Homework 2: Domain Specific Languages Embedded in Haskell

Due: problem 1, October 5, 1999; problem 2, October 28, 1999.

In this group homework you will design, implement, and demonstrate a domain specific language using Haskell.

Don't hesitate to contact the staff if you are not clear about what to do.

1. (80 points) This problem is a preliminary to the main problem. In this part your group will (a) choose a domain and (b) sketch out the design of your domain specific language, recording it in a preliminary language reference manual.

   The domain for the domain specific language, should be something your group understands. It should not be too ambitious, but it should be something for which many different programs in the selected domain can be written. Some possible domains include the following: statistics manipulation, active HTML forms, string manipulation (e.g., for mass-market mailings), medical information systems, legal information systems, programming with directed graphs, neural networks, genetic programming, motion planning in robotics, pattern recognition, business data processing, scientific data reduction, astronomical image processing, control scientific instruments, numerical analysis, or data conversion programs (like XDR). You can choose your own domain, however, it should be both specific and not something that was covered in one of the two papers we read about this topic.

   Recall that in the domain specific language, you are designing a notation for other people to write programs in this domain. The problem is not to write one or two such programs yourself!

   The sketch of your language should include the following at a minimum:

   - A list of the primitive computational elements (datatypes, functions, constants) that your providing in the language to help write programs for the domain. For each function, include at least:
     - a description of any arguments the need to be passed to the function,
     - any properties that must hold among these arguments,
     - the meaning of the function when it is called with such arguments (try to describe the meaning in terms of the meaning of the arguments),
     - a small example of its use.
   - A list of the forms that you provide for combining the computational elements (these may play the role of statements in your language). For each such form, include at least:
     - a description of any subforms,
     - any properties that must hold among these subforms,
     - the meaning of the form when it is used with such subforms,
     - examples of its use.
   - A list of the mechanisms, if any, that you provide for abstraction, (these may play the role of declarations and parameters in your language). For each such mechanism, include at least:
     - a description of any subparts,
     - any properties that must hold among these subparts,
     - the meaning of the mechanism when it is used with such subparts,
     - examples of its use.

2. (160 points) In this, the main problem, you'll (a) implement your language that your group designed in the first problem, (b) use your language in several examples, and (c) revise your language reference manual.

   The implementation is to be done as an embedded domain specific language in Haskell. That is the types that you designed should be encoded as Haskell types, the functions encoded as Haskell functions, etc. Note that while the user's of your language need not be experts at Haskell, you are expected to demonstrate your expertise by making the notation as helpful as possible for them.

   The example should show a fairly nontrivial use of your language; something about half page or a page or so. Each person in your group should do one example, and each of the examples should be different. It's best if each of the examples can show a different aspect of the use of your language in programming in the domain.

   For this part, hand in an introduction to your language that combines a brief commentary with the examples of each of you produced, test showing that your example programs work with your implementation, the revised reference manual, and the source code for your implementation. Be sure to include a statement from the group leader certifying who participated, and also an evaluation of your group process as described in the course policies and procedures document.