From LISP and ALGOL 60 to Interpreters and Abstract Machines

1 Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I

```
@article{m:lisp,
   author = {McCarthy, J.},
   title = {Recursive Functions of Symbolic Expressions and
            Their Computation by Machine, Part I},
   journal = {Communications of the ACM},
   volume = {3},
   number = {4},
   pages = {184--195},
   year = 1960
 }
```

Summary: McCarthy introduces LISP. LISP deviates from other contemporary languages such as ALGOL as its design revolves around the idea that partial recursive functions of symbolic expressions can be an effective programming language. McCarthy gives meaning to LISP through a meta-circular "defunctionalized" interpreter and discusses the implementation of LISP's runtime.

Evaluation: This is a seminal paper for multiple reasons. First, it introduces a new programming model that affected the design of programming languages in the decades to come. Second, LISP pioneers many ideas that have evolved nowadays into staple features of most programming languages such as anonymous first-class functions (closures), reflection and garbage collection. Third, the way McCarthy presents LISP foreshadows the formal definition of programming languages with abstract syntax and meta-circular interpreters.

2 Revised Report on the Algorithmic Language ALGOL 60

```
@article{algol60,
```

```
 author = {Backus, J. W. and Bauer, F. L. and Green, J. and Katz, C.
and McCarthy, J.
and Perlis, A. J. and Rutishauser, H. and Samelson, K. and Vauquois,
B. and Wegstein, J. H.
 and van Wijngaarden, A. and Woodger, M.},
 title = {Revised Report on the Algorithm Language ALGOL 60},
 journal = {Communications of the ACM},
 volume = {6},
 number = {1},
 pages = {1--17},
 year = 1963
}
```

Summary: This report specifies a standard for ALGOL 60. The standard consists of two parts: the precise and formal description of the concrete syntax of ALGOL 60, and the informal description of the evaluation of ALGOL 60 programs.

Evaluation: The report standardizes a seminal language. ALGOL introduced many important features that future programming languages adopted such as call-by-name, local definitions and local effects. Outside that, the report is one of the first efforts to specify a programming language as a precise and formal artifact. In particular, the report uses a BNF grammar to unambiguously describe the syntax of ALGOL. Even though the description of the semantics of the language is not equally rigorous, it is also innovative in that it is abstract (meaning not tied to a particular implementation). For example it explains function application in terms of substitution of the arguments for the formal parameters inside the left hand-side of the definition of a function.

3 A Correspondence Between ALGOL 60 and Church's Lambda-Notation: Part I

```
@article{l:algol60-lambda,
 author = {Landin, P. J.},
 title = {Correspondence Between ALGOL 60 and Church's Lambda-notation:
Part I},
 journal = {Communications of the ACM},
 volume = {8},
 number = {2},
 pages = {89--101},
 year = 1965
}
```

Summary: Landin explains how one can "compile" a real programming language such as ALGOL to a minimal but canonical model language. Landin describes in detail and formally how each piece of syntax of ALGOL has an encoding in the language of applicative structures.

Evaluation: This paper presents a methodology for analyzing programming languages that has shaped research in programming languages. The fact that today we

study programming languages by reducing them down to small but well-defined formal systems is largely due to Landin's efforts to analyze ALGOL.

4 A Formal Description of a Subset of ALGOL

```
@proceedings{m:algol6-formal,
 author = {McCarthy, J.},
 title = {A Formal Description of a Subset of ALGOL},
 booktitle = {Proceedings of the IFIP Working Conference on Formal
Language Description Languages},
 pages = {1--12},
 year = 1966
}
```

Summary: McCarty gives meaning to core ALGOL programs (call-by-value with mutation but no control operators) by compiling them to to mutually recursive schemas. As a result he obtains a meta-circular LISP interpreter for core ALGOL.

Evaluation: This is the first work that uses meta-circular interpreters to explain a non-trivial language, dubbed the defining language, using another language, dubbed the defined language. Furthermore the paper demonstrates that such an approach is possible even when the defining and defined languages seem far apart syntactically and semantically. The explanation by interpretation method is not popular in research nowadays but has had immense impact on the pedagogy of programming languages.