Mixed Week 12

Christian Nassif-Haynes

6 November 2013

Tutorial Exercises

1. (a) Domain-Specific Languages (DSLs) are typically more succinct. Because they describe a single domain, fewer and more specific language constructs are needed.
   (b) DSLs allow solutions to be expressed at a level of abstraction most appropriate to the task at hand.
2. (a) In principle, an arbitrary number of domains could be conceived. It is impractical to design a DSL for all of them. Even if domains were defined sensibly, designing and implementing a language takes time and money. The pro's and con's should be evaluated on a case-by-base (or domain-by-domain) basis.
   (b) It is impractical for employees to constantly learn new languages and undesirable for employers to pay for labour involving an esoteric skill.
3. Internal DSLs are implemented within an existing programming language. They may add domain-specific elements such as datatypes, functions and so on. On the other hand, external DSLs are languages in their own right. Examples include LaTeX, MATLAB and SQL.

   Things to consider when deciding between external and internal DSLs include:

   - How closely the syntax of the language needs to follow that used by domain experts.
   - How important it is for errors messages to be relevant to the domain.
   - Whether or not domain-specific optimisations and validation must be applied.
   - Whether or not the resources (e.g. time and money) are available for designing and implementing the language externally.
4. Using a DSL to implement a func332 has been useful. Writing a compiler for func332 in a general-purpose programming language (GPL) is a large task, considering every stage involved in compilation. As well, the code written in Kiama was more idiomatic than could be expected of code written in a GPL.

   An interesting feature of Kiama were the operators used during syntactic analysis (`^^`, `<~`, etc). They had the advantage of giving the programmer more power over the way in which the language is parsed. However, the syntax is not as conventional (that is, well-known) as, say, extended Backus–Naur form.

   The feature of Scala which makes it particularly well-suited to implementing a library like Kiama is pattern matching, which was useful during syntactic analysis when matching syntactic constituents.