Curriculum vitae

Torben Amtoft
email: tamtoft@cis.ksu.edu
URL: www.cis.ksu.edu/~tamtoft

November 14, 2004

1 Personal Record

Birth: the 6th of June, 1963, in Copenhagen.

Nationality: Danish citizen.

Marital status: Never married.

Languages: Danish; English; a little German, Russian, French.

2 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.

3 Employments

2002- Assistant professor (tenure track) 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.

4 Research Topics (chronologically)

Memoization and theory of computation. [16] investigates the meaning and practical use of the 2nd recursion theorems by Kleene and Rogers. My M.Sc. thesis [28] is about how to improve efficiency of program execution by means of a generalized form of memoization, resembling partial evaluation; as reported in [5] 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 [27] 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 [15]; 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 [14].

Specification of analysis and transformation. In [26], 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 [12]; the part concerning translation is published in [13].

Effect analysis for concurrent systems. [4] 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 [18], [19] and [20]) 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, cf. Sect. 9. [3] contains a description of the system, illustrated by several examples, as well as a brief account of the underlying theory. [11] shows that the system greatly assists in validating a number of safety properties for "realistic" concurrent systems.

Applications of partial evaluation. [25] 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.

[17] 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. [10] 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. [9] and [24] 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 [9], 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 [2] concentrates on this feature only.

In [24], 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 [22], and the further development in [7], is to provide type polymorphism of the kind that is usually present in polymorphic type systems for the λ-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 [8], 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 [6], 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. Moreover, given an insecure but terminating program, we show how strongest postconditions can be employed to statically generate failure explanations.

Slicing. In [21], 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 more suitable definitions. 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.

5 Teaching

5.1 Undergraduate Courses

Fall 2002 and onwards, Kansas State University I teach 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.

Spring 2001, Boston University Together with Assaf Kfoury, I taught an undergraduate course "CS525" on compiler design theory (www.cs.bu.edu/~kfoury/CS525-Spring01/520/index.html), 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.

5.2 Graduate Courses & Seminars

Spring 2004, Kansas State University I taught a course (CIS 761) on *Database Management Systems*.

Fall 2003, 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, I (assisted by Dave Schmidt) gave a "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 an almost weekly seminar 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.

5.3 Miscellaneous

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.

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 the years 1990–1992, I was a teaching assistant at the University of Aarhus on the first-year course "Dat1"; and in the years 1985-1999, I was a teaching assistant at the University of Copenhagen on the third-year course "Dat2". In both cases, 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.

6 Refereeing

6.1 Program Committees

I served on the program committee for PADO-II (Programs as Data Objects), a symposium organized by Andrzej Filinski and Olivier Danvy with an international program committee and refereed submissions, held together with MFPS 2001 in Aarhus, May 2001. The proceedings are published as LNCS volume 2053.

I served on the program committee for ESOP'04 (European Symposium on Programming), held as part of ETAPS in Barcelona, Spain, Spring 2004. The proceedings are published as LNCS volume 2986.

I served on the program committee for ITRS'04 (Workshop on Intersection Types and Related Systems), held co-located with LICS and ICALP in Turku, Finland, July 2004.

6.2 Journals

I have served as a reviewer on 10 journal submissions: for *Computer Languages*, in 2001; for *Higher-Order and Symbolic Computation*, in 2000 &

2002; for *Information and Computation*, in 2001 (joint with Assaf Kfoury and Santiago Pericas-Geertsen) & 2002/3; for *Information Processing Letters*, in 2004; for *Journal of Functional Programming*, in 1992 & 1996 & 2003; for *ACM Transactions on Programming Languages and Systems*, in 1994.

6.3 Conferences

On numerous occasions, I have reviewed conference submissions on request from members of program committees. The conferences include SAS (Static Analysis Symposium), 13 submissions; ICFP (International Conference on Functional Programming), 10 submissions; POPL (Principles of Programming Languages), 10 submissions; CONCUR (Conference on Concurrency Theory), 6 submissions; ESOP (European Symposium on Programming), 6 submissions; PEPM (Partial Evaluation and Semantics-based Program Manipulation), 4 submissions; ICALP (International Colloquium on Automata, Languages, and Programming), 3 submissions.

7 Scientific Interactions

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) 4 times (1990, 1994, 2000, 2001).

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 (NEPLS). In October 2003, I attended a Dagstuhl seminar on language-based security; and in May 2004, I attended the Open Source Quality Project Retreat, Santa Cruz, California.

7.1 Invited Talks

- I presented the work described in [6] on August 5th, 2004, at the University of Copenhagen.
- Assisted by Robert Muller, I presented our work extending [8] on May 28th, 2003, at Northeastern University; on June 2nd, 2003, at Boston University.

- I presented the work described in [24] on December 16th, 2002, at a graduate seminar at Boston University.
- I presented the work described in [9] on April 10th, 2001, at Universita Ca' Foscari di Venezia; on April 26th, 2001, at Kansas State University (when interviewing); on August 17th, 2001, at the Technical University of Denmark.
- I presented the work described in [10] on April 3rd, 2000, at the University of Copenhagen.
- I presented the work described in [3] on March 15th, 2000, at Northeastern University.

7.2 Shorter Visits

In January 1995, I spent 3 weeks at Northeastern University, Boston, visiting Mitch Wand and his group; we worked on constraint based program analysis.

In April 1996, I made a one-week visit to ECRC (Munich) and 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 April 2001, I spent a few days visiting Michele Bugliesi at Universita' "Ca Foscari", Venezia, exchanging ideas about advanced type systems for the ambient calculus.

In May & June of 2003, I spent a couple of weeks visiting Bob Muller at Boston College, extending the work reported in [8].

8 Administrative experience

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.

9 Programming experience

Concerning small-scale programming, I have many years of experience with mainly functional but also imperative and logic languages. On a somewhat

larger scale, 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.

References

[**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.) 3, 10

[**Journal papers**]

[2] 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). 4

[3] 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.) 3, 9

[4] 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. 3

[5] 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. 2

[**Conference papers**]

[6] 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. 5, 8

[7] 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. 4

[8] Torben Amtoft and Robert Muller. Inferring annotated types for interprocedural register allocation with constructor flattening. Proceedings of ACM SIGPLAN TLDI'03 Workshop, pages 86–97, ACM Press, January 2003. Acceptance rate: 42 %. 5, 8, 9

[9] 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. 4, 9

[10] 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 %. 3, 9

[11] 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 %. 3

[12] 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 %. 3

[13] Torben Amtoft. Minimal thunkification. In *Proc. WSA'93*, pages 218–229, Springer LNCS 724, 1993. Acceptance rate: 29 %. 3

[14] Torben Amtoft. Unfold/fold transformations preserving termination properties. In *Proc. PLILP'92*, pages 187–201, Springer LNCS 631, August 1992. Acceptance rate: 35 %. 2

[15] 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 %. 2

[16] 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. 2

[**Other reviewed papers**]

[17] 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. 3

[18] 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 %. 3

[19] 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 %. 3

[20] 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 %. 3

[**Miscellaneous**]

[21] Venkatesh Ranganath, Torben Amtoft, Anindya Banerjee, Matthew B. Dwyer and John Hatcliff. A New Foundation for Control-Dependence and Slicing for Modern Program Structures. Santos Technical report 2004-8, Department of Computing and Information Sciences, Kansas State University, October 2004. 5

[22] 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. 4

[23] 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.

[24] Torben Amtoft. Causal type system for ambient movements. Submitted for publication, October 2001. 4, 9

[25] Torben Amtoft. Partial evaluation for constraint-based program analyses. BRICS Technical Report BRICS-RS-99-45, DAIMI, University of Aarhus, Denmark, 1999. 3

[26] Torben Amtoft. Strictness types: An inference algorithm and an application. Technical Report PB-448, DAIMI, University of Aarhus, Denmark, August 1993. 2

[27] Torben Amtoft. *Sharing of Computations*. PhD thesis, DAIMI, University of Aarhus, Denmark, 1993. Technical report PB-453. 2

[28] 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. 2

A personal record or personal best (frequently abbreviated to PR or PB) is an individual's best performance in a given sporting discipline. It is most commonly found in athletic sports, such as track and field, other forms of running, swimming and weightlifting. The term "PR" came from the world of running, referring to a person's best time in a race of a specific distance. So, if someone runs their first 5K race in 28:45, that's their PR for the 5K distance. E-1 Personal Record Form. SSS membership application form for the issuance of SSS Number. Your Browser Doesn't Support Canvas. Showing the Text Content of the PDF Instead: Republic of the Philippines. Social security system. E-1. Ss number. Personal record for issuance of ss number. COV-01214 (09-2015). This form may be reproduced and is not for sale. A Personal Record is an autobiographical work (or "fragment of biography") by Joseph Conrad, published in 1912. It has also been published under the titles A Personal Record: Some Reminiscences and Some Reminiscences. Notoriously unreliable and digressive in structure, it is nonetheless the principal contemporary source for information about the author's life. It tells about his schooling in Russian Poland, his sailing in Marseille, the influence of his uncle Tadeusz Bobrowski, and the writing of