# Review of
# Proof, Language, and Interaction: Essays in Honour of Robin Milner*

Riccardo Pucella

Department of Computer Science
Cornell University

January 14, 2001

## Overview of Milner's work

The field of theoretical computer science has had a long tradition of providing wonderful edited books to honor various prominent members of the community. The idea behind such edited works is clear: celebrate the work of a particular researcher by having co-authors and colleagues write papers on subjects related to the researcher's body of work. When successful, such a project provides for a wonderful excursion through the work of a researcher, often highlighting an underlying coherence to that work.

On that account, the book "Proof, Language, and Interaction: Essays in the Honour of Robin Milner" (edited by Plotkin, Stirling and Tofte) is a success. In no small part this is due to Milner's span of work from theory to practice. It is hard to both briefly describe Milner's contributions and give an idea of the breadth of his effort. Two major tracks of research emerge. First, after some work with John McCarthy's AI group at Stanford, he developed LCF (specifically, Edinburgh LCF), a system for computer-assisted theorem proving based on Dana Scott's ideas on continuous partial functions for denotational semantics. Not only working on the implementation, Milner also worried about the semantic foundations [1]. LCF came with a programming language, Edinburgh ML, which evolved with Milner's help into Standard ML, a higher-order language that introduced many features now standard in advanced programming language, features such as polymorphism and type inference [4]. His second track of research involves concurrency. He invented CCS, the calculus of communicating systems [2]. Work on the semantics of CCS went from the traditional domain-theoretic approach to a new operational view of processes equality based on the notion of bisimulation. Subsequent work by Milner led to the development of other calculi, most notably the $\pi$-calculus, which included a powerful notion of mobility [3]. His later work in the area attempted to provide a unifying framework for comparing calculi, and unifying sequential and concurrent computation.

This range of interests is reflected in the table of contents of the book. The book splits across different subjects: *semantic foundations*, where we discover work aimed at understanding computation, both sequential and concurrent, *programming logic*, where we discover work derived from Milner's work on LCF, and aimed at understanding the role of formal mathematics, *programming languages*, where we discover Standard ML, and other languages based on Milner's idea about concurrent calculi, *concurrency*, where we discover work on Milner's early CCS approach and extensions, and *mobility*, where we discover work related to the $\pi$-calculus.

*G. Plotkin, C. Stirling, M. Tofte (eds), *Proof, Language, and Interaction: Essays in Honour of Robin Milner*, MIT Press, 2000, 722pp, ISBN 0262161885.

# Contents

12. **A Type-Theoretic Interpretation of Standard ML.**
Harper and Stone present the first interpretation of the full Standard ML language in type theory, by translating Standard ML into an explicitly typed $\lambda$-calculus; this allows the dynamic semantics to be defined on typed, rather than type-erased, programs.

13. **Unification and Polymorphism in Region Inference.**
Tofte and Birkedal present rules to perform region inference in a language with region-based memory management, using unification to allow for some amount of region-polymorphic recursion.

14. **The Foundations of Esterel.**
Berry gives an overview of Esterel, an imperative synchronous language, describing its underlying reactive model, its semantics and implementation in terms of sequential circuits, and issues related to verification.

15. **Pict: A Programming Language Based on the Pi-Calculus.**
Pierce and Turner describe Pict, a high-level programming language based on the $\pi$-calculus, in a way similar to the way ML is based on the $\lambda$-calculus; communication is the sole mechanism for computation.

IV. **CONCURRENCY**

16. **On the Star Height of Unary Regular Behaviours.**
Hirshfeld and Moller prove a conjecture of Milner, that the star height hierarchy of regular expressions over a unary alphabet is a genuine hierarchy when equivalence is taken to be bisimilarity; the corresponding question for regular languages is negative.

17. **Combining the Typed $\lambda$-Calculus with CCS.**
Ferreira, Hennessy and Jeffrey present a concurrent language based on a unification of CCS and the typed call-by-value $\lambda$-calculus, with the aim of defining a communicate-by-value concurrent language; the motivation is to analyze the behavior of CCS when the data communicated is taken from a non-trivial data space.

18. **Discrete Time Process Algebra with Silent Step.**
Baeten, Bergstra and Reniers discuss an extension of ACP (a process algebra similar to CCS) with discrete time and silent step $\tau$, of the kind found in CCS; three views of discrete time process algebra with silent step are presented.

19. **A Complete Axiom System for Finite-State Probabilistic Processes.**
Stark and Smolka give a sound and complete equational axiomatization of probabilistic bisimilarity for a probabilistic version of CCS, with binary summation of the form $E_p+ E'$ (meaning $E$ is chosen with probability $p$, $E'$ with probability $1 - p$).

V. **MOBILITY**

20. **A Calculus of Communicating Systems with Label Passing —- Ten Years After.**
Engberg and Nielsen describe a historical step in the development of the $\pi$-calculus from CCS by presenting ECCS, an extension of CCS that introduced a notion of channel passing.

21. **Trios in Concert.**
Parrow exhibits a fragment of the $\pi$-calculus, in the form of a subset of its terms, such that any $\pi$-calculus term is weakly equivalent to a term in the fragment; this highlights the root of the power of $\pi$-calculus.

22. **Concurrent Objects as Mobile Processes.**
Liu and Walker present an extension of the $\pi$-calculus suitable for the semantic definition of concurrent object-oriented languages; the calculus includes process abstraction as in the higher-order $\pi$-calculus and data other than channel names, but excludes higher-order interaction.

23. **$\lambda$-Calculus, Multiplicities, and the $\pi$-calculus.**
    Boudol and Laneve address the question of the semantics induced by $\lambda$-terms when they are encoded in the $\pi$-calculus; they show why the $\pi$-calculus is more discriminating than the $\lambda$-calculus, in the sense that two terms which are equivalent in the $\lambda$-calculus may be distinguished in an appropriate $\pi$-calculus context.

24. **Lazy Functions and Mobile Processes.**
    Sangiorgi examines the encoding of the lazy $\lambda$-calculus into the $\pi$-calculus; the encoding is used to derive a $\lambda$-model from the $\pi$-calculus processes; full abstraction of the model is obtained by strengthening the operational equivalence on $\lambda$-terms.

# Opinion

This volume is by no means an introductory book. Most of the articles are at the level of journal papers, and deal with technical issues. Those articles which are more expository or that provide historical perspective require a knowledge of the subject to be fully appreciated. Nevertheless, this volume should form an important part of any collection of work on semantics, programming languages or concurrency. The articles cover core subjects in those areas, and many present state-of-the-art techniques; for instance, the current trend in game semantics (cf. Abramsky's article), or the use of a type-theoretic internal language for programming language specification (cf. Harper and Stone's article). If anything, abstracts for the various articles would have been very helpful.

# References


[1] R. Milner. Fully abstract models of typed lambda-calculi. *Theoretical Computer Science*, 4:1–22, 1977.

[2] R. Milner. *A Calculus of Communicating Systems*. Number 92 in Lecture Notes in Computer Science. Springer-Verlag, 1980.

[3] R. Milner. *Communicating and Mobile Systems: The $\pi$-calculus*. Cambridge University Press, 1999.

[4] R. Milner, M. Tofte, R. Harper, and D. MacQueen. *The Definition of Standard ML (Revised)*. The MIT Press, Cambridge, Mass., 1997.