Noname manuscript No.
(will be inserted by the editor)

Stories About Calculations: Remembering Peter Landin

Tony Clark

the date of receipt and acceptance should be inserted later

Keywords Programming Languages, Theory of Computing

Abstract This article recalls Peter Landin as a PhD supervisor and introduces his final lecture notes.

Peter Landin was my PhD supervisor. I registered in 1988 for a part-time PhD while working for Marconi and met with Peter on Friday afternoons about once every couple of months from 88 to 93 and then sporadically until my graduation in 96. I had a rather idealized notion of Academia and Peter did not disappoint in terms of a dazzling performance at the chalk-face, together with a clear and personal approach to the subject and its history (including gossipy asides).

We would meet in Peter's room after lunch at Queen Mary College, University of London, which in those days was designated (by Peter) smoke-free, in the sense that the smokes were 'free'; we discussed at-length a subject of Peter's choosing and would typically repair to the bar, which amazingly was at the end of the Computer Science corridor, where Peter would continue the technical discussions. When I took up my first academic post at the University of Bradford in 94 I was dismayed to discover that not all departments have bars at the end of the corridor (and not all departments have people like Peter).

Peter had a consuming interest in Programming Languages and was both active and productive during the years I knew him. His approach is best described by John Reynolds: 'Peter Landin remarked long ago that the goal of his research was *to tell beautiful stories about computation*' Reynolds (1999). Furthermore, Peter aimed for *precision*, to be contrasted with formality, in story telling. In research and pedagogy Peter was interested in ways of communication and representation, informed by the essence of a computational issue; to *articulate precisely* was the key.

Peter is perhaps best known for discovering a connection and inventing a machine. The former, a correspondence between λ-calculi and programming languages, has had far-reaching consequences, the latter less so, possibly due to mis-understanding of the intended use of the SECD machine and due an emergence of a more 'abstract' approach in terms of semantic models based on relations.

Tony Clark
University of Middlesex, UK

The SECD machine was often viewed as a specific implementation mechanism for functional languages, however this lack of abstraction inherent in the SECD machine was seen, by Peter, as a virtue when the aim was to study the clunk and whir of evaluation mechanisms, for example see Danvy and Millikin (2008). Indeed, this aspect was core to Peter's approach to software (what did it *do*?, what can it *do*?, and, how can we make it *do* it?); descriptive tools that lack this feature would be argued as missing a trick.

In the late 80's, and at least until the early 00's, Peter was working on a system for describing program behaviour in a way that captured the essence of execution without over-specification. In discussions, he expressed frustration that the methods of the day provided a means for a relational discourse and thereby could not express sufficiently 'beautiful stories'.

While I was at Bradford he sent me a collection of notes, dated 1997, running to some 40 pages, on *calculations* that he was developing for publication, possibly as a book. The terms *calculation* and *calculational-step* appeared in a workshop presentation Landin (1997), and seem to have been coined around that time. Peter continued to work on calculations and, after I moved to King's College, University of London in 2000, I invited him to give a seminar where he presented them in overview. As far as I know these notes have not appeared elsewhere and are included in this volume as *Calculations*. They take the form of an 'interlude' which seems to be the basis for a presentation on the motivation for calculations, and a systematic definition of calculations and their relation to program design – a tutorial. The tutorial goes to great lengths to cover the key features of calculations from all angles and therefore has been edited down. In all other respects, the words and figures are those in Peter's notes.

From his notes Peter listed his motivations as including: 'being persuasive about the intuitions that guide small design choices; to explore the elementary concepts of computing without mentioning programs [...] undermined by the disappearance and diminished pragmatic relevance of the machine; to explain something full of implications regarding formulas/variables/algebra without actually resorting to them; to present, paradoxically, a wholly textual, picture free explanation of a highly pictorial concept'.

The notes are incomplete and trail off towards the end. There are indications of several sections yet to be completed: *77 programs for the same calculation*; *Stack calculations*; *Nested where-terms for calculations*; *Sub-calculations and composability*; *Synthesizing a program by unifying calculations*; *DFDs as calculations*; *Threads – finding them and finding opportunities for them.*

My recollection of Peter is that he was funny, generous, haughty, egalitarian, shy, uncertain, curious and ferociously intellectual. But, boy what *beautiful* stories!

References

Olivier Danvy and Kevin Millikin. A rational deconstruction of landin's SECD machine with the J operator. *Logical Methods in Computer Science*, 4(4), 2008.

Peter Landin. Histories of discoveries of continuations: Belles-lettres with equivocal tenses. In *ACM SIGPLAN Workshop on Continuations, number NS-96-13 in BRICS Notes Series*, 1997.

John C. Reynolds. *Theories of programming languages.* Cambridge University Press, New York, NY, USA, 1999. ISBN 0-521-59414-6.