1 Project Overview

In addition to reading about languages other people have designed, in this class you will design and build your own Embedded Domain-Specific Language (EDSL) in Haskell using Haskell’s quasi-quotation library. This project has a number of parts:

• In the first part (which you have already submitted), you wrote a design document describing and motivating your language. **Due October 15.**

• In the second part (which you have already submitted), you designed a syntax for your language documented as a grammar, defined an abstract syntax tree data structure to represent your program texts, and wrote a parser to map from program texts to abstract syntax trees. **Due October 29.**

• In the third part (which you have already submitted), you wrote a translator that maps abstract syntax trees for your language into data structures representing Haskell code. You connected these pieces together using Haskell’s quasi-quotation library. The quasi-quotation library allows end users to write programs in your language and have the Haskell compiler invoke your parser and translator to generate, compile, and execute the Haskell code providing the implementation for your language. You wrote any necessary “run-time” system code as a Haskell library that links with the generated code. The Hackage web site ([http://hackage.haskell.org](http://hackage.haskell.org)) has a large number of Haskell libraries that you were able to use in your implementation. The quasi-quotation framework is quite flexible in that you can use it to generate multiple interpretations for programs in your language (like the parser and pretty printing semantics for PADS). **Due November 21.**

• In the (almost) last week of class, you will present your language design to the class, explaining your design decisions and evaluating your design using the same criteria we’ve been applying to the DSL papers we’ve been reading for class. **Due December 1 and 3; specific dates for each DSL to be determined in class on November 24.**

• Finally, in lieu of a final exam, you will turn in a paper explaining your design and evaluating it. **Due December 8.**

You may choose to work in groups on this project. The scope of the project should be commensurate with the size of the team. Project assignments should be submitted and will be evaluated as a team. (That is, the entire team should submit only one response to each of the project parts and the entire team will receive the same grade.) Students in COMP 150PP may chose to do a joint project as long as it satisfies the project requirements of both courses. If you have any questions, please feel free to ask for clarification or input.
2 Paper

For this part of the project, you need to write a paper motivating, explaining, and evaluating your design. In writing your paper, you should strive to emulate the papers that we read in class, both in writing style and in substance. To that end, you should follow the instructions in the PLDI 2015 Call for Papers in formatting your paper as if it were a submission to the conference. Instructions are available from the conference web site: http://conf.researchr.org/track/pldi2015/pldi2015-papers#Instructions-for-Authors. Your paper should be no longer than the length specified in the PLDI Call for Papers, but it can be shorter.

For group projects, each member of the team must participate in writing the paper in roughly equal proportions. Note though that this requirement does not mean each person needs to be entirely responsible for 1/Team-Size fraction of the paper.

Your paper should address at least the following points:

1. What is the domain of your language?
2. Who are its intended users? What skills do they have and what kind of programming tasks do they need to accomplish?
3. What goals are you trying to accomplish with your language? (For example, the designers of ESP wanted to make it easier to develop correct, modular code for firmware while not paying too much of a performance cost.)
4. Why are existing languages ill-suited to the task?
5. Describe the features your language provides and explain why this set of features is appropriate.
6. Show at least one sample program in your language and explain what it does.
7. Describe how you implemented the language.
8. Explain what if any services are provided in a “run-time system.”
9. Describe any domain-specific type-checking, tools, or libraries that might be useful for future versions of your language.
10. Describe how you have (or would have if given more time) evaluated your language.
11. Discuss what you would do next on your language if you were going to continue working on it.

You should submit your slides using provide paper.