

Curriculum Vitae: Oege de Moor

1 ACADEMIC HISTORY

1983-1987 MSc in Computing Science (distinction), November 1987. Rijksuniversiteit Utrecht, The Netherlands.

1989-1992 DPhil, March 1992. University of Oxford. British Petroleum Research Student.

2 EMPLOYMENT HISTORY

1983-1984 Programmer, Nenijsa Computing, the Netherlands. Wordprocessor for Arabic and Hebrew.

1988-1989 Research assistant, Rijksuniversiteit Utrecht, the Netherlands. Project: "Specification and Transformation of Programs". Fellow of the British Council at Oxford.

1991-1993 Junior research fellow, St. John's College, University of Oxford.

1993-1994 Fujitsu endowed chair of information engineering (visitor), University of Tokyo, Japan.

1994-1994 Visiting fellow, Chalmers University, Gothenburg, Sweden. Host: Prof. John Hughes.

1994-2002 University Lecturer, University of Oxford. Tutorial fellow in computation, Magdalen College.

1997 Visiting researcher, Microsoft Research, Redmond, USA

1999 Visiting researcher, Microsoft Research, Cambridge, UK

2002-present Full professor of computer science, University of Oxford.

3 TEACHING

Lecture courses Introduction to Functional Programming, Principles of Programming Languages [new course], Computational Geometry [new course], Imperative Programming, Procedural Programming, Advanced Functional Programming [new course], Databases [new course], Object-Oriented Design [new course].

Practicals for all of the above, including demonstrating when a new practical is introduced, development of new supporting software (for instance a spreadsheet system implemented from scratch to illustrate the principles of incremental computation).

Classes (problem solving sessions) in Computational Geometry, Databases and Object-oriented design.

Tutorial teaching at Oxford, permanent members of staff who do not hold a chair have teaching duties in one or more colleges, where they discuss technical exercises with groups of 1-3 students, typically for 7 contact hours per week. Subjects taught: Functional Programming, Algorithm Design, Formal Logic, Models of Computation, Digital Design, Formal Program Design, Object-Oriented Programming, Computational Complexity, Computer Architecture, Distributed Systems, Concurrency, Compilers, Programming Languages.

4 ADMINISTRATION

Including research management and examining duties.

Programming tools group founder; organisation of weekly research meetings (for discussing work in progress), seminars, visits by overseas colleagues. [’98-present]

Departmental seminar series at the Computing Laboratory, jointly with Richard Brent. [2000]

Algebra of programming group founder, jointly with Richard Bird. [’93-’97]

Seminars on programming organiser; weekly research talks by outside speakers. [’93-’97]

Cakes meetings organiser. [’94-’97]

Schools liaison officer [’97-present]

Editor Technical Monograph series, Computing Laboratory. [’98-’00]

Windows lab Arranged donation by Microsoft. [’98, ’99]

Project coordinator Engineering and Computing Science, Computation. [’99-’01]

New curriculum member of working party. [’01-’02]

Academic appointments committee [’01-’02]

Subfaculty of CS secretary [’01-’03]

Finals examiner [’01-’04]

Chairman of finals examiners [’06]

PhD Examinations: 14, of which 8 overseas.

5 RESEARCH SUPERVISION

Undergraduate and MSc projects: 25 projects. 17 of these gained a distinction or prize. In 2005, *all* major prizes in computer science exams at Oxford (MSc, undergraduate) were won by members of De Moor's research team.

PhD supervision:

Clare Martin, '88-'91 Preordered Categories and Predicate Transformers. Co-supervisor with Tony Hoare.

Sharon Curtis, '92-'96 A relational approach to optimization problems. Co-supervisor with Richard Bird.

Michael Zhu Ning, '94-'98 Relational combinatorial optimization.

Richard McPhee, '94-'99 Towards a relational programming language.

Ganesh Sittampalam, '98-'01 Higher-order matching for program transformation.

David Lacey, '99-'02 Imperative program transformation by rewriting.

Yorck Hünke, '00-'04 Dependent types for a lazy functional programming language.

Stephen Drape, '01-'04 Hiding source by obfuscation.

Ran Ettinger, '01-present Correctness of refactoring transformations. (expected completion: December 2005)

Ian Lynagh, '02-present Soft types for template Haskell. (expected completion: December 2005)

Damien Sereni, '03-present Termination analysis. (expected completion: June 2006)

Duncan Coutts, '03-present A partial evaluator for Haskell.

Mathieu Verbaere, '04-present Scripting refactoring transformations.

Pavel Avgustinov, '05-present Optimising tracematches in AspectJ.

Elnar Hajiyev, '05-present Incremental compilation of aspect-oriented programs.

Neil Ongkingco, '05-present Information hiding in aspect-oriented programming.

Julian Tibble, '05-present Semantics of aspects.

6 INVITED LECTURES (selection)

1989 1st International summer school on Constructive Algorithmics, Ameland, The Netherlands.

1992 2nd International summer school on Constructive Algorithmics, Ameland, The Netherlands. Voted best lecturer by audience.

1993 IFIP State-of-the-art seminar on Formal Program Development, Rio de Janeiro, Brazil.

1995 International symposium on Programming Languages: Implementations, Logics and Programs (PLILP '95), Arnhem, The Netherlands.

1995 Relational Methods in Computer Science (RELMICS), Parati, Brazil.

1998 Workshop on Program Transformation, Shonan village, Japan.

1998 Dagstuhl seminar on Programs: Meaning, Complexity and Improvements, Germany.

1998 International summerschool on Advanced Functional Programming, Braga, Portugal.

1999 Workshop on Partial Evaluation, Waseda University, Tokyo, Japan.

2000 Workshop on Program Transformation, Hakone, Japan.

2000 International conference on Algebraic Methods and Software Technology (AMAST), Iowa, USA.

2000 International workshop on Attribute Grammars and their Applications (WAGA), Ponte de Lima, Portugal.

2001 British Computer Society, special interest group on programming. London.

2001 Irish Workshop on Formal Methods (IWFMM), Dublin.

2003 Emerging Technologies: Can optimization technology meet their demands? Dagstuhl, Germany.

2003 Perspectives workshop on Software Optimization, Dagstuhl, Germany.

2004 Workshop on Aspect-Oriented Programming (WAOP), Trento, Italy.

2005 International summer school on Program Analysis and Transformation (PAT), Copenhagen, Denmark.

2005 International conference on Generative Programming and Component Engineering (GPCE), Tallinn, Estonia.

2005 British Council UK-Israel workshop on semantics and its applications, Tel Aviv.

2006 Distinguished Speaker, PL group, IBM Watson Research Centre, USA.

2006 International conference on Mathematics of Program Construction, Kuressaare, Estonia.

2006 Aspects for Legacy Applications, Germany.

2006 Formal aspects of testing and runtime verification (FATES/RV 2006), Seattle, USA.

2006 IEEE AstreNet Aspect Analysis, Benevento, Italy.

2007 International summer school on Generative and Transformational Techniques in Software Engineering (GTTSE)

2007 ACM SIGPLAN 2007 Workshop on Partial Evaluation and Program Manipulation (PEPM 2007), Nice, France

7 GRANTS

For fellowship awards *etc.*, see ‘Employment history’.

Sep ’96 - Sep ’99 EPSRC travel grant “Generic solutions to optimization problems. 15K

Oct ’98 - Sep ’01 Microsoft collaborative project “A meta-language for IP”, 300K [equipment, 3 research students and 1 postdoc. To support joint work with Dr. Charles Simonyi, distinguished engineer and chief architect at Microsoft]

Oct ’01 - Sep ’03 Unrestricted gift from Microsoft to support Oege de Moor’s research team, 145K [1 postdoc and research student, plus travel and equipment]

Oct ’02 - Oct ’04 EPSRC ROPA: Language support for self-optimising libraries, 109K.

Oct ’02 - Sep ’05 studentship for Ian Lynagh, Microsoft Research, 45K.

Jan ’03 - Dec ’03 IBM Eclipse innovation award, 15K.

Sep ’03 - Aug ’04 EPSRC visiting fellowship for Laurie Hendren, 18K.

Jan ’04 - Dec ’04 IBM Eclipse innovation award, 15K.

Oct ’04 - Sep ’07 Microsoft donation to support research in code refactoring, 82K.

Oct ’05 - Sep ’08 EPSRC grant “The Design and Implementation of Aspect-oriented Programming Languages”, 373K (four research students).

Jan ’06 - Dec ’08 EPSRC platform grant “Centre for metacomputation”, 432K (jointly with Samson Abramsky, Luke Ong and Tom Melham).

Sep ’06 - Aug ’06 IBM Eclipse innovation award, 15K

8 ADDITIONAL RESEARCH ACTIVITIES

Recent program committees: ADI (2006); AOSD (2004,2006,2007,2008); Asian-AOSD (2005,2006); APLAS (2006); CC (2003, 2006); ESOP (1999,2004); EWAS (2006); FOAL (2005,2006); GTSSE (2005); MPC (2000,2003); PEPM (2006); PLDI (2002); POPL (2007); VMIL (2007); RTA (2003).

Member of the editorial board of *Higher-Order and Symbolic Computation*; Program chair of AOSD 2007. Member of EPSRC review college (2000-present).

Professional membership: IFIP Working Group 2.1 on Programming Languages and Calculi; IFIP Working Group 2.11 on Generative Programming; Association for Computing machinery, SIGPLAN.

9 RESEARCH PROGRAMME

Oege de Moor is a Professor of Computer Science at the University of Oxford. His research revolves around the theme of *meta-programming*: programs that manipulate other programs. Meta-programming is at the heart of many current trends in industry, including autonomous computing and automated tools for error detection. A unified approach to the subject, aimed at bringing it to the mainstream of programming practice, is the topic of his present research.

9.1 Theory In his early career, De Moor studied the theory of program transformation, in particular applied to the characterisation of algorithmic paradigms, via the use of category theory [1, 4, 5, 14, 15, 17–20, 24, 26, 52]. In the course of this work, he invented an algebraic view of predicate transformer semantics, which has pointed the way to extensions of the refinement calculus [3, 6, 16]. He also originated the idea of generic programs that are parameterised by type constructors [7, 9]: an example of such a program is a unification algorithm, parameterised by the type of terms. This sparked a lot of further work by Hinze, Jeuring and others under the banner of ‘polytypic’ programming. While it is natural to write programs that take other programs or types as parameters in the functional programming community, that is not the case in logic programming. Motivated by that observation, Richard McPhee and De Moor invented a simple form of higher-order logic programming, which is quite expressive but does not require higher-order unification [21]. With Geraint Jones and Richard S. Bird, De Moor demonstrated that there exists no complexity preserving transformation from lazy functional programs to pure strict functional programs [8].

9.2 Mechanised transformation From 1998, De Moor turned to the mechanised application of program transformations, motivated by a sabbatical he spent with Charles Simonyi at Microsoft Research [23]. Jointly with Ganesh Sittampalam, he invented a number of matching algorithms, which enabled the effective mechanisation of so-called fusion transformations [11, 22, 30, 34]. These transformations join two consecutive passes over a data structure into one. He also invented a new way of composing attribute grammars, for the purpose of easy experimentation with new language features in compiler frontends [10, 25, 32].

With Conal Elliott and Sigbjorn Finne, De Moor turned these ideas for transforming functional programs into practice in the specific application domain of image composition. It has long been known that declarative descriptions of images can be appealing; but when such descriptions pertain to moving images, efficiency becomes a major concern. By applying aggressive

rewriting and code motion optimisations, the approach was shown to be feasible in practice [13, 27].

While it is relatively easy to express transformations of declarative programs as simple rewrite rules (possibly using advanced matching algorithms), it is much more difficult to give simple specifications of transformations of imperative programs. Jointly with David Lacey, De Moor invented a simple notation for expressing such transformations, based on logic programming and model checking [12, 29, 31]. He also invented (with Sittampalam) an efficient way of executing such specifications of transformations [37]. De Moor lectured on that topic at the 2005 International Summer School on Program Analysis and Transformation in Copenhagen. This work directly impacts on the *code queries* and also on *refactoring*, further discussed below.

9.3 Aspect oriented programming Aspects provide a highly disciplined form of program transformation, where instead of transforming some representation of a program, one transforms traces of a computation. Viewed in this light, there is a close link between aspect-orientation and the above work on mechanised program transformation. Indeed, this observation led De Moor and Damien Sereni to propose a static analysis for the optimisation of aspect-oriented programs [33]. These results were only demonstrated on a small Pascal-like language, but their potential was clear. Therefore, Laurie Hendren (of McGill University, Montreal) and De Moor decided to use their simultaneous sabbaticals to pursue the compilation of aspect-oriented programming languages. Hendren’s sabbatical at Oxford was partially funded through an EPSRC Visiting Fellowship.

The first step was to do careful measurements of the runtime overheads of using aspects. Until that point, it was generally believed in the aspect-oriented programming community that there are no such overheads. Using a special tool to collect dynamic metrics, the team led by De Moor and Laurie Hendren was able to refute that common belief [36].

Because of the substantial overheads that were identified, it was decided to build a new, optimising compiler for the AspectJ language. AspectJ is the leading aspect-oriented programming language, and there exists one other compiler, named *ajc*. In addition to a workbench for research on analysis and optimisation, the new compiler was also intended as a testbed for new ideas in language design. The new compiler was named the *AspectBench* compiler, or *abc* for short [58].

We have already used this workbench to investigate important extensions to the existing AspectJ language. For instance, in [38], we show how extra code in an aspect can be triggered by traces (instead of just individual events) that satisfy a given pattern. Starting from a clear declarative semantics, we derived an operational semantics and from that a highly nontrivial implementation.

Another important topic is that of information hiding. Starting with the theoretical design of Jonathan Aldrich, the *abc* team has designed and implemented a

comprehensive set of language features to control the exposure of implementation detail [43].

A third contribution to the design of aspect-oriented languages is the use of *Datalog* to describe instrumentation points. We have shown that the whole pointcut language of AspectJ can be expressed via Datalog queries [49].

Six research papers (all in a journal or at top conferences: POPL, OOPSLA, PLDI, and AOSD) have been published about the development of *abc* [38, 40, 41, 43, 44, 49]. Several more have been presented as invited papers at international conferences [39, 47, 48]. At least 12 groups worldwide have already adopted *abc* for their own use.

The development of *abc* is now supported by a grant from EPSRC that provides funding for four research students.

9.4 Refactoring As part of his interest in mechanised program transformation, De Moor has pursued the implementation of *refactoring* transformations, jointly with Ran Ettinger and Mathieu Verbaere. The first step was a new refactoring tool for Java in the Eclipse development environment, that generalised the existing *extract method* refactoring. This new tool allows the developer to extract a *backward slice* of all statements that might contribute to the value of a given expression.

The prototype generated a lot of interest, and De Moor was invited to demonstrate and discuss this research direction personally with Bill Gates, the chairman of Microsoft. Following that discussion, Microsoft Research provided funding to develop a scripting language for refactoring, and a first design is described in [45]. This scripting language directly builds on De Moor’s earlier work on the mechanisation of program transformations.

9.5 Code queries From matching algorithms and logic programming, it is but a small step to *code queries*: queries for identifying interesting points in the source of a program, typically prior to manual or mechanised transformation.

With his students Elnar Hajiyev and Mathieu Verbaere, De Moor revisited the idea that one might store the whole program in a relational database, to facilitate fast query evaluation. A major difference with such earlier proposals was however the use of *Datalog* as a query language, in conjunction with an optimising compiler from Datalog to procedural SQL. In a series of detailed experiments, it transpired that this setup allows a degree of scalability that other code query systems cannot match. Indeed, for large code bases, this implementation method is vastly superior to the use of a highly tuned Prolog system [42, 46].

The work on refactoring, aspects and code queries are thus all linked via the use of Datalog. In refactoring, Datalog queries are used to identify source locations that are candidates for change, and to check the complex side conditions of refactoring transformations. In aspect-oriented programming, Datalog queries are used

to identify events that may be intercepted by aspects. In code queries, Datalog is used to express coding conventions, or just queries for program understanding.

10 LIST OF PUBLICATIONS

BOOKS:

- [1] Richard S. Bird and Oege de Moor. *Algebra of Programming*. International Series in Computer Science. Prentice Hall, 1996.
- [2] Jeremy Gibbons and Oege de Moor, editors. *The Fun of Programming*. Cornerstones in Computing. Palgrave, 2003.

JOURNAL PAPERS (REFEREED):

- [3] Oege de Moor. Inductive data types for predicate transformers. *Information Processing Letters*, 43(3):113–118, 1992.
- [4] Richard S. Bird and Oege de Moor. List partitions. *Formal Aspects of Computing*, 5(1):61–78, 1993.
- [5] Oege de Moor. Categories, relations and dynamic programming. *Mathematical Structures in Computing Science*, 4:33–69, 1994.
- [6] Paul H. B. Gardiner, Clare E. Martin, and Oege de Moor. An algebraic construction of predicate transformers. *Science of Computer Programming*, 22(1-2):21–44, 1994.
- [7] Richard S. Bird, Paul F. Hoogendijk, and Oege de Moor. Generic programming with relations and functors. *Journal of Functional Programming*, 6(1), 1996.
- [8] Richard S. Bird, Geraint Jones, and Oege de Moor. More haste, less speed: lazy versus eager evaluation. *Journal of functional programming*, 7(5):541–547, 1997.
- [9] Paul F. Hoogendijk and Oege de Moor. Container types categorically. *Journal of Functional Programming*, 10(2):191–225, 2000.
- [10] Oege de Moor, Kevin Backhouse, and S. Doaitse Swierstra. First-class attribute grammars. *Informatica*, 24:329–341, 2000.
- [11] Oege de Moor and Ganesh Sittampalam. Higher-order pattern matching for program transformation. *Theoretical Computer Science*, 269(1-2):135–162, 2001.
- [12] Oege de Moor, David Lacey, and Eric Van Wyk. Universal regular path queries. *Higher-Order and Symbolic Computation*, 16(1-2):15–35, 2003.
- [13] Conal Elliott, Sigbjorn Finne, and Oege de Moor. Compiling embedded languages. *Journal of Functional Programming*, 13(3):455–481, 2003.

CONFERENCE PAPERS (REFEREED):

- [14] Richard S. Bird and Oege de Moor. From dynamic programming to greedy algorithms. In B. Möller, H. Partsch, and S. Schuman, editors, *Formal Program Development*, volume 755 of *Lecture Notes in*

Computer Science, pages 43–61. Springer Verlag, 1993.

- [15] Richard S. Bird and Oege de Moor. Solving optimisation problems with catamorphisms. In R. S. Bird, C. C. Morgan, and J. C. P. Woodcock, editors, *Mathematics of Program Construction*, volume 669 of *Lecture Notes in Computer Science*, pages 45–66. Springer Verlag, 1993.
- [16] Paul Gardiner, Clare Martin, and Oege de Moor. An algebraic construction of predicate transformers. In C. C. Morgan and J. C. P. Woodcock, editors, *Mathematics of Program Construction*, volume 669 of *Lecture Notes in Computer Science*, pages 100–121, 1993.
- [17] S. Doaitse Swierstra and Oege de Moor. Virtual data structures. In B. Möller, H. Partsch, and S. Schumann, editors, *Formal Program Development*, volume 755 of *Lecture Notes in Computer Science*, pages 355–371, 1993.
- [18] Richard S. Bird and Oege De Moor. Relational program derivation and context-free language recognition. In A. W. Roscoe, editor, *A Classical Mind: Essays dedicated to C.A.R. Hoare*, pages 17–35. Prentice Hall International, 1994.
- [19] Richard S. Bird and Oege de Moor. The algebra of programming. In *Proceedings of the 1994 International Summer School, Marktoberdorf, NATO ASI Series F*. Springer, 1994.
- [20] Oege de Moor. A generic program for sequential decision processes. In M. Hermenegildo and D. S. Swierstra, editors, *Programming Languages: Implementations, Logics, and Programs*, volume 982 of *Lecture Notes in Computer Science*, pages 1–23, 1995.
- [21] Richard McPhee and Oege de Moor. Compositional logic programming. In M. Chakravarty, Y. Guo, and T. Ida, editors, *Proceedings of the JICSLP '96 post-conference workshop: Multi-paradigm logic programming*, Report 96-28, pages 115–127. Technische Universitat Berlin, 1996.
- [22] Oege de Moor and Ganesh Sittampalam. Generic program transformation. In *Advanced Functional Programming*, Lecture Notes in Computer Science, pages 116–149. Springer Verlag, 1998.
- [23] Brian Dickens, Paul Kwiatkowski, Oege de Moor, David Richter, and Charles Simonyi. Transformation in intentional programming. In Jeffrey Poulin, editor, *5th International Conference on Software Re-use*, pages 114–123. IEEE Press, 1998.
- [24] Oege de Moor. Dynamic programming as a software component. In N. Mastorakis, editor, *CSCC, July 4-8, Athens*, 1999.
- [25] Oege de Moor, Simon Peyton-Jones, and Eric Van Wyk. Aspect-oriented compilers. In *First International Symposium on Generative and Component-based Software Engineering*, Lecture Notes in Computer Science 1799, pages 121–133, 1999.
- [26] Oege de Moor and Jeremy Gibbons. Pointwise relational programming. In *Algebraic Methodology*

- and *Software Technology*, volume 1816 of *Springer Lecture Notes in Computer Science*, pages 371–390, 2000.
- [27] Conal Elliott, Sigbjorn Finne, and Oege de Moor. Compiling embedded languages. In Walid Taha, editor, *SAIG*, volume 1924 of *Lecture Notes in Computer Science*, pages 9–27, 2000.
- [28] Bernard A. Sufrin and Oege de Moor. Modeless structure editing. In A. W. Roscoe and J. C. P. Woodcock, editors, *Millennial perspectives in computing*. Palgrave, 2000.
- [29] David Lacey and Oege de Moor. Imperative program transformation by rewriting. In R. Wilhelm, editor, *Proceedings of the 10th International Conference on Compiler Construction*, volume 2027 of *Lecture Notes in Computer Science*, pages 52–68. Springer Verlag, 2001.
- [30] Ganesh Sittampalam and Oege de Moor. Higher-order pattern matching for automatically applying fusion transformations. In *Second symposium on Programs as Data Objects*, volume 2053 of *Lecture Notes in Computer Science*, pages 218–237, 2001.
- [31] Stephen J. Drape, Oege de Moor, and Ganesh Sittampalam. Transforming the .NET intermediate language using path logic programming. In *Principles and practice of declarative programming (PPDP)*, pages 133–144, 2002.
- [32] Eric Van Wyk, Paul Kwiatkowski, Oege de Moor, and Kevin Backhouse. Forwarding in attribute grammars for modular language design. In *Compiler Construction '02*, Volume 2304 of *Lecture Notes in Computer Science*, pages 128–142, 2002.
- [33] Damien Sereni and Oege de Moor. Static analysis of aspects. In *Proceedings of the 2nd International Conference on Aspect-Oriented Software Development (AOSD)*, pages 30–39, 2003.
- [34] Ganesh Sittampalam and Oege de Moor. Mechanising fusion. In Jeremy Gibbons and Oege de Moor, editors, *The Fun of Programming*, pages 79–103. Palgrave, 2003.
- [35] Brian Barry and Oege de Moor. eTX and the eclipse phenomenon. *Electr. Notes Theor. Comput. Sci.*, 107:1–5, 2004.
- [36] Bruno Dufour, Christopher Goard, Laurie Hendren, Oege de Moor, Ganesh Sittampalam, and Clark Verbrugge. Measuring the dynamic behaviour of AspectJ programs. In *19th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2004)*, pages 150–169, 2004.
- [37] Ganesh Sittampalam, Oege de Moor, and Ken Friis Larsen. Incremental execution of transformation specifications. In *Principles of Programming Languages (POPL)*, pages 26–38, 2004.
- [38] Chris Allan, Pavel Avgustinov, Aske Simon Christensen, Laurie Hendren, Sascha Kuzins, Jennifer Lhoták, Oege de Moor, Damien Sereni, Ganesh Sittampalam, and Julian Tibble. Adding trace-matching with free variables to AspectJ. In *20th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA 2005)*, 2004.
- [39] Chris Allan, Pavel Avgustinov, Aske Simon Christensen, Laurie Hendren, Sascha Kuzins, Jennifer Lhoták, Ondřej Lhoták, Oege de Moor, Damien Sereni, Ganesh Sittampalam, and Julian Tibble. The AspectBench Compiler for AspectJ. In *Generative Programming and Component Engineering*, Volume 3676 of *Lecture Notes in Computer Science*, pages 10–16, 2005.
- [40] Pavel Avgustinov, Aske Simon Christensen, Laurie Hendren, Sascha Kuzins, Jennifer Lhoták, Ondřej Lhoták, Oege de Moor, Damien Sereni, Ganesh Sittampalam, and Julian Tibble. *abc*: An extensible AspectJ compiler. In *Aspect-Oriented Software Development (AOSD)*, pages 87–98. ACM Press, 2005.
- [41] Pavel Avgustinov, Aske Simon Christensen, Laurie Hendren, Sascha Kuzins, Jennifer Lhoták, Ondřej Lhoták, Oege de Moor, Damien Sereni, Ganesh Sittampalam, and Julian Tibble. Optimising AspectJ. In *Programming Language Design and Implementation (PLDI)*, pages 117–128. ACM Press, 2005.
- [42] Elnar Hajiyev, Mathieu Verbaere, Oege de Moor, and Kris de Volder. *CodeQuest*: Querying source code with Datalog. In *Object-Oriented Programming Languages and Systems (OOPSLA) Companion*, 2005.
- [43] Neil Ongkingco, Pavel Avgustinov, Laurie Hendren, Oege de Moor, Julian Tibble, and Ganesh Sittampalam. Adding open modules to AspectJ. In: Awais Rashid and Hidehiko Masuhara (editors), *Proceedings of the International Conference on Aspect-Oriented Software Development (AOSD)*, ACM Press 2006.
- [44] Pavel Avgustinov, Aske Simon Christensen, Laurie Hendren, Sascha Kuzins, Jennifer Lhoták, Ondřej Lhoták, Oege de Moor, Damien Sereni, Ganesh Sittampalam, and Julian Tibble. *abc*: An extensible AspectJ compiler. *Transactions on Aspect-Oriented Software Development (AOSD)*, Volume 1, Number 1, pp. 293–334. Springer Verlag, 2006. (Journal version of [40])
- [45] Mathieu Verbaere, Oege de Moor, and Ran Etinger. Scripting refactoring transformations with JunGL. In: Dieter Rombach and Mary Lou Soffa (editors), *International Conference on Software Engineering (ICSE)*, pp. 127–181. ACM Press, 2006.
- [46] Elnar Hajiyev, Mathieu Verbaere, and Oege de Moor. *CodeQuest*: Scalable source code queries with Datalog. In: David Thomas (editor), *European Conference on Object-Oriented Programming (ECOOP '06)*, *Lecture Notes in Computer Science* 4067, pp. 2–27. Springer, 2006.
- [47] Pavel Avgustinov, Eric Bodden, Elnar Hajiyev, Oege de Moor, Neil Ongkingco, Damien Sereni, Ganesh Sittampalam and Julian Tibble. Aspects and data refinement. In: Tarmo Uustalu (editor),

- 8th International Conference on Mathematics of Program Construction* (MPC '06), Lecture Notes in Computer Science 4014, pp. 4–9. Springer, 2006.
- [48] Pavel Avgustinov, Eric Bodden, Elnar Hajiyev, Laurie Hendren, Ondrej Lhotak, Oege de Moor, Neil Ongkingco, Damien Sereni, Ganesh Sittampalam, Julian Tibble, Mathieu Verbaere. Aspects for Trace Monitoring. In: Klaus Havelund, Manuel Nunez, Grigore Rosu and Burkhart Wolff (editors) *Formal Approaches to Testing and Runtime Verification* (FATES/RV '06), Lecture Notes in Computer Science 4262, pp. 20–39. Springer, 2006.
- [49] Pavel Avgustinov, Elnar Hajiyev, Neil Ongkingco, Oege de Moor, Damien Sereni, Julian Tibble, Mathieu Verbaere. Semantics of Static Pointcuts in AspectJ. : In: Matthias Felleisen (editor) ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, 2007.
- [58] The abc team. The AspectBench Compiler. Home page with downloads, FAQ, documentation, support mailing lists, and bug database. <http://aspectbench.org>.

TECHNICAL REPORTS:

- [50] Oege de Moor. The elimination of variables from functional programs. Technical report RUU-CS-87-24, Department of Computer Science, Utrecht University, The Netherlands, 1987.
- [51] Oege de Moor. The role of *divide* in divide-and-rule algorithms. Technical report RUU-CS-87-26, Department of Computer Science, Utrecht University, The Netherlands, 1987.
- [52] Oege de Moor. Categories, relations and dynamic programming. D.Phil. thesis. Technical Monograph PRG-98, Computing Laboratory, Oxford, UK, 1992.
- [53] Albert Visser, Oege de Moor, and Michiel Walsteijn. Proving the first and second incompleteness theorem using concatenation. Technical report RUU-CS-86-15, Department of Computer Science, Utrecht University, The Netherlands, 1986.

SOFTWARE:

- [54] Oege de Moor. A linear-time (independent of line width) paragraph formatter in C++. Available from <http://progttools.comlab.ox.ac.uk/members/oege/publications/documents/scp99.tar.gz>, 1999.
- [55] Ganesh Sittampalam and Oege de Moor. The program transformation system mag. Available from <http://web.comlab.ox.ac.uk/oucl/research/pdt/progttools/mag/index.html>, 1999-2002.
- [56] Oege de Moor. The image manipulation language PAN. Available from <http://conal.net/pan>, 2000.
- [57] Oege de Moor and Meurig Sage. An extensible structure editor. Available from <http://progttools.comlab.ox.ac.uk/members/oege/publications/documents/sedistr.zip>, 2001.