supported by the Belgian GOA "Non-Standard Applications of Abstract Interpretation". Danny De Schreye is Senior Research Associate of the Fund for Scientific Research — Flanders Belgium.

Authors' addresses: M. Leuschel, B. Martens and D. De Schreye, Departement Computerwetenschappen, K.U.Leuven, Celestijnenlaan 200A, B-3001 Heverlee, Belgium. E-mail : {michael,bern,dannyd}@cs.kuleuven.ac.be

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works, requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept, ACM Inc., 1515 Broadway, New York, NY 10036 USA, fax +1 (212) 869-0481, or `permissions@acm.org`.

tion and correctness (i.e. ensuring the independence and closedness conditions of [Lloyd and Shepherdson 1991]).

### 2.1 Local Control

Two major issues arise in the local control.

- Avoiding search space explosion as well as work duplication:
  *Determinacy* [Gallagher and Bruynooghe 1991] — only (except once) select atoms that match a single clause head — has proven to prevent these kinds of problems. The strategy can be refined with a so-called "look-ahead" to detect failure at a deeper level.
- Termination:
  Imposing some (essentially) *well-founded* [Bruynooghe et al. 1992; Martens and De Schreye 1996] or *well-quasi* relations on selected atoms ensures termination in a non ad hoc fashion, taking the structural properties of the program into account. Recently, well-quasi relations such as the homeomorphic embedding relation, extended to improve the treatment of variables [Leuschel et al. 1998], have gained popularity.

A suitable combination of determinacy and ordering leads to a practically viable local control.

### 2.2 Global Control

The global control is usually embodied in a so-called *abstraction operator*, which, given a set $\mathcal{A}$, constructs a new set $\mathcal{A}'$, generalising atoms in $\mathcal{A}$, such that every atom in $\mathcal{A}$ is an instance of an atom in $\mathcal{A}'$. A good abstraction operator achieves global termination with minimal specialisation loss.

The main problem in that context is the control of polyvariance: when is it sensible to generate different specialised versions for a particular predicate and when should abstraction be performed instead. A good abstraction operator should not just be based on the syntactic structure of the atoms under consideration. Indeed, two atoms can behave identically in the context of one program and very differently in the context of another one. A refined approach is based on *characteristic trees* [Gallagher and Bruynooghe 1991] or *trace terms* [Gallagher and Lafave 1996], which capture the *specialisation behaviour* of atoms, and m-trees [Martens and Gallagher 1995], which register their relationship.

To avoid specialisation losses, it is important that characteristic trees be preserved upon abstraction. Prior techniques were incapable to achieve this. The method in [Leuschel et al. 1998] relies on *imposing* characteristic trees on the generalised atoms as well as adapting the homeomorphic embedding relation for characteristic trees. In this way, it is possible to spot sequences of growing characteristic trees and perform generalisation in order to avert the danger of non-termination. The resulting depth bound free technique is at the heart of the ECCE system [Leuschel 1996].

Finally, the control of polyvariance problem occurs in different guises in many areas of program analysis, manipulation and optimisation. In many cases, current techniques impose depth bounds to ensure termination. This is for instance the case for neighbourhoods in supercompilation. We conjecture that our techniques can be fruitfully adapted for specialisation and analysis in other (declarative) programming

paradigms (see also [Glück and Sørensen 1994] for the relation of partial deduction to driving of functional programs).

## 3. CONJUNCTIVE PARTIAL DEDUCTION

Classical, "atom based", partial deduction is incapable of performing certain useful unfold/fold transformations, like tupling or deforestation. Conjunctive partial deduction has been designed with the aim of overcoming these limitations. The essential aspect lies in the joint treatment of *entire conjunctions* of atoms, connected through shared variables, at the global level (complemented with some renaming to deliver program clauses) [Leuschel et al. 1996; Glück et al. 1996; Leuschel and Sørensen 1996; Jørgensen et al. 1996].

Apart from this aspect, the conventional control notions described above also apply in a conjunctive setting. In essence, it was possible to consolidate partial deduction and unfold/fold program transformation, incorporating most of the power of the latter while keeping the automatic control and efficiency of the former.

Deforestation and tupling like transformations are useful even in the absence of partial input. This warrants the integration of our techniques into a compiler, as their systematic use seems highly beneficial and will allow users to more easily decompose and combine procedures and programs without having to worry about the ensuing inefficiencies of intermediate data structures.

## 4. INCORPORATING ABSTRACT INTERPRETATION

Ongoing work attends to integrating *abstract interpretation* with partial deduction. It has been shown [Leuschel and De Schreye 1996] that both program manipulation methods have limitations on their own which can be overcome by combining them. A fine-grained algorithm interleaves top-down (conjunctive) partial deduction steps with bottom-up abstract interpretation steps and is able to obtain specialisation and analysis beyond the reach of either method alone.

## 5. PROSPECTS FOR FURTHER PROGRESS

For some applications, the developed control techniques remain too restrictive. In particular, they do not always deal satisfactorily with fluctuating structure (arising e.g. for certain meta-interpretation tasks). The use of characteristic trees remedies this problem to some extent, but not totally.

Next, a good way to fully prevent slowdowns in practice remains a pressing open question. Also, in some cases, the present global control regimes will allow too many versions, unwarranted by the actual specialisation. Both problems may be succesfully tackled through a more detailed efficiency and cost estimation.

Although conjunctive partial deduction can achieve deforestation and tupling, doing so automatically, always avoiding slowdowns, is still a difficult problem.

Further efforts are also required to extend partial deduction for tabled (see this volume) and constraint logic programming (see also [Leuschel and De Schreye 1998; Lafave and Gallagher 1997]).

Finally, a full synthesis of partial deduction and abstract interpretation, conceptual as well as technical, will probably prove beneficial to both fields.

## REFERENCES


Bruynooghe, M., De Schreye, D., and Martens, B. 1992. A general criterion for avoiding infinite unfolding during partial deduction. *New Gen. Comput. 11,* 1, 47–79.

Gallagher, J. and Bruynooghe, M. 1991. The derivation of an algorithm for program specialisation. *New Gen. Comput. 9,* 3 & 4, 305–333.

Gallagher, J. and Lafave, L. 1996. Regular approximations of computation paths in logic and functional languages. In *Proceedings of the 1996 Dagstuhl Seminar on Partial Evaluation*, O. Danvy, R. Glück, and P. Thiemann, Eds. LNCS 1110. Springer-Verlag, 115–136.

Glück, R. and Sørensen, M. H. 1994. Partial deduction and driving are equivalent. In *Programming Language Implementation and Logic Programming. Proceedings, Proceedings of PLILP'94*, M. Hermenegildo and J. Penjam, Eds. LNCS 844. Springer-Verlag, 165–181.

Jørgensen, J. and Leuschel, M. 1996. Efficiently generating efficient generating extensions in Prolog. In *Proceedings of the 1996 Dagstuhl Seminar on Partial Evaluation*, O. Danvy, R. Glück, and P. Thiemann, Eds. LNCS 1110. Springer-Verlag, 238–262.

Jørgensen, J., Leuschel, M., and Martens, B. 1996. Conjunctive partial deduction in practice. In *Proceedings LOPSTR'96*, J. Gallagher, Ed. LNCS 1207. Springer-Verlag, 59–82.

Glück, R., Jørgensen, J., Martens, B., and Sørensen, M. H. 1996. Controlling conjunctive partial deduction of definite logic programs. In *Proceedings PLILP'96*, H. Kuchen and S. Swierstra, Eds. LNCS 1140. Springer-Verlag, 152–166.

Lafave, L. and Gallagher, J. 1997. Constraint-based partial evaluation of rewriting-based functional logic programs. In *Proceedings LOPSTR'97*, N. Fuchs, Ed. LNCS. Springer-Verlag. To Appear.

Leuschel, M. 1996. The ecce partial deduction system and the dppd library of benchmarks. Obtainable via `http://www.cs.kuleuven.ac.be/~lpai`.

Leuschel, M. and De Schreye, D. 1998. Constrained partial deduction and the preservation of characteristic trees. *New Gen. Comput.* . To Appear.

Leuschel, M. and De Schreye, D. 1996. Logic program specialisation: How to be more specific. In *Proceedings PLILP'96*, H. Kuchen and S. Swierstra, Eds. LNCS 1140. Springer-Verlag, 137–151.

Leuschel, M., De Schreye, D., and de Waal, A. 1996. A conceptual embedding of folding into partial deduction: Towards a maximal integration. In *Proceedings JICSLP'96*, M. Maher, Ed. MIT Press, Germany, 319–332.

Leuschel, M., Martens, B., and De Schreye, D. 1998. Controlling generalisation and polyvariance in partial deduction of normal logic programs. *ACM Trans. Program. Lang. Syst.* To Appear.

Leuschel, M. and Sørensen, M. H. 1996. Redundant argument filtering of logic programs. In *Proceedings LOPSTR'96*, J. Gallagher, Ed. LNCS 1207. Springer-Verlag, 83–103.

Lloyd, J. W. and Shepherdson, J. C. 1991. Partial evaluation in logic programming. *J. Logic Program 11,* 3& 4, 217–242.

Martens, B. and De Schreye, D. 1996. Automatic finite unfolding using well-founded measures. *J. Logic Program. 28,* 2, 89–146.

Martens, B. and Gallagher, J. 1995. Ensuring global termination of partial deduction while allowing flexible polyvariance. In *Proceedings ICLP'95*, L. Sterling, Ed. MIT Press, 597–613.