

# Curriculum vitae

Torben Amtoft

email: tamtoft HAT cis DOT ksu DOT edu

URL: [people.cis.ksu.edu/~tamtoft](http://people.cis.ksu.edu/~tamtoft)

November 28, 2008

## Contents

<b>Academic Record</b>	<b>2</b>
<b>Current Research Interests</b>	<b>2</b>
<b>Awards and Honors</b>	<b>2</b>
<b>Professional Employments</b>	<b>2</b>
<b>Teaching</b>	<b>3</b>
<b>Publications</b>	<b>6</b>
<b>Software Development</b>	<b>11</b>
<b>Invited Talks and Research Presentations</b>	<b>11</b>
<b>Professional Contributions</b>	<b>13</b>
<b>Professional Development</b>	<b>15</b>
<b>Personal Record</b>	<b>16</b>

## Academic Record

**1993** June 21th: Awarded the Ph.D. degree in Computer Science at DAIMI, University of Aarhus. Advisor: Brian H. Mayoh.

**1989** August 18th: Graduated as “cand.scient” (corresponding to an M.Sc.) in Computer Science at DIKU, Copenhagen. Advisor: Neil D. Jones.

**1985** Completed “bifag” (corresponding to a Bachelor’s degree) in Computer Science and in Mathematics, at the University of Copenhagen.

## Current Research Interests

Program analysis, type systems (with effects and/or polymorphism), logical approaches. Language-based security, information flow & dependency analysis, program slicing. Calculi for concurrency (ambients), pointer languages.

## Awards and Honors

**January 2006** was co-awarded a grant of \$450K over 3 years from AFOSR (Air Force Office of Scientific Research) for the proposal *An Integrated Specification and Verification Environment for Component-based Architectures of Large-scale Distributed Systems*. (John Hatcliff is the PI, Anindya Banerjee is also co-PI.)

**August 2004** received the Best Paper Award for the 2004 edition of Static Analysis Symposium (SAS), together with Anindya Banerjee, for our paper *Information Flow Analysis in Logical Form*. We were invited to submit an extended version to a special issue of *Science of Computer Programming*.

## Professional Employments

**2008-present** Associate professor (tenured) at Kansas State University.

**2002-2008** Assistant professor at Kansas State University.

**2002** Research associate at Heriot-Watt University, employed by the DART project (funded by the European Union) and part of the ULTRA group.

**1999-2002** Research associate at Boston University, working on the Church Project.

**1992-1998** Research assistant/research associate at DAIMI, University of Aarhus, working with Hanne Riis Nielson and Flemming Nielson on the projects

- DART (Design, Analysis and Reasoning about Tools), funded by the Danish Research Councils; and
- LOMAPS (Logical and Operational Methods in the Analysis of Programs and Systems), funded by the European Union

and latest also with Olivier Danvy on BRICS.

**1989-1992** Ph.D. scholarship from University of Aarhus, including duties as teaching assistant on the first-year course “Dat1”.

**1985-1989** Part time teaching assistant at the University of Copenhagen, on the third-year course “Dat2”. As later in Aarhus, my tasks included weekly class exercises, grading assignments and projects (some of which proposed by me), and meeting with the lecturer for planning and evaluation.

## Teaching

### Undergraduate Courses

**Fall 2002 and onwards, Kansas State University** Each semester (except for Spring 2006 and Spring 2007), I have taught the undergraduate course CIS 301 on *Logical Foundations of Programming*, since Spring 2003 using the text book *Language, Proof and Logic* by Barwise & Etchemendy which comes with a software package. I have developed supplemental lecture notes on program verification and on induction.

**Spring 2001, Boston University** Together with Assaf Kfoury, I taught an undergraduate course CS525 on compiler design theory, adapting material developed by Laurie Hendren and Michael Schwartzbach.

**Fall 1996, University of Aarhus** I was responsible for the one-month part concerning denotational semantics in an undergraduate course on semantics taught by Flemming Nielson.

### Graduate Courses & Seminars

**Spring 2004 and onwards, Kansas State University** Each Spring, I teach a course (CIS 761) on *Database Management Systems*.

**Fall 2008, Kansas State University** I teach a graduate course (CIS 775) on *Analysis of Algorithms*.

**Spring 2005, Kansas State University** Together with Anindya Banerjee, I taught a graduate course (CIS 905) on *Program Analysis*.

**Fall 2003 & Fall 2004, Kansas State University** Together with Anindya Banerjee, I taught a graduate course (CIS 890) on *Language Based Security*.

**Spring 2003, Kansas State University** I taught a graduate course (CIS 905) on *Program Analysis*, with emphasis on Abstract Interpretation. In September 2003 (and repeated in February 2004), I (assisted by Dave Schmidt) gave a Ph.D. prelim exam (for one student) based on the course.

**Fall 1999 & Spring 2000 & Fall 2000, Boston University** Together with Assaf Kfoury and Santiago Pericas, I organized almost weekly seminars on “Programming the Web” (later: “Programming the Internet”) and presented several research papers.

**Fall 1993, University of Aarhus** I gave several lectures, presenting and discussing a selection of research papers, in a graduate course on program analysis taught by Flemming Nielson.

**Spring 1993, University of Aarhus** Flemming Nielson and I jointly gave a graduate course on functional languages.

## Students

**Committees for Ph.D.** William Deng (defended June 2007, major professors: John Hatcliff & Robby), Edwin Rodriguez (still to defend, major professor: John Hatcliff), Waleed Aljandal (still to defend, major professor: Bill Hsu), Oksana I Tkachuk (still to defend, major professor: Matt Dwyer).

**Major professor for Master's.** Santosh Bejjamshety (current MSE project), Sandhya Bathini (upcoming MSE project), Nayan Ancha (upcoming MSE project), Abhilash Manne (upcoming MSE project), Vamsi Mummaneni (upcoming MSE project), Nisha Stephen (upcoming Master report).

**Committees for Master's.** Reshma Sawant (carried out MSE project September 2007–March 2008, major professor: Dan Andresen), Rahul Deshmukh (carried out MSE project October–December 2007, major professor: Dan Andresen), Raja Sanjeev Nakka (defended Master's report August 2007, major professor: Dan Andresen); Sundeep Gopisetty (carried out MSE project January–August 2007, major professor: Bill Hankley); Praveen Turpu Seema Rachamalla (defended Master's report August 2007, major professor: Neilsen); Mrudula Talloju (defended Master's report July 2007, major professor: Bill Hankley); Shalaka Borker (defended Master's report October 2006, major professor: Bill Hankley); Javier Ramos Rodriguez (carried out MSE project Spring/Summer 2006, major professor: Bill Hankley); Divya Dalapathi (defended Master's report July 2006, major professor: Dan Andresen); Sruthi Bandhakavi (defended Master's thesis in November 2005, major professor: Anindya Banerjee); Ganeshan Jayaraman (defended Master's report in September 2005, major professor: John Hatcliff); Edwin Rodriguez (defended Master's thesis in April 2005, major professor: John Hatcliff) Nagarjuna Nagulapati (defended Master's report in May 2004, major professor: Dan Andresen).

**Project Supervision.** Naga Sowjanya Karumuri (implementation project January–May 2008).

**Research Advising.** In the Spring of 2008, I hosted Ye Zhang, a Ph.D. student from Technical University of Denmark (advisor: Flemming Nielson) who works on constraint solvers for program analysis (as reported in [10]), and how to optimize them.

In the first part of 2007, Anindya Banerjee and I met regularly with Jonathan Hoag, working on implementing our algorithms for conditional information flow.

Since early 2005, I have participated in regular group meetings with Edwin Rodriguez, a Ph.D. student of John Hatcliff, who is interested in language-based security.

In the Spring of 2006, Anindya Banerjee and I met weekly with Scott Harmon, looking into papers on concurrency and resource control.

From the summer of 2004 until the end of 2005, Anindya Banerjee and I met at least once (but often more) a week with his research assistant Sruthi Bandhakavi; a main focus of our work has been on developing a logic for information flow in pointer languages.

During 2004 and 2005, I participated in regular group meetings with Venkatesh Ranganath, a Ph.D. student of John Hatcliff, who is interested in the theory of (concurrent) slicing.

In the late Spring of 2003, and also in December 2003, I met frequently with Oksana Tkachuk, a graduate student supervised by Matthew Dwyer, assisting her in setting up a correctness proof for an alias & side effect analysis of Java proposed by her.

Other informal advisees include: Tamara Rezk, a Ph.D. student from INRIA Sophia-Antipolis who visited Kansas State for 6 weeks in the Spring of 2004 and who is interested in language-based security; Henning Korsholm Rohde, a Ph.D. student at BRICS, Aarhus, Denmark (advisors: Olivier Danvy & Andrzej Filinski) who is interested in string matching from a programming language perspective; Alexander Landraitis, a student of Robert Muller at Boston College who is implementing our type system for register allocation.

## Publications

### [Monographs]

- [1] Torben Amtoft, Flemming Nielson, and Hanne Riis Nielson. *Type and Effect Systems: Behaviours for Concurrency*. Imperial College Press,

1999. (A preliminary version appeared as the technical report PB-529, DAIMI, University of Aarhus.)

**[Journal papers]**

- [2] Torben Amtoft. Flow-sensitive type systems and the ambient calculus. *Higher-Order and Symbolic Computation*, 21(4):411–442, 2008.
- [3] Torben Amtoft. Slicing for modern program structures: a theory for eliminating irrelevant loops. *Information Processing Letters*, 106:45–51, 2008.
- [4] Venkatesh Prasad Ranganath, Torben Amtoft, Anindya Banerjee, John Hatcliff, and Matthew B. Dwyer. A new foundation for control dependence and slicing for modern program structures. *ACM TOPLAS*, 29(5), 2007. A special issue with extended versions of selected papers from the 14th European Symposium on Programming (ESOP’05).
- [5] Torben Amtoft and Anindya Banerjee. A logic for information flow analysis with an application to forward slicing of simple imperative programs. *Science of Computer Programming*, 64(1):3–28, 2007.
- [6] Torben Amtoft, Assaf J. Kfoury, and Santiago M. Pericas-Geertsen. Orderly communication in the ambient calculus. *Computer Languages, Systems & Structures*, 28:29–60, 2002 (Elsevier Science).
- [7] Torben Amtoft, Hanne Riis Nielson, and Flemming Nielson. Behaviour analysis for validating communication patterns. *Software Tools for Technology Transfer*, 2(1):13–28, 1998. (A preliminary version is available as the technical report PB-527, DAIMI, University of Aarhus.)
- [8] Torben Amtoft, Flemming Nielson, and Hanne Riis Nielson. Type and behaviour reconstruction for higher-order concurrent programs. *Journal of Functional Programming*, 7(3):321–347, May 1997.
- [9] Torben Amtoft and Jesper Larsson Träff. Partial memoization for obtaining linear time behavior of a 2DPDA. *Theoretical Computer Science*, 98(2):347–356, May 1992.

**[Conference papers]**

- [10] Ye Zhang, Torben Amtoft, and Flemming Nielson. From generic to specific: off-line optimization for a general constraint solver. In *Proceedings of Generative Programming and Component Engineering (GPCE’08)*, pages 45–53, ACM Press, October 2008. Acceptance rate: 31 %.

- [11] Torben Amtoft, John Hatcliff, Edwin Rodriguez, Robby, Jonathan Hoag, and David Greve. Specification and checking of software contracts for conditional information flow. In Proceedings of the 15th International Symposium on Formal Methods (FM'08), pages 229–245, Springer LNCS 5014, May 2008. Acceptance rate: 21.7 %. An extended version appears as Technical Report SAnToS-TR2007-5, CIS Department, Kansas State University.
- [12] Torben Amtoft and Anindya Banerjee. Verification condition generation for conditional information flow. In Proceedings of the 5th ACM Workshop on Formal Methods in Security Engineering (FMSE'07), pages 2–11, George Mason University, November 2007. Acceptance rate: 28.6 %. An extended version appears as Technical Report 2007-2, Dept. of Computing and Information Sciences, Kansas State University, August 2007.
- [13] Torben Amtoft, Sruthi Bandhakavi, and Anindya Banerjee. A logic for information flow in object-oriented programs. In Proceedings of the 33rd Annual ACM SIGPLAN - SIGACT Symposium on Principles of Programming Languages (POPL'06), pages 91–102, ACM Press, 2006. Acceptance rate: 19.8 %.
- [14] Venkatesh Ranganath, Torben Amtoft, Anindya Banerjee, Matthew B. Dwyer, and John Hatcliff. A new foundation for control-dependence and slicing for modern program structures. In *Proc. ESOP 2005* (part of *ETAPS 2005*), pages 77–93, Springer LNCS 3444, 2005. Acceptance rate: 24.6 %.
- [15] Torben Amtoft and Anindya Banerjee. Information flow analysis in logical form. In *Proc. SAS 2004*, pages 100–115, Springer LNCS 3148, 2004. Received the SAS'04 Best Paper Award. Acceptance rate: 36.5 %. An extended version appears as the technical report 2004-3, Department of Computing and Information Sciences, Kansas State University, April 2004.
- [16] Torben Amtoft and Henning Makholm and J. B. Wells. PolyA: true type polymorphism for mobile ambients. In *Proc. TCS 2004*, pages 591–604, Kluwer, 2004. An extended version appears as the technical report HW-MACS-TR-0015, School of Mathematical and Computer Sciences, Heriot-Watt University, February 2004.

- [17] Torben Amtoft and Robert Muller. Inferring annotated types for inter-procedural register allocation with constructor flattening. Proceedings of ACM SIGPLAN TLDI'03 Workshop, pages 86–97, ACM Press, January 2003. Acceptance rate: 42 %.
- [18] Torben Amtoft, Assaf J. Kfoury, and Santiago M. Pericas-Geertsen. What are polymorphically-typed ambients? In *Proc. ESOP 2001* (part of *ETAPS 2001*), pages 206–220, Springer LNCS 2028, 2001. Acceptance rate: 34 %. An extended version appears as the technical report BUCS-TR-2000-021, Boston University.
- [19] Torben Amtoft and Franklyn Turbak. Faithful translations between polyvariant flows and polymorphic types. In *Proc. ESOP 2000* (part of *ETAPS 2000*), pages 26–40, Springer LNCS 1782, 2000. Acceptance rate: 31 %.
- [20] Hanne Riis Nielson, Torben Amtoft, and Flemming Nielson. Behaviour analysis and safety conditions: a case study in CML. In *Proc. FASE'98* (part of *ETAPS'98*), pages 255–269, Springer LNCS 1382, 1998. Acceptance rate: 31 %.
- [21] Torben Amtoft. Local type reconstruction by means of symbolic fixed point iteration. In *Proc. ESOP'94*, pages 43–57, Springer LNCS 788, 1994. Acceptance rate: 28 %.
- [22] Torben Amtoft. Minimal thunkification. In *Proc. WSA '93*, pages 218–229, Springer LNCS 724, 1993. Acceptance rate: 29 %.
- [23] Torben Amtoft. Unfold/fold transformations preserving termination properties. In *Proc. PLILP'92*, pages 187–201, Springer LNCS 631, August 1992. Acceptance rate: 35 %.
- [24] Torben Amtoft. Properties of unfolding-based meta-level systems. In *Partial Evaluation and Semantics-Based Program Manipulation (PEPM'91)*, New Haven, Connecticut. Sigplan Notices, vol. 26, no. 9, pages 243–254, 1991. Acceptance rate: 41 %.
- [25] Torben Amtoft, Thomas Nikolajsen, Jesper Larsson Träff, and Neil D. Jones. Experiments with implementations of two theoretical constructions. In *Logic at Botik, USSR*, pages 119–133, Springer LNCS 363, July 1989.

**[Other reviewed papers]**

- [26] Torben Amtoft, Charles Consel, Olivier Danvy, and Karoline Malmkjær. The abstraction and instantiation of string-matching programs. In *The Essence of Computation: Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones*, Torben Mogensen and David Schmidt and I. Hal Sudborough (editors), pages 332–357, Springer LNCS 2566, 2002. An extended version appeared as Technical Report BRICS RS-01-12, DAIMI, Aarhus, Denmark, April 2001.
- [27] Hanne Riis Nielson, Flemming Nielson, and Torben Amtoft. Polymorphic subtyping for effect analysis: the static semantics. In *Analysis and Verification of Multiple-Agent Languages*, pages 141–171, Springer LNCS 1192, 1997. Acceptance rate: 87 %.
- [28] Torben Amtoft, Flemming Nielson, Hanne Riis Nielson, and Jürgen Ammann. Polymorphic subtyping for effect analysis: the dynamic semantics. In *Analysis and Verification of Multiple-Agent Languages*, pages 172–206, Springer LNCS 1192, 1997. Acceptance rate: 87 %.
- [29] Flemming Nielson, Hanne Riis Nielson, and Torben Amtoft. Polymorphic subtyping for effect analysis: the algorithm. In *Analysis and Verification of Multiple-Agent Languages*, pages 207–243, Springer LNCS 1192, 1997. Acceptance rate: 87 %.

**[Miscellaneous]**

- [30] Torben Amtoft, Anindya Banerjee, Matthew Dwyer, John Hatcliff, Robby, and Virgil Wallentine. Model driven development for component-based integration of high-confidence medical software. Presented at the High Confidence Medical Device Software and Systems (HCMDSS) Workshop, June 2–3, 2005, Philadelphia, PA.
- [31] Torben Amtoft and J.B. Wells. Mobile processes with dependent communication types and singleton types for names and capabilities. Technical report 2002-3, Department of Computing and Information Sciences, Kansas State University, December 2002.
- [32] Ian Westmacott, J. B. Wells, Robert Muller, and Torben Amtoft. A mechanical verification of region inference: using an automatic theorem prover to verify a type-based program transformation. Submitted for publication, 2002.

- [33] Torben Amtoft. Causal type system for ambient movements. Submitted for publication, October 2001. Technical report 2002-04, Department of Computing and Information Sciences, Kansas State University, 2002.
- [34] Torben Amtoft. Partial evaluation for constraint-based program analyses. BRICS Technical Report BRICS-RS-99-45, DAIMI, University of Aarhus, Denmark, 1999.
- [35] Torben Amtoft. Strictness types: An inference algorithm and an application. Technical Report PB-448, DAIMI, University of Aarhus, Denmark, August 1993.
- [36] Torben Amtoft. *Sharing of Computations*. PhD thesis, DAIMI, University of Aarhus, Denmark, 1993. Technical report PB-453.
- [37] Torben Amtoft and Jesper Larsson Träff. Memoization and its use in lazy and incremental program generation. Master's thesis, DIKU, University of Copenhagen, Denmark, August 1989. No. 89-8-1.

## Software Development

Concerning small-scale programming, I have many years of experience with mainly functional but also imperative and logic languages. On a somewhat larger scale, in Spring 1997 I implemented (assisted by Kirsten Gasser) a system for behavior analysis of CML (based on [1]); the system is programmed in Moscow ML and makes use of the “compilation unit” system (to gain modularity) and a selection of efficient data structures (to achieve satisfying response times). In a similar vein, in Spring 1998 I developed a system that implements and visualizes a number of the type and effect systems described in Chapter 5 of the book *Principles of Program Analysis* by Flemming Nielson, Hanne Riis Nielson and Chris Hankin.

## Invited Talks and Research Presentations

### Conference/workshop talks

*From Generic to Specific: Off-line Optimization for a General Constraint solver*. GPCE'08, Nashville, Tennessee, October 2008.

*Verification Condition Generation for Conditional Information Flow.* FMSE'07, Fairfax, Virginia, November 2007.

*A New Foundation for Control-Dependence and Slicing for Modern Program Structures.* ESOP'05, Edinburgh, Scotland, April 2005.

*Information Flow Analysis in Logical Form.* SAS'04, Verona, Italy, August 2004.

*What are polymorphically-typed ambients?* ESOP'01, Genova, Italy, April 2001.

*Faithful translations between polyvariant flows and polymorphic types.* ESOP'00, Berlin, Germany, March 2000.

*Local type reconstruction by means of symbolic fixed point iteration.* ESOP'94, Edinburgh, Scotland, April 1994.

*Minimal thunkification.* WSA'93, Padova, Italy, September 1993.

*Unfold/fold transformations preserving termination properties.* PLILP'92, Leuven, Belgium, August 1992.

*Properties of unfolding-based meta-level systems.* PEPM'91, New Haven, Connecticut, June 1991.

### **Invited talks (recent)**

*Slicing for Modern Program Structures: a Theory for Eliminating Irrelevant Loops.* Technical University of Denmark, June 2nd, 2008.

*Verification Condition Generation for Conditional Information Flow.* Northeastern University, May 30th, 2007.

*A Logic for Information Flow in Object-Oriented Programs.* Technical University of Denmark, May 23rd, 2006; University of Copenhagen, May 22nd, 2006.

*Information Flow Analysis in Logical Form.* University of Copenhagen, August 5th, 2004; Open Source Quality Project Retreat (Santa Cruz, California), May 14th, 2004.

*The Semantic Soundness of a Type System for Interprocedural Register Allocation and Constructor Flattening.* Boston University, June 2nd, 2003; Northeastern University, May 28th, 2003. (These presentations were assisted by Robert Muller.)

*Causal Type System for Ambient Movements.* Boston University, December 16th, 2002.

*Causal Type Systems for Ambients.* Technical University of Denmark, August 17th, 2001.

*What are Polymorphically Typed Ambients?* Kansas State University (interview talk), April 26th, 2001; Università Ca' Foscari di Venezia, April 10th, 2001; NEPLS at Brown University (Providence, RI), December 7, 2000.

*Faithful Translations between Polyvariant Flows and Polymorphic Types.* University of Copenhagen, April 3rd, 2000; NJPLS (New Jersey Programming Language Seminar), September 1, 1999.

*Behaviour Analysis for Validating Communication Patterns.* Northeastern University, March 15th, 2000.

## **Outreach**

I am part of a team at Kansas State University which is collaborating with Rockwell Collins' Advanced Technology Center (ATC) in Cedar Rapids, Iowa. This includes a visit in June 2006 (prior to which I had authored white papers on possible collaboration) when I presented work on information flow analysis for heap-manipulating programs, a visit in June 2007 when I presented work on conditional information flow for array-manipulating programs (and also gave a talk to motivate separation logic), and a visit in October 2007 where I went into deeper details on conditional information flow. Our techniques appear able to handle features that are very important areas of ongoing research for ATC; therefore they have funded us with a preliminary grant, and we are writing a joint proposal to the NSA (National Security Agency).

## **Professional Contributions**

### **Refereeing**

### **Program Committees**

I shall serve on the program committee for

- ESOP'09 (European Symposium on Programming), held as part of ETAPS in York, United Kingdom, Spring 2009. The proceedings will be published in the Springer LNCS series.

I served on the program committees for

- ICFP'06 (The 11th ACM SIGPLAN International Conference on Functional Programming), held in Portland, Oregon, September 2006.
- ESOP'04 (European Symposium on Programming), held as part of ETAPS in Barcelona, Spain, Spring 2004. The proceedings are published as LNCS volume 2986.
- ITRS'04 (Workshop on Intersection Types and Related Systems), held co-located with LICS and ICALP in Turku, Finland, July 2004.
- PADO-II (Programs as Data Objects), a symposium held together with MFPS 2001 in Aarhus, Denmark, May 2001. The proceedings are published as LNCS volume 2053.

## Journals

I have served as a reviewer on 22 journal submissions: for *Computer Languages*, in 2001; for *Formal Aspects of Computing*, in 2006; for *Higher-Order and Symbolic Computation*, in 2000 & 2002 & 2004 & 2005–7; for *Information and Computation*, in 2001 (joint with Assaf Kfoury and Santiago Pericas-Geertsen) & 2002–3 & 2006–7; for *Information Processing Letters*, in 2004 & 2005; for *Journal of Functional Programming*, in 1992 & 1996 & 2003 & 2007; for *Theoretical Computer Science*, in 2006–7 & 2007; for *ACM Transactions on Programming Languages and Systems*, in 1994 & 2005 & 2005–6 & 2006 & 2006–7.

## Conferences

On numerous occasions, I have reviewed conference submissions on request from members of program committees. The conferences include POPL (Principles of Programming Languages), 13 submissions; SAS (Static Analysis Symposium), 13 submissions; ICFP (International Conference on Functional Programming), 11 submissions; ESOP (European Symposium on Programming), 8 submissions; CONCUR (Conference on Concurrency The-

ory), 7 submissions; ICALP (International Colloquium on Automata, Languages, and Programming), 5 submissions; PEPM (Partial Evaluation and Semantics-based Program Manipulation), 5 submissions.

### **Invited Seminar and Conference Participation**

Workshop-fest in Honor of Neil Jones, Copenhagen, Denmark, August 25-26, 2007.

First International Workshop on Programming Language Interference and Dependence (PLID). Verona, Italy, August 25, 2004.

Dagstuhl Seminar 03411 on Language-Based Security. October 5–10, 2003.

Advanced Course on the *Principles of Program Analysis*. Dagstuhl Event No 98451, November 9–13, 1998.

### **Administration**

During the LOMAPS project (1993–97), I assisted the project manager Flemming Nielson by compiling deliverables to be sent to Brussels (based on contributions from the project partners), by arranging project meetings, and by maintaining the project’s Web-page. For the Church Project, I organized the seminar schedule. For the ULTRA Project, I was responsible for the design and development of the project web pages.

### **Professional Development**

I have attended numerous international conferences, for example POPL (Principles of Programming Languages) 5 times (1995, 1997, 1999, 2000, 2001), and ESOP (European Symposium on Programming) 5 times (1990, 1994, 2000, 2001, 2005).

Also, I participated in the biannual meetings of the LOMAPS project (1993–1997), and in several meetings within the projects DART (Design, Analysis and Reasoning about Tools), Semantique (an ESPRIT project), Atlantique (a joint European and American research project), and more recently the New England Programming Languages and Systems Symposium Series (NE-PLS).

In April 2006, I attended an NSF sponsored workshop in Wichita, Kansas, on writing (CAREER) grant proposals. In April 2002, at Heriot-Watt University, I took the course *Developing Postgraduates' Teaching Skills* with six 3-hour sessions; at the end, I successfully completed an assignment on reflecting upon previous teaching experiences.

In July of 2004, Franklyn Turbak visited Anindya Banerjee and me for one week, in particular discussing the notion of declassification. Since moving to Kansas, I have several times visited Bob Muller at Boston College, continuing our work on type-based register allocation. In April 2001, I spent a few days visiting Michele Bugliesi at Università ‘Ca Foscari’, Venezia, exchanging ideas about advanced type systems for the ambient calculus. In April 1996, I made a one-week visit to ECRC (Munich), and in November 1996, I made a one-week visit to ICL (London), in both cases visiting Lone Leth and Bent Thomsen; we worked on an analysis for the concurrent language Facile. In January 1995, I spent 3 weeks at Northeastern University, Boston, visiting Mitch Wand and his group; we worked on constraint based program analysis.

## Personal Record

**Nationality:** citizen of Denmark; permanent resident of the U.S.

**Languages:** Danish; English; a little (and rusty) German, Russian, French.

**References** can be obtained from, e.g., Flemming Nielson, Hanne Riis Nielson, Olivier Danvy, Assaf Kfoury, J.B. Wells, David Schmidt, Anindya Banerjee, John Hatcliff.

## Appendix: Detailed Summary of Research

Below is a description of my research contributions, categorized by topic; a chronological ordering of the topics has been attempted.

**Memoization and theory of computation.** [25] investigates the meaning and practical use of the 2nd recursion theorems by Kleene and Rogers.

My M.Sc. thesis [37] is about how to improve efficiency of program execution by means of a generalized form of memoization, resembling partial evaluation; as reported in [9] Cook’s ingenious linear-time simulation of 2DPDAs can be thought of as an instance of this technique.

**Models for program optimization.** The main purpose of my Ph.D. thesis [36] is to develop a model enabling one to reason about various techniques for program optimization, in particular wrt. *speedup* and *correctness*. Concerning speedup, some of the results are presented in [24]; in particular the reasons why a program transformation may yield *more than* a constant speedup are factored out. Concerning correctness, some results (generalizing previous approaches from the literature) about preservation of termination properties for a logic language are presented in [23].

**Specification of analysis and transformation.** In [35], strictness analysis is formulated in terms of type inference. A type reconstruction algorithm is presented, and the strictness information is used to avoid some superfluous “thunkifications” when translating from call-by-name into call-by-value. Of particular interest is the proof technique: the correctness of the translation is proved *simultaneously* with the correctness of the analysis. The part concerning type reconstruction is published in [21]; the part concerning translation is published in [22].

**Effect analysis for concurrent systems.** [8] develops a sound and complete type and behavior reconstruction algorithm for a fragment of Concurrent ML (CML), the starting point being the inference system presented by Hanne Riis Nielson and Flemming Nielson at POPL’94. The algorithm returns a set of constraints; and we show how to solve these in the monomorphic case (but not in general).

The monograph [1] gives an overview over type and effect systems, and then (improving upon the results of [27], [28] and [29]) develops an annotated type and effect system for a fragment of CML; the system uses constraints on the left hand side of the turn-stile and integrates Hindley-Milner polymorphism, subtyping, and effects. We show that the system is semantically sound; and develop a reconstruction algorithm that is sound and also complete. This algorithm has been used as the basis of a prototype implementation, available for experimentation on the WWW. [7] contains a description of the system, illustrated by several examples, as well as a brief account of the

underlying theory. [20] shows that the system greatly assists in validating a number of safety properties for “realistic” concurrent systems.

**Applications of partial evaluation.** [34] reports on experiments investigating whether control flow analysis can be optimized by partially evaluating the analyzer. So far the results have been negative, except that the residual program pinpointed a serious source of inefficiency, leading to the (re)invention of an incremental version of the analyzer.

[26] exposes how one by partial evaluation of a single generic string matching algorithm can achieve the effect of the Knuth & Morris & Pratt string matcher, as well as the effect of (several variants of) the Boyer & Moore string matcher. This has been known for at least a decade, when similar results were made public by the authors (together and independently); the primary goal of this paper is thus to summarize the findings and put them into perspective.

**Frameworks for polyvariant analysis.** [19] demonstrates that there is a close relationship between polyvariant flow analyses and type systems with finitary polymorphism. We present a flow logic, based on the general approach of Nielson & Nielson and augmented with ideas from Palsberg & Pavlopoulou, and also present a type system employing union and intersection types; both these systems satisfy a subject reduction property. We then provide translations between types and flows that are “faithful” in that they act as the identity on “canonical” elements, and otherwise canonicalize.

**Type systems for the ambient calculus.** [18] and [33, 2] consider the Ambient Calculus, proposed by Cardelli and Gordon as a formal framework to study issues of mobility and migrant code, and develop type systems for the calculus. These systems employ a notion of causality in that processes are assigned “behaviors”, where a behavior is essentially a regular set of traces. Thus type checking (of fully annotated processes) is decidable, using techniques borrowed from finite automata theory. (Under certain restrictions, type inference is also possible.)

In [18], the focus is on extending the ambient calculus so as to allow a more natural, yet safe, style of programming. This is done by embedding a functional language, and by designing the type system to smoothly integrate several kinds of “polymorphism”: (*i*) the well-investigated notion of

subtyping; *(ii)* “arity polymorphism”, allowing the same ambient to hold several topics of conversation *simultaneously*; and *(iii)* “orderly communication”, allowing the same ambient to hold several topics of conversation *consecutively*. A subject reduction property ensures that communicating subprocesses agree on their “topic of conversation”. As “orderly communication” is the main technical innovation of the above, the journal paper [6] concentrates on this feature only.

In [33], summarized and put into perspective in [2], the focus is on security in that the type system is parameterized by a set of security constraints: static ones expressing where a given ambient may reside, and dynamic ones expressing where a given ambient may be dissolved. A subject reduction property then guarantees that a well-typed process never violates these constraints. It is argued that the presence of causality significantly increases the precision of the analysis and compensates for the lack of “co-capabilities” (an otherwise increasingly popular extension to the ambient calculus).

The goal of [31], and the further development in [16], is to provide type polymorphism of the kind that is usually present in polymorphic type systems for the  $\lambda$ -calculus, thereby allowing mobile agents to follow non-predetermined paths and to carry non-predetermined types of data from location to location. This is achieved by letting the type of an ambient process  $P$  give an upper bound on the possible ambient nesting shapes of any process  $P'$  to which  $P$  can evolve. Because these shapes can depend on which capabilities and names are actually communicated, the types support this with explicit dependencies on communication. The type of an ambient name may thus depend on where the ambient has traveled, whereas in previous type systems for ambient calculi, there is a global assignment of types to ambient names.

**Type systems for register allocation.** In [17], we design for a compiler intermediate language an annotated type system supporting interprocedural register allocation and the representation of tuples and variants directly in the register file.

**Information Flow Analysis.** In [15], we specify an information flow analysis for a simple imperative language, using a Hoare logic. The logic facilitates static checking of a larger class of programs than can be checked by extant type-based approaches in which a program is deemed insecure when it contains an insecure subprogram. The logic is based on an abstract interpretation of program traces that makes independence between program

variables explicit. Unlike other, more precise, approaches based on Hoare logic, our approach does not require a theorem prover to generate invariants. We demonstrate the modularity of our approach by showing that a frame rule holds in our logic.

In [5], we extend the results of [15] to handle also nontermination sensitive information flow analysis, and show how the logic gives rise to a (provably correct) algorithm for forward slicing.

In [13], we modify the logic of [15] to handle object-oriented languages; to reason about aliasing, ubiquitous in such languages, the information flow logic is built on top of a logic for abstract locations (just as any analysis for higher order functional programs is built on top of a control-flow analysis). The logic enjoys “small” specifications, so as to facilitate modular reasoning; these can be combined by a frame rule. Under certain assumptions, it is possible to compute “strongest postconditions”. Our language permits programmer assertions, in the style of ESC/Java.

In [12], we present an alternative approach to information flow analysis of sequential heap manipulating programs. We use “object flow invariants” to express the information flow properties an object must satisfy; such an invariant may be temporarily violated while an object is being updated but must be restored at the end (when the “scope” of the object is closed). Hence there is no need for explicit reasoning about aliasing. In general, information flow properties are expressed using assertions that are conditional in that they depend on standard Hoare assertions being satisfied. We define an algorithm `VCgen` which from a program and its desired postcondition generates a precondition which is strong enough to establish the postcondition. `VCgen` must be supplied with object flow invariants as well as with flow invariants for loops; if these invariants are not strong enough then the verification conditions generated by `VCgen` cannot be satisfied. The current analysis is intraprocedural.

In [11], we employ the techniques of [12], extended to an interprocedural setting, to enhance the SPARK information flow annotation language with conditional information flow contracts. Unlike [12], we now have a method for automatically inferring loop flow invariants; therefore contracts can be compositionally checked and inferred (the SPARK subset of Ada deliberately omits constructs that are difficult to reason about; hence we do not need to worry about inferring flow invariants for heap objects). We report on the use of this framework for a collection of SPARK examples; our experiments are based on an implementation that allows various degrees of assertion

simplification.

**Slicing.** In [14], we examine the notion of *control dependence*, underlying many program analysis techniques such as slicing. We argue that existing definitions are difficult to apply seamlessly to modern program structures which make substantial use of exception processing and increasingly support reactive systems designed to run indefinitely; we repair on that by developing definitions that apply also for control flow graphs without end nodes (or with more than one end node). For these new definitions, we show that they conservatively extend classic definitions, and show that the slicing algorithm induced by them is correct, wrt. a correctness criterion based on weak bisimulation. Algorithms for computing the new control dependences form the basis of a publicly available program slicer that has been implemented for full Java.

In [4], we extend [14] so as to handle control flow graphs without end nodes also if they are *irreducible*. This requires a new notion of control-based dependence, called “order dependence”. A detailed correctness proof is given for the slicing induced by the modified definitions.

[3] provides a feature missing in [4]: the foundation of an approach to slicing which allows for the elimination of loops that do not affect the values of relevant variables, and thereby is more likely to generate slices of manageable size. The corresponding correctness criterion is based on “weak simulation”, implying that the observational behavior of the original program is a *prefix* of the behavior of the sliced program. A crisp correctness proof shows that for slicing to satisfy this correctness property, even wrt. control flow graphs that are irreducible or have no end nodes, it is sufficient that the given slice set is closed under (data dependency and) “weak control dependency”, one of the new dependencies proposed in [4].