Graduate Programming Languages
Homework Assignment

Exercise 1: **Bookkeeping.** How long did this take you? Was it easy? Tell me something about yourself that I do not already know. All non-empty answers receive full credit. The usual submission mechanism applies.

Exercise 2: Read *Abstraction Mechanisms in CLU* by Liskov et al. and then read *What is "Object-Oriented Programming"?* by Stroustrup. Write a formal reaction to the two articles. Your writeup should not exceed four paragraphs. Write as if you were writing the motivation section or related work section for a paper submission (perhaps a paper proposing a new notion of "abstraction" that must highlight the pros and cons of previous efforts). Email your writeup to me as a **flat ASCII text file**.

Exercise 3: *Regular Expressions* are commonly used as abstractions for string matching. Here is an abstract grammar for regular expressions:

r	::=	"x"	singleton — matches the character `x`
	|	empty	skip — matches the empty string
	|	$r_1\ r_2$	concatenation — matches r_1 followed by r_2
	|	$r_1 \mid r_2$	or — matches r_1 or r_2
	|	$r*$	Kleene star — matches 0 or more occurences of r
	|	.	matches any single character
	|	["x" – "y"]	matches any character between `x` and `y` inclusive
	|	$r+$	matches 1 or more occurences of r
	|	$r?$	matches 0 or 1 occurence of r

We will call the first five cases the *primary* forms of regular expressions. The last four cases can be defined in terms of the first five. We also give an abstract grammar for strings (modeled as lists of characters):

s	::=	nil	empty string
	|	"x" :: s	string with first character `x` and other characters s

We write $\texttt{"bye"}$ as shorthand for $\texttt{"b"} :: \texttt{"y"} :: \texttt{"e"} :: \mathsf{nil}$. This exercise requires you to give large-step operational semantics rules of inference related to regular expressions matching strings. We introduce a judgment:

$$\vdash r \;\mathsf{matches}\; s \;\mathsf{leaving}\; s'$$

The interpretation of the judgment is that the regular expression r matches some prefix of the string s, leaving the suffix s' unmatched. If $s' = \mathsf{nil}$ then r matched s exactly. Examples:

$$\vdash \texttt{"h"}(\texttt{"e"}+) \;\mathsf{matches}\; \texttt{"hello"} \;\mathsf{leaving}\; \texttt{"llo"}$$

Note that this operational semantics may be considered *non-deterministic* because we expect to be able to derive all three of the following:

$$\begin{array}{l}
\vdash (\texttt{"h"} \mid \texttt{"e"})* \;\mathsf{matches}\; \texttt{"hello"} \;\mathsf{leaving}\quad \texttt{"ello"} \\
\vdash (\texttt{"h"} \mid \texttt{"e"})* \;\mathsf{matches}\; \texttt{"hello"} \;\mathsf{leaving}\quad \texttt{"hello"} \\
\vdash (\texttt{"h"} \mid \texttt{"e"})* \;\mathsf{matches}\; \texttt{"hello"} \;\mathsf{leaving}\quad \texttt{"llo"}
\end{array}$$

Here are two rules of inference:

$$\frac{s = \texttt{"x"} :: s'}{\vdash \texttt{"x"} \;\mathsf{matches}\; s \;\mathsf{leaving}\; s'} \qquad \frac{}{\vdash \mathsf{empty} \;\mathsf{matches}\; s \;\mathsf{leaving}\; s}$$

Give large-step operational semantics rules of inference for the other three primal regular expressions.

Exercise 4: We will use *denotational semantics* to model the fact that a regular expression can match a string leaving many possible suffices. Let S be the set of all strings, let $\mathcal{P}(S)$ be the powerset of S, and let RE range over regular expressions. We introduce a semantic function:

$$\mathcal{R} : \mathsf{RE} \to (S \to \mathcal{P}(S))$$

The interpretation is that $\mathcal{R}(r)$ is a function that takes in a string-to-be-matched and returns a set of suffices. We might intuitively define $\mathcal{R}$ as follows:

$$\mathcal{R}[\![\, r \,]\!](s) = \{s' \mid \;\vdash r \;\mathsf{matches}\; s \;\mathsf{leaving}\; s'\}$$

In general, however, one should not define the denotational semantics in terms of the operational semantics. Here are two correct semantic functions:

$$\begin{array}{ll}
\mathcal{R}[\![\, \texttt{"x"} \,]\!](s) & = \{s' \mid s = \texttt{"x"} :: s'\} \\
\mathcal{R}[\![\, \mathsf{empty} \,]\!](s) & = \{s\}
\end{array}$$

Give the denotational semantics functions for the other three primal regular expressions. Your semantics functions *may not* reference the operational semantics.

Exercise 5: We want to update our operational semantics for regular expressions to capture multiple suffices. We want our new operational semantics to be deterministic — it should give the same the same answer as the denotational semantics above. We introduce a new judgment:

$$\vdash r \text{ matches } s \text{ leaving } S$$

And use rules of inference like the following:

$$\frac{}{\vdash \text{"x" matches } s \text{ leaving } \{s' \mid s = \text{"x"} :: s'\}} \qquad \frac{}{\vdash \text{empty matches } s \text{ leaving } \{s\}}$$

$$\frac{\vdash r_1 \text{ matches } s \text{ leaving } S \quad \vdash r_2 \text{ matches } s \text{ leaving } S'}{\vdash r_1 \mid r_2 \text{ matches } s \text{ leaving } S \cup S'}$$

You must do one of the following:

- *either* give operational semantics rules of inference for $r*$ and $r_1 r_2$. Your opsem rules may *not* reference the denotational semantics. You may *not* place a derivation inside a set constructor, as in: $\{x \mid \exists y.\ \vdash r \text{ matches } x \text{ leaving } y\}$. Each inference rules must have a finite and fixed set of hypotheses.
- *or* argue in one or two sentences that it cannot be done correctly in the given framework. Back up your argument by presenting two attempted but “wrong” rules of inference and show that each one is either unsound or incomplete with respect to our intuitive notion of regular expression matching.

Part of doing research is getting stuck. When you get stuck, you must be able to recognize whether “you are just missing something” or “the problem is actually impossible”.

Exercise 6: Optional. In the class notes (but not, alas, in class) we defined an equivalence relation $c_1 \sim c_2$ for IMP commands. Computing equivalence turned out to be undecideable: $c \sim c$ iff c halts. We can define a similar equivalence relation for regular expressions: $r_1 \sim r_2$ iff $\forall s \in S.\ \mathcal{R}[\![r_1]\!](s) = \mathcal{R}[\![r_2]\!](s)$. You must *either* claim that $r_1 \sim r_2$ is undecideable by reducing it to the halting problem *or* explain in two or three sentences how to compute it *or* write “I choose not to do this problem” (you will receive full credit). You may assume that I am familiar with the relevant literature.

Exercise 7: Download the Homework 3 code pack from the course web page. Write an interpreter for regular expressions based on the denotational semantics. In particular, you must write a function $\text{matches} : \text{RE} \to S \to \mathcal{P}(S)$. Your interpreter must handle all of the regular expression forms, not just the primal ones.

Exercise 8: Submit an `example-re` file according to the instructions in the code pack.