COMP2212 Programming Language Concepts Coursework

Semester 2 2014/15

Introduction

Languages are sets of words over some alphabet. In Theory of Computing you studied a hierarchy of different kinds of languages: *regular* languages which are accepted by finite automata, *context free* languages, which are accepted by pushdown automata and *recursively enumerable* languages, which are accepted by Turing machines.

Your task is to *design and implement* a domain specific programming language that performs certain simple computations on languages. You are required to (1) invent an appropriate syntax and (2) write an interpreter (possibly using `ocamllex` and `ocamlyacc` for lexing and parsing). Your design should be guided by the five example problems listed below.

The specification is deliberately loose. If we haven't specified something precisely, it is a design decision for you to make: e.g. how to handle syntax errors, illegal inputs, etc. A significant part (30%) of the mark will be awarded for these qualitative aspects. The remaining 70% of the mark will be awarded for correctly solving a number of problems using your programming language. The correctness of your solution will be checked with a number of automated tests.

Problems

The inputs will take the following form:

L1
L2
.
.
.
Ln
k

where every line is terminated with a new line character. Here k is a positive integer and each of the Ls is a finite language. For the purposes of this coursework, all languages are assumed to consists of words constructed from the English lowercase alphabet. The integer k refers to how many words are to be printed from each of the expected output languages.

Languages are specified with curly braces. Apart from ordinary words over the lowercase letters, : is a special symbol denoting the empty word. The words are separated with commas, with arbitrary numbers of spaces and tabs in between. There is no assumption that the words are given in any particular order in any input language. Words can also be repeated in the input, and you are to ignore repetition (since languages are sets!).

There may be more than one output language. **You are required to order words lexicographically (dictionary ordering) in each output language**. See the examples in the list of problems below.

Problem 1 - Postfix

Take a language L and produce the language $La = \{\, xa \mid x \in L \,\}$, that is, the language that is like L, but an additional a at the end of each word.

Example input:	Expected output:
{a, b, c, ad} 15	{aa, ada, ba, ca}

Example input:	Expected output:
{a, a, b, c} 4	{aa, ba, ca}

Example input:	Expected output:
{:, a} 2	{a, aa}

Example input:	Expected output:
{:, a} 1	{a}

Example input:	Expected output:
{} 1	{}

Problem 2 - Union

Take two languages L_1 and L_2 and output their union.

Example input:	Expected output:
{a, b} {c, d} 4	{a, b, c, d}

Problem 3 - Union 2

Take three languages L_1, L_2 and L_3 and output $L_1 \cup L_2$ and $L_1 \cup L_3$.

Example input:	Expected output:
{a} {b} {c} 2	 {a, b} {a, c}

Problem 4 - Contatenation 1

Take a language and concatenate it with $a^* = \{:, a, aa, aaa, \ldots\}$.

Example input:	Expected output:
{b} 6	{b, ba, baa, baaa, baaaa, baaaa}

Problem 5 - Concatenation 2

Take a language and contatenate it with the language that consists of all words of length 2 over the letters a, b and c:

Example input:	Expected output:
{a} 5	{aaa, aab, aac, aba, abb}

Submission instructions

First submission

The first deadline is on Thursday March 5. All submissions will be done through handin. Precise submission instructions will be released a few days before the deadline.

You will be required to submit a zip file containing:

- the sources for an interpreter for your language, written in OCaml
- five programs, written in **YOUR** language, that solve the five problems specified above. The programs should be in files named `pr1.spl`, `pr2.spl`, `pr3.spl`, `pr4.spl`, `pr5.spl`.

We will compile your interpreter using the command `make` on linuxproj so you will need to include a Makefile in your zip file that produces an executable named `mysplinterpreter`. It is thus extremely important that prior to submission, you make sure that your interpreter **compiles on linuxproj** when the `make` command is executed in shell: if your code does not compile then you will be awarded 0 marks.

Your interpreter is to take a file name (that contains a program in your language) as a single command line argument and then expect input on standard input. After the input has been read in, the interpreter should produce any output on standard output and any error messages on standard error.

For each problem, we will test whether your code performs as expected by using a number of tests. We only care about correctness and performance will not be assessed (within reason). You can assume that for the tests we will use correctly formatted input.

For example, when assessing your solution for problem 1 we will execute the following command in shell

```
mysplinterpreter pr1.spl < input
```

where `input` will be a textfile containing the input for one of several tests. You will receive feedback based on the performance of your code in our test harness prior to the second deadline. The marks awarded for the first submission will count 20% of the total coursework mark (4% for each of the five problems).

Second submission

Shortly after the first deadline we will release a further five problems. The second deadline will be Thursday March 26. You will be required to submit two separate files.

First, you will need to submit a zip file containing programs (`pr6.spl`, `pr7.spl`, `pr8.spl`, `pr9.spl`, `pr10.spl`) written in your language that solve the additional problems. We will run our tests on your solutions and award marks for solving the additional problems correctly. This will form 50% of the total coursework mark (10% each for the five additional problems). You will have the option of resubmitting the interpreter, for a 50% penalty on this component. Thus, if you decide to resubmit your interpreter in the second submission the maximum possible total coursework mark is capped at 75%.

Second, you will be required to submit a 3 page user manual for your language in pdf format that explains the syntax, and describes any additional features (programmer convenience, type checking, informative error messages, etc.) This report, together with the five programs will be evaluated qualitatively and your marks will be awarded for the elegance and flexibility of your solution and the clarity of the user manual. These qualitative aspects will be worth 30% of the total coursework mark.

As you know, the coursework is to be done in pairs. As part of the second submission we will require a declaration of how marks are to be distributed amongst the members of your group. You will receive all feedback and your marks by Thursday April 30.