Homework 2 — APL (due Wednesday, June 12, 2013)

Mayer Goldberg

June 2, 2013

Contents

1 Using parsing combinators

For this assignment, you need to select some language – it can be a small programming language, or a DSL, etc, and implement a parser using the implementation I posted for parsing combinators in Scheme. Once you have a parser for your language, you should be able to use it to do something interesting: Write a pattern matcher, an interpreter, a compiler, et cetera.

This assignment is *very* open; Please make the most of it, and use this as an opportunity to do something fun and/or useful. Here are some ideas, but please don't feel constrained by them; The only constraint is that the application should be non-trivial:

1. Support a language for pattern-matching using regular expressions. Write a parser for a grammar of regular expressions. The language you support should be similar to what is available in Unix/Linux in such applications as *sed*, *awk*, or the Java regular expression library. At the very least, it should be flexible enough to allow such expressions as `[A-Z]*[0-9]?`, et cetera. Once you have a parser, you can write the procedure `match?`, that will be used as follows: `(match? "[a-z]+" "moshe") #t`, `(match? "[a-z]+" "moshe_yossi") #f`, `(match? "[a-z]+.*" "moshe_bla1234 ") #t`

2. Implement a toy interpreter for a meaningful and non-trivial subset of BASIC or Pascal. It would actually be simpler to write a compiler into C.

3. A tiny language extending HTML, and that provides better support for tables. You can compile it to straight HTML.

et cetera.

If you have any doubts as to whether your application for this problem is sufficiently non-trivial, please contact me ahead of time!