

Ph. D. Study Plan

Noah Torp-Smith
The IT University of Copenhagen

September, 2004

Student

Noah Torp-Smith, CPR: (Omitted in web-version)
IT University of Copenhagen
Department of Theoretical Computer Science
Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark
Phone: +45 7218 5288, e-mail: noah@itu.dk

Project: Reasoning about Resources
Admission: August 1st, 2001
Submission of thesis: July 31st, 2005

Supervisor

Associate Professor Lars Birkedal
IT University of Copenhagen
Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark
Head of Department of Theoretical Computer Science
Phone: +45 7218 5280, e-mail: birkedal@itu.dk

Preface

[September 2003] This is a revised study plan. The time schedule has been modified to conform with the deadlines for half year reports at the IT University.

[March 2004] Publication list, talk list, plans for teaching updated. Time schedule modified slightly to become more realistic.

[September 2004] Publication list updated, plan for teaching updated. Time schedule updated to include studies of reflexive graphs, data abstraction and modularity.

Project Description

Objective

Practical programming languages used in industry, such as C, traditionally provide explicit access to computational resources such as, for example, the memory of a computer. This approach provides relatively fine-grained control over resource consumption, but the drawback is that it is extremely error-prone, since a sufficient in-depth understanding of the semantics of resources in a programming context does not exist as of now. The goal of this project is to provide some of this knowledge.

Background

It is important to be able to reason about resources. One of the major applications is in *program verification*: a better comprehension of resources can be used to verify that a program (written in a language which provides control over computational resources) has certain properties. This is tightly connected to the subject of *program logics*: one needs to use a logic in which it is possible to express and prove the properties we are dealing with. One often also uses a *type system* to aid the process of program verification. When one deals with different type systems, and one needs to compare them, some models are built, and these are compared. Category theory is used to unify and compare the different models of computation, logics, and type systems, so it is important to study category theory to get a better understanding of the semantics of resources. This insight can in turn be used to design programming languages with specially designed constructs for handling resources, where the gained knowledge can be used to give reasoning principles for these constructs. This can narrow the gap between languages with explicit control (and a lack of reasoning principles) and those with a well-developed set of reasoning principles (but a lack of control over resources).

Some proposals for resource oriented logics have already been made, including Girard's *Linear Logic*, and the *Logic of Bunched Implications* (BI) by Pym and O'Hearn. In addition to a better insight in the semantics of resources, the last of these has proven to be very useful for the purpose of program verification, and different branches of it have been used to prove correctness of different algorithms; the most complicated example so far is undoubtedly in Yang's thesis [7, chp. 7], where Schorr-Waite's graph marking algorithm is proved correct.

One of the goals of this project is to use, relate, and compare some of the branches of BI. Roughly speaking, different branches of BI can be used as program logics for programming languages with different constructs to manipulate resources, and these logics can then be compared by considering different *doubly closed categories* (DCCs) that have properties corresponding to the characteristics of the branches of BI. This will give us a better comprehension of the matter, which can then be applied to other areas. This will be done by addressing the open research problems that are indicated below.

Research Questions

As mentioned, there is still a severe lack of understanding of the formal semantics of resources in a programming context. Here is a short list of issues that need to be addressed. More problems are likely to arise along the way.

Garbage Collection In [2], a program logic in a garbage collected setting is proposed, and a correctness criteria for a garbage collector is given. It is an important open problem to implement and prove the correctness of such a garbage collector. We believe it can be done using program logics related to that in [4], which is based on BI. If successful, we can also start working on other graph algorithms than garbage collectors. Another interesting problem is the following: The garbage collector is part of a memory manager for some (relatively high-level) *user language*, whereas the garbage collector itself is written in some (low-level) *implementation language*. How does one formally describe the *interface* between these two languages, and how should the program logics for the two languages be related? This is highly related to the modularity question described below.

Mobility and Concurrency How can the ideas of program logics for sequential languages based on BI be extended to provide reasoning principles for shared computational resources for parallel and communicating systems? More generally, we seek reasoning principles for resources in mobile systems.

Connection to Type Systems There is a tight connection between program verification and type systems (cf. types for low-level languages like TAL, alias types, etc.). It will be worthwhile to consider whether the new insight that one gain from the work with the other questions can be applied to type systems.

Modularity In the world of object-oriented programming languages, the notions of *modules* and their *interfaces* are important. Modules might utilize other modules' interfaces and this creates an important abstraction, since the programmer is free to choose how to *implement* the module. However, if the programming language in which the module is implemented has explicit access to memory, it is not clear what the criteria are for two implementations to be equivalent (and therefore interchangeable). Some work has already been done in this regard by Banerjee and Naumann [1], by Clarke, Noble, and Potter [3], and by Reddy and Yang [5]. During my visit to Queen Mary, University of London, I have worked in Prof. O'Hearn's research group, and a lot of focus was directed on this area. In particular, we have worked with the notion of a *separation context*, and have tried to devise this notion to give conditions (relying on Separation Logic) for two implementations to be equivalent.

Graph Representation In the course of proving the correctness of a garbage collector, the following question arises: what does it mean for a graph to be represented in a heap? Since graphs are not inductively defined, it is not possible to give a straightforward definition like for trees and lists [4], [6]. It is interesting to see if one can give a *recursive* specification, and prove the existence of such specifications using a method due to Freyd and Pitts. This might ease the reasoning when proving properties about programs that manipulate graph-like structures.

In my research report, there is a formal proof of a copying garbage collector which uses a variant of BI as a program logic, so at least some of the questions in the first item in this list have been answered.

International Research Collaboration

In the spring of 2003, I have visited Prof. Peter O'Hearn's research group at Queen Mary University of London. Given O'Hearn's publication list, and the people that take part in this research group (e.g., Bornat, Calcagno), this was the most relevant place in the world for me to go to. This collaboration has also resulted in a conference paper written in collaboration with a ph. d. student (I. Mijajlovic) from Queen Mary. Furthermore, I have written an article in collaboration with Prof. John C. Reynolds from Carnegie Mellon University, Pennsylvania which has been accepted for publication in the proceedings of POPL'04.

Educational Requirements

The educational requirements for a Ph.D. student in the 4+4 arrangement are that the student should obtain 60 ECTS points. This is expected to be fulfilled by attending Ph. D. courses, summer schools, workshops, conferences, etc. I have already fulfilled these requirements.

- *Introduction to Semantics for Programming Languages*. Course at Dept. of Computer Science, University of Copenhagen, held by Andrzej Filinski, Fall 2001. 10 ECTS, passed.
- *Reasoning about Resources*. Ph.D. seminar at the IT University, organized by Lars Birkedal. Fall 2001, 7.5 ECTS, passed.
- *Topos Theory Seminar* Ph. D. seminar organized by Lars Birkedal at the IT University. Fall 2001, 7.5 ECTS, passed.
- *Domains and Lambda-Calculi*. Ph. D. course at the IT University by Carsten Butz. Spring 2002, 7.5 ECTS.
- *Models and Languages for Concurrency and Mobility* Ph. D. course by Lars Birkedal at the IT University of Copenhagen. Spring 2002, 7.5 ECTS.
- *Ecole de Printemps 2002 - Semantiques des Langages des Programmation*. Spring School in Agay, France, March 24 - 29, 2002.
- *Category Theory and Computer Science (CTCS'02)*. Conference at the University of Ottawa, Canada, August 12 - 17, 2002 (including a preconference for students)
- *Topos Theory Seminar*. Ph. D. seminar organized by Lars Birkedal at the IT University of Copenhagen. Fall 2002, Fall 2003, Spring 2004. 7.5 ECTS each, 22.5 ECTS total.

- *Models for Context-Dependent Mobile Communication* Ph. D. Seminar organized by Thomas Hildebrandt at the IT University of Copenhagen. Fall 2002. 7.5 ECTS.

Independent Studies

As a Ph.D. student, it is vital that one studies material that is beyond the specific courses that one attends, in order to get a clearer picture of the research area. In the bibliography of my research report, some examples of what I have studied independently, can be seen. I will continue to keep up with the research area by reading relevant articles.

Presentation Requirements

According to “Cirkulære om Aflønning af ph.d.-studerende for arbejdsopgaver i forbindelse med forskeruddannelsen” from the Danish Finance Ministry, 840 hours of work is to be done for The IT University. This is expected to be fulfilled by writing papers and technical reports, attending conferences and workshops, and teaching courses.

Teaching and Supervising Student’s Projects

560 out of the 840 hours is to be spent on teaching. I have done the following to meet this requirement:

Web Programming, Spring 2002 — I have prepared and given two lectures in the spring semester of 2002.

Student’s Projects, Spring 2002 — I supervised 17 student’s projects jointly with Thomas Hildebrandt in the spring semester of 2002.

Student’s Projects, Fall 2002 — I supervised 12 student’s projects jointly with Rasmus Møgelberg in the fall semester of 2002.

Student’s Projects, Fall 2003 — I supervised 10 student’s projects jointly with Rasmus Møgelberg in the fall semester of 2003.

IPBR, 2004 I am currently teaching the course “Introduktion til Programmering – Begreber og Redskaber” jointly with Jens Chr. Godskesen and Nina Bohr. I am giving 7 out of 12 lectures in this course.

Students’ Projects, Spring 2004 I taught 19 students’ projects in the four week projects period after the Spring semester in 2004, in collaboration with Jens Chr. Godskesen.

Papers, Reports, Presentations, and Conferences

It is unrealistic to exhibit a schedule for publishing, but a few topics for publications have already been investigated, and I am currently working on the garbage collection issue mentioned under the research problems. I have identified some conferences that are interesting, both as publishing fora and as opportunities for improving knowledge. Major conferences of interest are: *Principles of Programming Languages* (POPL), *Logic in Computer Science* (LICS), *Category Theory and Computer Science* (CTCS), and *Mathematical Foundations of Programming Semantics* (MFPS), and more can be mentioned.

Publications

- N. Torp-Smith, *Proving Correctness of a Garbage Collector via Local Reasoning*. Master's thesis, The IT University of Copenhagen, July 2003.
- L. Birkedal, N. Torp-Smith, J. C. Reynolds, *Correctness of a garbage collector via local reasoning*. Technical Report 30, The IT University of Copenhagen, Denmark, July 2003.
- L. Birkedal, N. Torp-Smith, J. C. Reynolds, *Local Reasoning about a Copying Garbage Collector*. Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'04), Venice, Italy, 2004.
- Ivana Mijajlovic, Noah Torp-Smith, and Peter W. O'Hearn, *Refinement in a Separation Context*. Proceedings of the FSTTCS conference, Chennai, India, December 2004 (accepted).

Submitted Papers The following papers have been submitted, but have either not been reviewed yet, or need amendments before they can be published.

- N. Torp-Smith, L. Birkedal, and J. C. Reynolds, *Local Reasoning about a copying garbage collector*, submitted to ACM Transactions on Programming Languages and Systems (TOPLAS), July, 2004.
- B. Biering, L. Birkedal, and N. Torp-Smith, *BI hyperdoctrines and separation logic*, submitted to FSTTCS conference, July, 2004.

Talks

I have already presented the work on garbage collection at the following fora:

- Spatial Logics Workshop, Nottingham, 25th March 2003. In connection with the First APPSEM II workshop in Nottingham, 26th to 28th March 2003.
- 1st MM-NET Workshop on Analytical Techniques for Memory Management. University of Kent at Canterbury, 15th May 2003.
- The 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. January 14-16, 2004.

- Guest lecture on a minicourse by John Reynolds on Separation Logic at BRICS, Aarhus. December 2003.

Time Schedule

August 2001 - January 2002 Getting started at the new environment and at the Theory Department. Attend courses: *The formal Semantics of Programming Languages*, *Topos Theory Seminar I*, and *RAR seminar: Reasoning about Low-Level Languages*. Initiate research concerning resources and advanced category theory.

Jan 2002 - Jun 2002 Attend courses and seminars: *Domains and Lambda Calculi*, *Models and Languages for Concurrency and Mobility*, and *Topos Theory Seminar II*. Attend Spring School in Agay France: Ecole de Printemps 2002: *Semantiques de Langages de Programmation*. Teach Web Programming course and supervise students' projects. Continue research in logics and models for resources.

July 2002 - Dec 2002 Attend the conference *Category Theory and Computer Science* (CTCS'02), at the University of Ottawa. Supervise students' projects in connection with the Web Programming course. Start arranging stay at foreign research institution by initiating contact with relevant researchers from other universities. Intensify research in Logics and Languages for reasoning about resources.

Jan 2003 - June 2003 Stay at foreign university.

July 2003 - March 2004 Work with modularity and refinement issues, in collaboration with Prof. O'Hearn's research group at Queen Mary, University of London, and write a technical report about this. Write journal version of Master's thesis. Attend POPL'04 conference. Supervise student's projects in the project period. Teach Basic Programming in the Spring semester.

April 2004 - September 2004 Work on modularity, data abstraction, and refinement. Finish first solution to to separation logic and data abstraction, and test it on some examples. Consider more abstract models of separation logic. Study related work. Teach the course IPBR, supervise student's projects.

October 2004 - March 2005 Finish work on data abstraction, and submit paper on reflexive graphs, data abstraction, and modularity. Start writing thesis. Finish paper on BI hyperdoctrines.

April 2005 - July 2005 Finish thesis.

Signatures

Noah Torp-Smith

Lars Birkedal

Litteratur

- [1] A. Banerjee and D. A. Naumann. Representation independence, confinement and access control [extended abstract]. In *POPL 02*, 2002.
- [2] C. Calcagno, P. O’Hearn, and R. Bornat. Program logic and equivalence in the presence of garbage collection. *Theoretical Computer Science*, 298(3):557 – 581, 2003.
- [3] D. G. Clarke, J Noble, and J. M. Potter. Simple ownership types for object containment. In *Proc. European Conference on Object-Oriented Programming*, June 2001.
- [4] P. W. O’Hearn, J. C. Reynolds, and H. Yang. Local reasoning about programs that alter data structures. In *Proceedings of 15th Annual Conference of the European Association for Computer Science Logic: CSL 2001*, Lecture Notes in Computer Science, Berlin, 2001. Springer-Verlag.
- [5] U. S. Reddy and H. Yang. Correctness of data representations involving heap data structures. In P. Degano, editor, *Proc. of the 12th European Symposium on Programming, ESOP 2003*, pages 223 – 237. Springer Verlag, 2003.
- [6] J. C. Reynolds. Intuitionistic reasoning about shared mutable data structures. In J. Davies, B. Roscoe, and J. Woodcock, editors, *Millenial Perspectives in Computer Science*, pages 303 – 321. Palgrave, Houndsmill, Hampshire, 2000.
- [7] H. Yang. *Local Reasoning for Stateful Programs*. PhD thesis, University of Illinois, 2001.