Fundamental Concepts of Programming Languages by Christopher Strachey

A review by Manikanta Reddy

September 22, 2015

Christopher Strachey's paper on Fundamental Concepts in Programming Languages was written in the autumn of 1967, based on his lectures given t an International School in Computer Programming, held in Copenhagen. He intended that his paper be published in the proceedings of the School, but the proceedings never materialized, and it remained an unpublished pre-print for more than thirty years. Its a highly influential paper which has been widely circulated among the community of computer scientists.

The paper begins with a philosphical remarks on semantic issues we have in design of programming languages. He emphasis that a formal jargon is necessary as core concepts are so vague.

"The advantages of rigour lie, not surprisingly, almost wholly with those who require construction rules."

He then proceeds to give a clear and analytical description of many of his insights into programming language design and formal semantics, covering many topics.

He then proceeds to propose a semantics, initially refered to as 'mathematical semantics' or simply 'Strachey Semantics', which was later developed to be known as 'Denotational Semantics'. After the untimely death of Strachey, this paper has been exploited in various textbooks and papers by numerous authors and they have in turn become an established basis for our understanding of issues.

Thus this paper has indeed a high degree of historical interest and relevance. The paper can be recommended to anyone for its simplistic approach backed by clearly formulated concepts. However I would recommend you proceed with caution. To highlight one, most of the illustrations are in a very lesser known Language called CPL, a language developed by Strachey himself and colleagues. It was supposed to be a successor of Algol, but was never fully made. But fortunately CPL is understandable.

Another possible source of confusion is that the semantic functions L and R for expressions, introduced in Sect. 3.3.2, do not explicitly take any environment arguments. However, Strachey clearly explains that "we speak of evaluating an expression in an environment (or sometimes relative to an environment) which provides the values of component [identifiers]" (Sect. 3.2.2), and moreover, the representation of functions in Sect. 3.5.2 makes explicit reference to such an environment.

He provides an extremely broad survey of core issues in programming language design that provided much of the terminology we use today, including definitions of the kinds of polymorphism and the kinds of expressions we see in programming languages.

In his later works, Strachey was careful to distinguish between the domains of ‘denotable’ and ‘storable’ values, rather than lumping them together as ‘R-values’; the conceptual model, may be a bit misleading, in that a constant identifier would normally be mapped directly to a value by the environment, without any involvement of the abstract store, and moreover, a numeral would directly denote an abstract number.

The reader should not be disconcerted by Strachey’s operational interpretation of λ-expressions: the paper was written a full two years before Scott provided a model for the λ-calculus and established the domain theory that forms the mathematical foundations of denotational semantics .

Strachey writes also (at the very end of the paper, when comparing his proposal for a semantic method to previous proposals)

[:::] the ultimative machine required (and all methods of describing semantics come to a machine ultimately) is in no way specialised. Its only requirement is that it should be able to evaluate pure λ -expressions. [:::]

There is however no indication that the intended machine should be deterministic. In conclusion, Strachey’s paper is still well worth reading carefully, some 48 years after he wrote it. Enjoy it!