Generating Software from Specifications WS 2013/14 - Assignment 2

Published: Oct 23 -- Turn in until Oct 29 at 12h

What to turn in: see Assignment 1

Exercise 3 (Explore Eli commands)

The wrapper specification of Assignment 1 is used to explore the facilities offered by Eli commands.

First prepare your working environment. For working effectively with Eli, it is recommended to keep the following four windows open and accessible on the screen:

- A browser window, that schows Eli's online documentation.
- A terminal window, that is used to execute commands in the working directory where the specification files are.
- An editor window used to modify the main specification file, i.e. Wrapper.fw.
- A terminal window, in which Eli is running.

It is recommended not to close any of these windows, and not to stop Eli running while working with Eli.

Those who are new to Eli or need to refresh their Eli knowledge find an introduction in Eli's "Guide for New Eli Users". Find in the Eli documentation the description of "Products and Parameters" and be prepared to consult it when working on the following steps:

1. Call Eli in your Eli window and reset the cache. Execute the command "loglevel=4"; it lets Eli report all single steps it executes. Do not terminate Eli if not necessary.
2. Derive warnings from the construction of the executable wrapper processor.
3. Derive the executable wrapper processor, and store it in your working directory. (You may edit the previous command.) Which operations did Eli execute now, compared to those in the previous step? Explain.
4. Introduce an error in a specification fragment, e.g. in the concrete syntax specified in the fragment named Wrapper.con. Then derive warnings again.
5. Modify the previous command such that the warnings are stored in a file in your working directory.
6. Derive help instead of warning, follow the error message, correct the error, and derive warning again.
7. Extract the executable wrapper processor to your working directory, and call it there (outside of Eli) with a suitable input file.
8. Create a sub-directory called "src", ask Eli to derive the source of the wrapper processor and to put it into that directory.
9. cd into that directory and find out how many files Eli generated (Unix command: ls | wc).
10. Have a look into the file "Makefile". What is described there?
11. To make the executable wrapper generator: just say "make".
12. Execute the thus compiled processor with suitable input.
13. Look up Eli's "run" command in the documentation. Then use that command to execute the wrapper generator on your input file. Where do you find the files generated by the wrapper generator?
14. Introduce an error into your input and re-execute the previous command.
15. Explore further commands described in "Products and Parameters".
16. Leave Eli by "ctrl-D"; and clean up your working directory.

Do not forget to take notes on what you did, what you learned, and which problems you encountered, and turn them in (see Assignment 1).

Exercise 4 (Explore Token Specifications)

The file `TryScan.fw` contains a specification of a very simple processor which accepts sequences of tokens separated by commas. Copy it into a fresh directory, setup your working environment, and use the file to explore how tokens are specified. Be prepared to consult the document on "Lexical Analysis" when working on the following:

1. Create a processor and execute it on a suitable input.
2. A function "myToken" is attached to each token specification. Check what it outputs and how it is implemented.
3. Prepare an input that causes the processor to report errors. Try to produce two different messages of type ERROR.
4. Delete the connection to "myToken" from the token specification. Lookup the parameter "+printtokens", create the processor with it, execute it, and understand its output.
5. Use Eli’s monitoring tool Noosa (instead of the "+printtokens" parameter) to inspect the results of lexical analysis (see "Monitoring a Processor’s Execution" in the Eli documentation):

   ```
   TryScan.fw+monitor+arg=(in):mon
   ```

   will open a Noosa window. When you execute "Run" the input is shown in the upper window. Then select some tokens, and examine the tokens in the lower window.
6. Introduce new tokens to be accepted by the processor, e.g. for days of the week, time in hours and minutes, or car plates. Explore how to design regular expressions for them. Consult Eli’s manual on "Lexical Analysis" for further information. Test whether the scanner accepts and rejects tokens as intended.

Do not forget to take notes on what you did, what you learned, and which problems you encountered, and turn them in (see Assignment 1).

Exercise 5 (Homework: CFG Design)

This exercise is intended to recover your knowledge on the design of context-free grammars, as you may have learned in the course on "Programming Languages and Compilers" (PLaC).

1. Repeat slides 3.1 to 3.4f of Chapter 3 in PLaC and answer the following questions:
2. Describe the different roles of the concrete and the abstract syntax in a language specification.
3. How are precedences of operators specified in a concrete syntax?
4. Why is it acceptable that an abstract syntax is ambiguous?
5. Explain the strategy for grammar design given on slide PLaC-3.4aa using the grammar on slide GSS-2.5
6. Consider the grammar on slide GSS-2.5. Find all productions that specify arbitrary long sequences of certain language constructs.
7. Describe in what sense the sequences are structurally different.
8. What other forms of sequences are possible, but do not occur in this grammar?
9. Find all productions in that grammar which specify that some language construct is optional.
10. Consider the grammar on slide GSS-2.5. Translate each production into an English sentence, which describes the meaning of the production quite formally. You may combine strongly related productions into one description.

Do not forget to take notes on what you did, what you learned, and which problems you encountered, and turn them in (see Assignment 1).