

Assignment 3

Parser:

In this assignment, you will use ML-Yacc to build a parser for the Fun language. Chapter 3 of your textbook (particularly 3.4) explains many of the concepts you'll need to know to complete this assignment. There is also documentation on ML-Yacc here:

<http://www.smlnj.org/doc/ML-Yacc/>

You will also need to refer to the Fun language reference:

www.math.us.edu.pl/~pgladki/teaching/2016-2017/tk_fun.html

Make a new directory **lab3** for this assignment, and copy all your **lab2** files into it. Then download the file

www.math.us.edu.pl/~pgladki/teaching/2016-2017/tk_lab3.zip

to get some new files: sources.cm fun.grm parse.sml compile.sml all.fun testcases.sml. This will overwrite your sources.cm with a new one.

For this assignment you will use the fun.lex you built in the previous assignment. If yours doesn't work well enough to be usable, ask me for a working one.

Edit **fun.grm** until it specifies a parser for the Fun language. Don't worry at first about filling in the semantic actions; just get the parsing and grammar disambiguation taken care of. Here are some comments and issues to take particular care of:

- The Fun language reference contains grammar rules that describe the syntactic structure of Fun.
- You will need to specify terminal names that correspond to the names of tokens generated by the lexer. ML-Yacc automatically generates a structure that will build the tokens. In other words, **fun.grm** will need to contain the following code (extended with the other terminal names):

```
%term EOF
| INT of int
| ID of string
| COMMA | SEMICOLON | COLON
| LPAREN | RPAREN | ...
```

The nonterminals you use can have any names that you want. For example:

```
%nonterm foo of Absyn.exp | ...
```

- The Fun language definition isn't 100% clear about the exact grammar you have to use—that's part of the "Fun" of this assignment. However, the file testcases.sml should be very helpful here: it's a list of pairs, where each pair of strings should parse into the same abstract syntax.
- If you use precedence directives, do so in a reasoned way. In other words, don't just throw in precedence directives to correct any random error in your grammar; be able to explain in your README how your use of precedence directives achieves the desired result, perhaps with reference to the fun.grm.desc file generated by ml-yacc.

- Compile using **CM.make "sources.cm"**; This will automatically run **ml-lex** and **ml-yacc** as necessary. Test your parser by invoking **Compile.compile "filename"**. This calls **Parse.parse** and then invokes the interpreter on the file. For example, use "all.fun" as the filename.
- Another way to test is to do `val x = Compile.go();` For each pair of strings in `testcases.sml`, it runs the parser on that pair of strings, and compares the abstract syntax (after all the Pos markers have been removed).

Once you have your grammar disambiguated and debugged, fill in the parser's semantic actions. These should generate abstract syntax in the Fun intermediate language. Here are a list of concerns to keep in mind.

- Some of the source level operations do not appear in the intermediate language. For instance, the "and", "or", "sequencing" (i.e., semicolon), and "unary minus" need to be compiled into other existing constructs. For example, "e1 and e2" can translate as if it were "if e1 then (if e2 then 1 else 0) else 0".
- Each object that you generate in the abstract syntax should be wrapped in a `Fun.Pos` constructor that marks the beginning and ending locations of the construct in the source code.

Add some support for detection and correction of common errors, using ML-Yacc constructs such as "%value" and "%change" described on pages 80-81 of MCIinML and in the ML-Yacc manual. Explain the support you add in your README file.

The file **all.fun** tests all syntactic constructs in a trivial way but you will certainly want to write other fun files to debug your compiler. If you want to submit additional test files, please do so.

Submit your README, fun.grm, and any *.fun test files you wish to. The deadline for this assignment is **Monday, December 19th, 2016**.