HW 9 – Parse Trees, Ambiguous Grammars and Recursive Descent Parsing

CS 421 – Fall 2014
Revision 1.0

Assigned Tuesday, October 28, 2014
Due Thursday, November 6, 2014, 23:59 PM
Extension 48 hours (20% penalty)

1 Change Log

1.0 Initial Release.

2 Turn-In Procedure

For the main problem, answer the problem below, save your work as a PDF (either scanned if handwritten or converted from a program), add the PDF to the subversion repository (`svn add hw9-submission.pdf`) and commit it (`svn commit -m "submitting hw9"`). Your file should be named `hw9-submission.pdf` and committed in your `assignments/hw9` directory. For the extra credit problem, your answer needs to be put in a seperate file named `hw9.ml` and added and committed to your `assignments/hw9` directory.

3 Objectives and Background

The purpose of this HW is to test your understanding of

- BNF grammars
- Grammar disambiguation
- Parse trees
- Writing a recursive descent parser for an LL(1) grammar

Another purpose of HW9 is to provide you with experience answering non-programming written questions of the kind you may experience on the second midterm and final.
Caution: It is strongly advised that you know how to do these problems before the second midterm.

4 Problems

1. (23 points) Consider the following grammar over the terminal alphabet {`int`, `list`, $*$, `'a`, `'b`, (,)} and non-terminal alphabet {<ty>, <var>}:

$$\begin{array}{rcl} < \texttt{ty} > & ::= & < \texttt{var} > \mid \texttt{int} \mid < \texttt{ty} > * < \texttt{ty} > \mid < \texttt{ty} > \texttt{list} \mid (< \texttt{ty} >) \\ < \texttt{var} > & ::= & \texttt{'a} \mid \texttt{'b} \end{array}$$

(This is a grammar for a simple language of types similar to those in OCaml.)

a. (9 points) Show that the above grammar is ambiguous by showing at least two distinct parse trees for the string "`'a list * int list`"

b. (9 points) Write a new grammar accepting the same language that is unambiguous, and such that the product type constructor $<$ `ty` $>$ `*` $<$ `ty` $>$ binds more tightly than the list type constructor $<$ `ty` $>$ `list`, and such that `*` associates to the right.

c. (5 points) Give the parse tree for "`'a list * int list`" using the grammar you gave in the previous part of this problem.

2. (**Extra Credit**) (10 points) Write a recursive descent parser for the grammar you gave in part b. We have given you the files `hw9common.ml`, `hw9common.cmi` and `hw9common.cmo`, which give you the datatype:

```
type token = INT | LIST  | STAR | VAR of string | LPAR | RPAR
```

We have also given you the files `type_lex.mll`, `type_lex.ml`, `type_lex.cmo`, and `type_lex.cmi`. These files give you the function `Type_lex.get_all_tokens : string -> Hw9common.token list`.

You should define a type `ty` (and some others) that represents parse trees based on grammar you gave in Problem 1, part b, and a function `parse: string -> ty option` that returns `Some` of a parse tree for the string if the whole string parses, or `None` if the string contains no lex errors but fails to fully parse.

Your answer needs to be put in a seperate file named `hw9.ml` and added and committed to your `assignments/hw9` directory.