Philology of Programming Languages

Baptiste Mélès*[1]

[1]Laboratoire d'Histoire des Sciences et de Philosophie - Archives Henri Poincaré (LHSP) – CNRS : UMR7117, Université Nancy II – 91, avenue de la Libération BP 454. F-54001 NANCY Cedex, France

Abstract

Abstract programming languages (Turing machines, lambda calculus, PCF...) can be fruitfully used to highlight many essential properties of "real-life" programming languages (such as assembly, C, Perl or Java), but they also drop some of their interesting properties. "Real-life" programming languages have many characteristics nobody would ever want in a good formal language.
1) Their syntax has many irregularities (in C, "void" functions do not need to be declared as such).
2) The syntax is widely redundant: they often have as well "for" as "while" loops, and other signs of "syntactical sugar", against all principle of economy.
3) Their abstract and complete formal definition (as in BNF) usually comes long after their current use (e.g. with ANSI C and XHTML).
4) They even have historical residues: C-like syntax is used in many younger languages, such as C++, Java, Perl and JavaScript; the obsolete "register" keyword in C still belongs to the language.
These properties make them similar to natural languages. We will try to show some examples of philological concepts which can be applied to the concrete study of programming languages. Just like Chomsky described English as a formal language, we intend to describe C as a natural language.

*Speaker