

# INFORMATION PROCESSING BY AMBIGUOUS FORMAL ENTITIES

*Corrado Böhm*

Dip.to di Scienze dell'Informazione, Università di Roma "La Sapienza",  
via Salaria, 113 - 00198 Roma - Italia  
Email: bohm@dsi.uniroma1.it

## ABSTRACT

Formal entities like combinators, which can ambiguously act as operands as well as operators, are introduced, discussed and represented by formal expression or by tree-like graphs, in order to give an intuition of a number of significant facts. We try to develop a computational model different from a Turing machine model or from actual computers [9]. They share the feature that the essential operations needed to benefit of their existence are first some kind of coding of structured data and secondly the writing down some algorithms to process the coded data [8]. All the criticism done by J. Backus [6] applies here and it motivated the development of Functional Programming, very related to Combinatory logic [7].

Theory is knowledge of the truth; a formalism is based on a set of signs or symbols (concrete tokens) whose assemblages obey very precise formation rules and whose features mimic those of more abstract concepts; constructivity, today, means computability or representability inside a computer. Once a phenomenon is theoretically well known, the same contents of information gained by an experiment, can equally be provided by the result of a computation, obtained by feeding into a computer some (simulation) program and some data (representing the initial state or environment of the experiment), and leaving the computer to run autonomously. In such a case we may say that the program is a process explication of the phenomenon. Since today a program is mainly written in a high level language, very different kinds of data or data types are available, like booleans (or truth values), integers, (approximate) real numbers and more structured data as: arrays of..., linear sequences of..., trees of..., where the dots may be replaced by any one of the previous data types. Data and data types are certainly formal entities, but they do not exhaust the latter category of objects.

About eighty years ago M. Schönfinkel [1] defined a family of entities (combinators) whose main property was that, combined together, they could act as operators as well as operands. This dichotomy was solved in an impartial way, depending only on their mutual position. The theory of combinators was discovered independently also by H.B.Curry [2,3] in the Thirties and most of the work on the  $\lambda$ -calculus invented by A. Church [4] may be viewed as a contribution to the theory of combinators. The ambiguity operator-operand is exactly what we wish to exploit systematically, resuming and generalizing the results obtained by Church. Our proposal is to reintroduce ambiguity up to a certain extent, following a principle which has always been successful each time it has been introduced in computer science (and perhaps not exclusively): the "lazyness" principle (see lazy evaluation lazy compilers, etc.). To apply lazyness means to delay the decisions until the last moment. We propose for instance to delay the decision of what is the meaning of a symbol to the moment where that symbol is written down together with other symbols.

We show that defining correctly data systems as free algebras enables to derive automatically from their recursive definitions all the algorithms needed to transform such data into the wished results. To every datum is assigned in a precise way a meaning of how it will act as operator such that a terrific gain in expressiveness is achieved. The generalization of the operators to any sort of data is at hand, so that different yet consistent meanings can be safely assigned to the same combinator. Concepts like polymorphism and polysemantics become usable in a natural way. Some new

analogies rise automatically, for example considering finite sequences of objects as natural numbers decorated with such objects, or even some unifications of distinct data structures become harmless, like natural numbers and strings on a singleton alphabet.

Combinators model also in a natural way transformations where each operator acts only its immediate neighbours and not instantaneously, in harmony with the physical laws, so that they are good candidates for developing a new kind of hardware such as neuromorphic hardware. It is difficult to foresee the impact that a combinatory way of thinking will have on neuroscientists in general. We present the last developments of a doctrine of the human mind (whose seeds were sown 100 years ago to provide simple bases of the logical thought) with the hope to point out an alternative to process description through states transitions.

**Keywords:** Lambda-calculus, combinatorial thinking, mind theory.

## References

- [1] Schönfinkel M. (1924) Über die Bausteine der mathematischen Logik. *Mathematische Annalen* **92**: 305-316.
- [2] Curry H.B., Feys R. (1958) *Combinatory Logic*, Vol. 1, North-Holland.
- [3] Curry H.B., Hindley, J.R., Seldin, J.P. (1972) *Combinatory Logic*, Vol. 2, North-Holland.
- [4] Church A. (1941) The calculi of  $\lambda$ -conversion, *Annals of Mathematical Studies* Vol. 6, Princeton University Press, Princeton.
- [5] Böhm C., Berarducci A. (1985) Automatic Synthesis of Typed Lambda-Programs on Term Algebras, *Theoretical Computer Science* **39**: 135-154.
- [6] Backus J. (1978) Can programming be liberated from the Von Neumann style? A functional style and its algebra of programs, *CACM* **21**: 613-641.
- [7] Böhm C. (1982) Combinatory foundation of functional programming, ACM Symposium on Lisp and Functional Programming (16-19 August 1982, Pittsburgh), pp. 29-36.
- [8] Böhm C. (1999), Fixed point equations inside the algebra of normal forms, *Fundamenta Informaticae* **37**: 329-342..
- [9] Böhm C., Intrigila B. (1994) The Ant-Lion Paradigm for Strong Normalization, *Information and Computation* **114**: 30-49.