Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 265 (2010) 5–10

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

# Robin Milner's Work on Concurrency

Samson Abramsky[1]

*Oxford University Computing Laboratory*
*Wolfson Building, Parks Road, Oxford OX1 3QD, U.K.*

**Abstract**

We give a short appreciation of Robin Milner's seminal contributions to the theory of concurrency.

*Keywords:* Concurrency, process algebra, CCS, $\pi$-calculus, bigraphs.

## 1 Introduction

This brief note is based on the talk I was asked to give at a session honouring Robin's work which took place at MFPS in Ottawa, on May 8th 2010, just a few weeks after the sad news that he had passed away.

Bob Harper spoke in the same session on Robin's work on LCF and ML. I decided to focus on his work on concurrency. I believe that this was the work which meant the most to him — and which was the deepest of his many remarkable achievements.

In 1989 I wrote [2] that Robin's ideas had become part of the air we breathe in our scientific community. That is even more true today!

There is so much we have learnt, and can continue to learn, from Robin, over and above technical details of this or that formalism. I will try to articulate just a little of this in the brief remarks which follow.

## 2 1979: The First Revolution

Let us begin with some background.

[1] Email: samson@comlab.ox.ac.uk

[2] The context was a letter in support of Robin's nomination for a Turing award, which he of course received in 1991.



doi:10.1016/j.entcs.2010.08.002

The 1970's had seen an extensive development of domain theory and denotational semantics. The basic paradigm here is that programs are modelled as **functions**. This is fine for functional programming (*pace* sequentiality); less good for effects; but what about concurrency?

There was already a theory of concurrency, Petri's Net Theory. This contained some deep ideas, but was lacking some critical structural features. There were also a number of important programming ideas: Dijkstra's semaphores, Brinch Hansen's monitors, Hoare's critical regions etc., but no systematic and comprehensive theory.

At the same time, there were urgent technological imperatives: distributed systems, the Internet, etc. Who would meet the challenge?

*Enter CCS*

Robin's introduction of CCS transformed the field. The crucial ingredients seem to me the following:

- Processes are seen as a **mathematical structure** (rather than a machine model).
- In particular, processes are given an **algebraic structure**.

$$P \quad ::= \quad a.P \mid \mathbf{0} \mid P + Q \mid P \parallel Q \mid P \setminus a \mid P[S] \mid \ldots$$

  This has been enormously important: for the first time, processes could be studied **compositionally**.
- We knew what functions were already! But what are processes? CCS established the fundamental methodological point of studying them through their **observational behaviour** (defined in terms of labelled transition systems via SOS).
  This led inexorably in turn to notions of **observational equivalence**.
- In the train of these ideas came a host of technical developments: equational axiomatizations, Hennessy-Milner logic, algorithmics of bisimulation, and more.

CCS was not just a new calculus, but a new **paradigm** — that has played a central rôle in the subsequent development of our subject. It opened up the world of compositional behavioural modelling.

*MFPS 1989*

Many people would have rested on their laurels. Robin never did.

- Having revolutionized the field with his CCS book in 1980 [2], in large part stimulating the growth of a whole research community, including related developments such the extensive work on CSP (Brookes, Hoare, Roscoe et al.) and the Dutch school of process algebra (Bergstra, Klop et al.);
- Having led the effort to standardize ML and personally crafted, with Bob Harper, Mads Tofte et al. the formal definition and commentary on the language [5,9];
- Having written the masterly text *Communication and Concurrency* [3] in 1989, as a polished and definitive presentation of CCS:

"They might have said: 'He stopped here'." [3]

This is the opposite of what happened. Robin's talk at MFPS was one of the first occasions where he presented what was to become known as the **$\pi$-calculus**.

At that meeting he was re-energized, brimming with ideas. A whole new phase of the subject was about to begin!

*Robin's quest: the $\lambda$-calculus of concurrency*

The $\lambda$-calculus is an incredibly rich, concise, comprehensive theory of functions and functional computation.There are infinitely many (fruitful) extensions and variations, but the core calculus can be presented in a few lines — and there is (essentially) only one!

Could something of similar quality be achieved for concurrency?

For all the success of CCS and its variants, Robin realized — well before anyone else (and some still haven't!) — that they had not achieved this goal. As one telling and key example, the $\lambda$-calculus itself cannot be compositionally encoded in CCS in a satisfactory fashion.

With a remarkable leap of insight, Robin saw that one could use the notion of **name** as a building block to allow the expression of **mobility** of processes, and hence to open up a huge increase of expressive power in process calculi.

The importance of naming, unique and fresh names etc. had long been known to systems researchers; but no-one had imagined how this could be distilled into such an elegant and expressive theory.

## 3 The $\pi$-calculus

The $\pi$-calculus developed from the initial version developed by Robin with Joachim Parrow and David Walker [8], into a beautiful, concise, and expressive calculus.

The core calculus can indeed be conveyed in a single line:

$$P \quad ::= \quad x(y).P \mid \bar{x}y.P \mid \mathbf{0} \mid P \parallel Q \mid \nu x.P \mid !P \ldots$$

Again, what was created was not just a particular calculus, but a **paradigm**. It was to spawn whole sub-genres of 'mobile' and 'nominal' calculi.

Some key ingredients of the $\pi$-calculus:

- Structural congruence. This was Robin's transmutation of ideas from Berry and Boudol's Chemical Abstract Machine, and has proved an extremely useful idea which has been widely adopted.
- Names, freshness, scope extrusion. This led to a whole sub-paradigm of 'nominal calculi', in functional as well as process forms.

[3] Robin said this to me during MFPS 1989 — a rare occasion when I heard him consider what might have been, rather than focus intensely on what was.

- Behavioural types, which constrain dynamical process behaviour, rather than statically classifying values as in classical type theories.
- The ability to represent a huge range of computational phenomena, including higher-order and object-oriented computation, security protocols, biological modelling, business processes, and more.

The $\pi$-calculus proved to be even more hugely successful than CCS. Robin gave another polished and masterly presentation in his book *Communicating and Mobile Systems: the Pi-Calculus* in 1999 [6] .

Was he going to be satisfied now?

## 4 The Quest Continues

For all the beauty and the success of the $\pi$-calculus, Robin realized that it was not **the** $\lambda$-calculus of concurrency. There were too many possible variations and alternatives. [4]

With immense drive, energy and conviction Robin rewrote the script again. Now he sought, rather than a single unique, comprehensive **calculus**, to find the canonical, comprehensive structure at the **metalevel** (cf. rewriting systems).

This led to the work on **action calculi** and **control structures** [4,1]. This uncovered much fascinating static structure, but for a while seemed stalled on the issue of capturing the behavioural dynamics of systems at this level of generality.

Once again, Robin had other preoccupations for a time, as Head of Department of Cambridge. What next?

## 5 Bigraphs

Robin's last phase of work was on bigraphs, culminating in his last book, published in 2009, on *The Space and Motion of Communicating Agents* [7].

Once again, Robin recast the paradigm. Here are some salient features:

- Two kinds of linking structure are recognized, which are orthogonal to a surprising degree: one the kind of linking by naming introduced in the $\pi$-calculus, the other a **spatial structure**, as introduced by Cardelli and Gordon in the Ambient Calculus, and subsequently developed extensively as a sub-paradigm.
- The **static structure** is analyzed in terms of symmetric monoidal categories, building on the earlier work on action structures.
- There is a remarkable treatment of the dynamics, solving the problems which had stymied the earlier work, using ideas of **relative pushouts** in a highly original way — by no means "off-the-shelf" category theory!
- A natural and compelling graphical formalism.

[4] This is indeed an instance of one of the deepest issues of our subject: the 'Next 700 ...' syndrome.

- A major push towards tool support and applications, in Ubiquitous Computing and Systems Biology, led, encouraged and guided by Robin.

How important will this new paradigm be? It is too soon to tell; but I wouldn't bet against it ...

## 6 Some lessons we can learn from Robin

- **No Stone Tablets! No Disciples!**
  Having created a paradigm, he recast it and made it new, not once, but in three major phases. Always with a purpose, moving the subject to a higher level.
- **Follow through!**
  Four major books on concurrency: 1980, 1989, 1999, 2009. Led, inspired and encouraged a broad body of work, from theory to tool support and applications.
- **Be open to the work of others, and learn from them**.
  (Although in Robin's hands, the ideas were usually transformed in some way!)
  Key examples include: Structural Operational Semantics (Gordon Plotkin), bisimulation (David Park), structural congruence (Gérard Berry and Gérard Boudol), monoidal categories of processes (José Meseguer and Ugo Montanari), spatial structure (Luca Cardelli and Andy Gordon).
- Use **the right mathematics** to realize your scientific vision; don't tailor your approach to suit some preconceived mathematics you happen to like.
  The mathematical structures used by Robin in various phases of his work included domains, labelled transition systems and SOS, categories, and more.
- **Think it through**.
  Whenever anyone raised a question concerning why such-and-such feature of one of his calculi was the way it was, or suggested some possible alternative, it became clear that Robin had considered all of these issues, carefully and deeply.
  Robin was not given to lightning-fast rejoinders or intellectual pyrotechnics. Intellectually, he had the speed of the long-distance runner. [5] His ideas, too, have shown their staying power.

There are many more ...

## 7 A final word

Robin was **both** a great scientist, **and** a great human being. Our community has been fortunate indeed to have someone so inspiring in human as well as intellectual terms.

Thank You Robin.

[5] Yoram Hirshfeld, who spent some time working closely with Robin, once remarked something on these lines, which has stayed with me.

# References

[1] Mifsud, A., R. Milner and J. Power, **Control structures**, in: **Symposium on Logic in Computer Science (LiCS)**, Published by the IEEE Computer Society Press, 1995, p. 188.

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

[3] Milner, R., "Communication and Concurrency," Prentice Hall, 1989.

[4] Milner, R., **Action calculi, or syntactic action structures**, Mathematical Foundations of Computer Science 1993 (1993), pp. 105–121.

[5] Milner, R., "The Definition of Standard ML: revised," The MIT Press, 1997.

[6] Milner, R., "Communicating and Mobile Systems: the $\pi$-calculus," Cambridge Univ Pr, 1999.

[7] Milner, R., "The Space and Motion of Communicating Agents," Cambridge University Press, 2009.

[8] Milner, R., J. Parrow and D. Walker, **A calculus of mobile processes, I**, Information and Computation **100** (1992), pp. 1–40.

[9] Milner, R. and M. Tofte, "Commentary on Standard ML," MIT press, Cambridge, MA, 1991.