From Interpreters and Abstract Machines to the λ Insight

1 The Mechanical Evaluation of Expressions

```
@article{ pl:secd,
 author={Landin, P. J.},
 title={The Mechanical Evaluation of Expressions},
 journal={Computer Journal},
 volume=6,
 issue=4,
 pages={308--320},
 year=1964
}
```

Summary: Landin introduces the language of applicative structures as a small but abstract language that can naturally express various forms of calculations such as calculations with numbers, booleans and lists. Landin gives meaning to programs in the language through their evaluation by an abstract and formally defined machine.

Evaluation: This paper introduces numerous novel and important ideas: 1) it makes an explicit syntactic connection between the λ-calculus and programming languages; 2) it suggests the use of abstract syntax as a way to sidestep cosmetic superficial differences between the syntaxes of different languages; 3) it pioneers the formal definition of programming languages with the innovation of abstract machines that manipulate the abstract syntax of programs; 4) it coins the term "syntactic sugar" to explain how we can grow a core language with more programmer-friendly forms that are in reality compositions of the core language. Overall, Landin suggests the use of formal small model languages as the means to understand real languages.

2 The Next 700 Programming Languages

```
@article{ pl:iswim,
 author={Landin, P. J.}
 title={The Next 700 Programming Languages},
 journal={Communications of the ACM},
 volume=9,
```

```
 issue=3,
 pages={157--166},
 year=1966
}
```

Summary: Landin builds on his previous work on the language of applicative structures to suggest a systematic way to design new languages. In particular, he argues that each language consists of two distinct sets of features that are often conflated: a set of features that address the basic computational requirements of any language and a set of features that are specific to the domain of the problem that the language is intended to help with. Furthermore, he identifies the language of applicative structure as the minimal language that can capture the computational aspect of any language. The combination of this minimal core with domain-specific plugins gives rise to a family of languages dubbed ISWIM. Landin claims ISWIM helps programming language designers avoid the pitfalls of various idiosyncrasies of real languages and helps them focus on the problem of designing for the domain the language targets.

Evaluation: This paper together with "The Mechanical Evaluation of Expressions" has had significant impact on the way programming languages researchers analyze and design languages. The analysis and design of programming languages using a formally specified and studied core model is nowadays the norm.