

Toward A Browsable Design Space and Integrated Pedagogical Environment

Fritz Ruehr, Willamette University

Abstract

We describe two related, nascent projects motivated by pedagogical concerns: the first is a structured repository of description and program fragments constituting a multi-dimensional, browsable design space for the specification and implementation of small teaching languages. The second is an integrated *pedagogical* environment (by contrast with the usual IDE, or integrated *development* environment), optimized for the presentation and study of short programs, with an emphasis on comparative analysis of alternative designs. (A related third project updates and extends the traditional formal languages syllabus to address issues in language design, semantics and logic.) In our motivations and approach we presume that example programs will be relatively concise—we will use Haskell, but other functional languages could serve as well.

Many textbooks and monographs (e.g., those of Stoy, Schmidt or Friedman et al.) use a series of “little languages” to introduce progressively more complex ideas in syntax and semantics. We propose to develop a relatively complete set of language examples in this vein, structured along several dimensions, as well as a framework for storing and accessing these examples which will facilitate incremental comparison. The main dimensions of this design space are:

- a progression of *language features*, from trivial numerals and algebraic expressions, through the incorporation of imperative variables, to loops, abstractions and types;
- a variety of *language components*, including concrete and abstract syntax, denotational and operational semantics, and various tools such as parsers, calculators and translators;
- a range of *descriptive techniques*, from informal English to formal specifications and Haskell implementations, the latter including a number of alternative representation strategies;
- a variety of *semantic domains*, from numeric and boolean algebras through regular string languages and computable functions.

These dimensions are not orthogonal: increases in feature complexity often require new kinds of language components and more sophisticated and varied descriptive techniques. Significant issues in the design of the design space concern representations which make incremental changes in components efficient and which allow flexible recombination of alternative choices.

Our proposed *Haskell Integrated Pedagogical Environment* (HIPE) is intended mainly as a browser for the design space described above, but also as an independently useful tool for teaching functional programming. We hope to leverage the conciseness of functional programs and serve the particular needs of pedagogy by facilitating dynamic review of various programming techniques and language features. Toward these ends we hope to incorporate side-by-side comparisons of code fragments, metrics and run-time properties; flexible code-focusing techniques; and expedited selection of design and style alternatives. Significant issues in the design of the environment are the degree of integration of (and thus dependence on) outside tools and an appropriately light-weight modules overlay.