Generating Software from Specifications WS 2013/14 - Assignment 6

Published: Nov 25, 2013 - Turn in until Dec 4, at 12h, 2013

What to turn in: see Assignment 1

Exercise 17 (Use definition table operations systematically)

The directory `blatt6/PDLTask` contains a file `StatementsPDL.fw`. It is a specification of the structuring phase of Exercise 14 in Assignment 5. Extend the specification by solutions of the following tasks. Specify basic name analysis, and use PDL as described in Chapter 4. Work explicitly along the design steps given on slide GSS-4.4. Test the processor with suitable test cases for each subtask.

1. Use the basic name analysis module to bind all identifier occurrences to keys applying Algol-like scope rules (see GSS-4.1a). Test the generated processor.
2. At each defining occurrence output its line number; at each using occurrence output its line number and the line number of the defining occurrence.
3. Report a message if an entity is defined more than once. Give that message at all offending definitions.
4. Report a message if a use occurs before its definition.
5. At at every occurrence of any identifier, except the first one, output the line number of the previous occurrence. Definitions and uses are treated in the same way; it is irrelevant which entities are referenced.
6. Implement the "Do it once" pattern of slide GSS-4.5, and apply it to uses and definitions of variables, such that for each variable at most one occurrence of either kind is printed.

Exercise 18 (Work on your project)

Make sure that you have decided for which task you want to design and implement a domain specific language. (See Exercise 15) Then the next step is to systematically elaborate the requirements for your language and its translation:

First, discuss with you co-designers what shall be expressable in your language. Write down examples in some informal notation until you are convinced that the set of your examples cover all aspects of the intended language. Do not yet work on the syntax specification before you have completed this step. Keep all these examples for the next steps.

Second, describe what you want the DSL-implementation to generate from input as elaborated in the previous step. Make sure that you have considered all aspects of the intended language, and that you know in principal how that translation can be done. Take a non-trivial example, translate it by hand into the intended output, and check whether it will fulfill its purpose. In case that it turns out, that the translation task contains subtasks which are too complex or inadequate for the project, modify your idieas of the DSL now.

Consider the results of these two steps as the written requirements of your project. Update them as far as necessary when you proceed with your project.

Hand this project description in.

Do not forget to take notes on what you did, what you learned, and which problems you encountered, and turn them in.