Linear Logic: Theory and Applications
Scuola di Dottorato in Informatica della Università degli Studi di Milano
Curriculum di Logica Computazionale
April 28–May 11, 2011

Marco Gaboardi[1], Alberto Momigliano[2], and Carsten Schürmann[3]

Dipartimento di Scienze dell'informazione, Università degli Studi di Bologna, Italy
Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, Italy
IT University of Copenhagen

Introduction

Linear logic is a refinement of classical and intuitionistic logic. Instead of emphasizing *truth*, as in classical logic, or *proof*, as in intuitionistic logic, linear logic emphasizes the role of formulas as *resources.*

Logic, or at least proof-theory, is focused on formal proof systems: intuitionistic predicate calculus, classical predicate calculus, arithmetics, higher order calculi, and a wealth of similar consistent and structured sets of process-building rules. Intuitionistic and constructive logic began when people saw the possibility of reading $A \to B$ as if you give me an A, I will give you a B, which is a significant departure from the classical reading B is true whenever A is.

Computer science, on the other hand, focuses on computational mechanisms: function application, exception handling, method invocation in object oriented languages, variable assignment and similar sets of process-building rules. Except that the mechanisms of these processes have to made explicit: a function of type $A \Rightarrow B$ gives a formal account of how to *transform* an A into a B.

At a given moment these two sciences met. People realized that the set of implication-only intuitionistic deductions was a core functional language called simply-typed lambda-calculus: the programming language was a logic, the logic a programming language. This memorable meeting was called the Curry-Howard isomorphism.

Linear logic begins with a further twist in the reading of $A \to B$: read now give me as many A as I might need and I will give you one B. The notion of *copy* which is so central to the idea of computation is now wired into the logic.

This new viewpoint opens up new possibilities, including:

– new formulas expressing refined operational properties like give me A once and I will give you B. Applications here range from knowledge representation in AI, refined logic programming where the ability of linear logic to represent states is put to use, analysis of classical logic and computational interpretations thereof, namely exception mechanisms in programming languages, by means of embeddings in linear logic, refined temporal logics, linearity analysis.
– new rules expressing constraints on the use of copies resulting in a fragment of linear logic for polytime computations to mention only the most spectacular application.
– new ways of representing proofs

Practicalities

Instructors: Marco Gaboardi, Alberto Momigliano and Carsten Schürmann

Dates and time: April 28–May 11, 2011, 10am – 12:30am .

Duration: 24 hours.

Location: Sala Riunioni primo piano.

Pedagogical outcomes and prerequisites The course will provide the students with the tools to understand linear logic from a proof and model theoretic point of view. On top of that, students will learn how to use the state-of-the art (concurrent) linear logical framework and logic programming language *Celf* [8] from its main author. The only prerequisites to attend the course are a basic knowledge of logic and some exposition to logic programming.

Exams: Students wishing to take the course for credits will be asked to develop a small project with liner logic programming. Other modalities are possible, subject to agreement with one of the instructors.

Part 1: Alberto Momigliano. Introduction to linear logic (*4 hours, April 28,29*)

Lect 1 Introduction, motivation, applications [3].
Lect 2 Proof theory of linear logic [7].

Part 2: Carsten Schürmann. Linear Logical Programming (*12 hours*, May 2–6)

Lect 1 *Backward Chaining Linear Logic Programming.* We start with a gentle introduction to theory, and backtracking, followed by a tutorial introduction to the Celf system [6, 8]. To experience some programming practice we will focus on the blocks world, and program it.
Lect 2 *Forward Chaining Linear Logic Programming.* We will learn the basics of multiset-rewriting and its relationship to linear logic programming [9].
Lect 3 *Applications.* We will further exercise linear logic programming as a way of introducing some interesting test-cases.
Lect 4 *Concurrent Logic Programming.* We will study the concept of evidence in Celf, saturation, and confluence.
Lect 5 *Epistemic Logic Programming.* Certificates for authorization [4] and security.

Part 3: Marco Gaboardi. Semantics of linear logic (*8 hours*, May 9–11)

Lect 1 Phase semantics as a model for *provability*: definition, interpreting the formulae, applications. [5].
Lect 2 Game semantics as a model for *proofs*: definition, interpreting the proofs, applications [1].
Lect 3 Geometry of interaction as a model for *normalization*: definition, interpreting the reduction, applications [2].

References

1. S. Abramsky and R. Jagadeesan. Games and full completeness for multiplicative linear logic. *Journal of Symbolic Logic*, 59(2):543–574, 1994.
2. S. Abramsky and R. Jagadeesan. New foundations for the geometry of interaction. *Information and Computation*, 111(1):53–119, 1994. Extended version of the paper published in the Proceedings of LICS'92.
3. Roberto Di Cosmo and Dale Miller. Linear logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy.* Fall 2010 edition, 2010.

4. Deepak Garg, Lujo Bauer, Kevin D. Bowers, Frank Pfenning, and Michael K. Reiter. A linear logic of authorization and knowledge. In Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, editors, *ESORICS*, volume 4189 of *Lecture Notes in Computer Science*, pages 297–312. Springer, 2006.
5. J.-Y. Girard. Linear logic: Its syntax and semantics. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 1–42. Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.
6. Dale Miller. Overview of linear logic programming. In Thomas Ehrhard, Jean-Yves Girard, Paul Ruet, and Phil Scott, editors, *Linear Logic in Computer Science*, volume 316 of *London Mathematical Society Lecture Note*, pages 119–150. Cambridge University Press, 2004.
7. Frank Pfenning. Linear logic. Notes for a graduate course, `http://www.cs.cmu.edu/fp/courses/linear/handouts/linear.pdf`, 2002.
8. Anders Schack-Nielsen and Carsten Schürmann. Celf - a logical framework for deductive and concurrent systems (system description). In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *IJCAR*, volume 5195 of *Lecture Notes in Computer Science*, pages 320–326. Springer, 2008.
9. Robert J. Simmons and Frank Pfenning. Linear logical algorithms. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 336–347. Springer, 2008.