Quiz Section 8: Languages

Task 1 – Regular Expressions

For each of the following, construct regular expressions that match the given set of strings:

a) Base 10 numbers (e.g., there should be no leading zeroes).

b) All base-3 numbers that are divisible by 3.

c) All binary strings that contain the substring "111", but not the substring "000".

Task 2 – CFGs

For each of the following, construct a context-free grammar that generates the given set of strings.

If your grammar has more than one variable (also called a non-terminal symbol), write a sentence describing what sets of strings you expect each variable in your grammar to generate. For example, if your grammar was:

$$\begin{aligned} \mathbf{S} &\to \mathbf{E} \mid \mathbf{O} \\ \mathbf{O} &\to \mathbf{EC} \\ \mathbf{E} &\to \mathbf{EE} \mid \mathbf{CC} \\ \mathbf{C} &\to 0 \mid 1 \end{aligned}$$

You could say "C generates binary strings of length one, E generates (non-empty) even length binary strings, and O generates odd length binary strings." It is also fine to use a regular expression, rather than English, to describe the strings generated by a variable (assuming such a regular expression exists). If you only use one variable (i.e., S), an explanation is not required but you should provide the grammar.

a) All binary strings that start with 11.

b) All binary strings that contain at most one 1.

c) All binary strings that contain at least three 1's.

d) All strings over $\{0,1,2\}$ with the same number of 1s and 0s and exactly one 2.

Hint: Try modifying the grammar from lecture for binary strings with the same number of 1s and 0s. (You may need to introduce new variables in the process.)

Task 3 – Recursively Defined Sets of Strings

In this problem, we will demonstrate by example that context-free grammars (with exactly one variable) and recursively defined sets are capable of describing exactly the same languages.

As an example, consider binary strings of even length. We can define a grammar for them as follows:

$$\mathbf{S} \to \mathbf{S}00 \mid \mathbf{S}01 \mid \mathbf{S}10 \mid \mathbf{S}11 \mid \varepsilon$$

To do the same thing with recursion, we first define a set $\mathcal{T}$ that records a sequence of substitutions used by the grammar. This can be done as follows:

Basis: Empty $\in \mathcal{T}$
Recursive Step: If $L \in \mathcal{T}$ and $a, b \in \{0, 1\}$, then Append$(L, a, b) \in \mathcal{T}$

For example, the derivation $S \to S01 \to S1101 \to \varepsilon 1101$ would be represented by the tree

$$\text{Append}(\text{Append}(\text{Empty}, 1, 1), 0, 1)$$

Note that Append is not a function call! Append(L, a, b) is data, not code. We saw the same thing in Homework 6 Task 6, where Tree(L, R) denotes a node in the tree. Here, Append makes tree nodes.

(Recursively defined sets that keep track of the operation of a grammar are generically referred to as "parse trees". Our set $\mathcal{T}$ consists of parse trees for the grammar **S** above.)

To get the string in the language, we define the following recursive function that traverses the tree and outputs the string that was generated:

$$\begin{aligned} \text{traverse}(\text{Empty}) &:= \varepsilon \\ \text{traverse}(\text{Append}(L, a, b)) &:= \text{traverse}(L)ab \end{aligned}$$

Note that traverse returns a string, so on the second line, traverse$(L)ab$ means that we make a recursive call to traverse(L) and then append characters 'a' and 'b' to the string that it returned.

With this definition, we can see, for example, that

$$\begin{aligned} &\text{traverse}(\text{Append}(\text{Append}(\text{Empty}, 1, 1), 0, 1)) \\ &= \text{traverse}(\text{Append}(\text{Empty}, 1, 1))01 && \text{def of traverse} \\ &= \text{traverse}(\text{Empty})1101 && \text{def of traverse} \\ &= \varepsilon 1101 && \text{def of traverse} \end{aligned}$$

showing that the this was indeed a derivation of the string "1101".

Having seen a first example, let's try out some more....

Consider the following grammar:

$$\mathbf{S} \to 0\mathbf{S}1 \mid \mathbf{S}1 \mid \varepsilon$$

It generates all strings of the form 0^m1^n with $m \leqslant n$.

a) Write a recursive definition of a set $\mathcal{T}$ of parse trees for this grammar.

b) Define a recursive function that traverses your parse trees from part (a) and outputs the string that was generated by the grammar above.

Be careful to indicate anywhere that string concatenation is used with $\bullet$ rather than juxtaposition. (The latter is just for appending one character to a string.)

To see the other direction, consider the following recursive definition:

Basis: $\text{End} \in \mathcal{T}$
Recursive Step: If $L \in \mathcal{T}$ and $a \in \{0, 1\}$, then $\text{StartsWith}(a, L) \in \mathcal{T}$

And suppose that we generate strings from these with the following traversal:

$$\text{traverse}(\text{End}) := 00$$
$$\text{traverse}(\text{StartsWith}(a, L)) := a \bullet \text{traverse}(L)$$

The set $S = \{\text{traverse}(L) \ : \ L \in \mathcal{T}\}$, which consists of all strings that can be generated by building a parse tree and then traversing it, contains all binary strings ending with "00".

c) Define a context-free grammar that matches these definitions.

Note that your grammar must not only generate all strings ending in "00". It must have derivations that exactly correspond to the parse trees above!

Task 4 – Good, Good, Good, Good Relations

For each of the relations below, determine whether or not it has each of the properties of reflexivity, symmetry, antisymmetry, and/or transitivity. If a relation has a property, simply say so without any further explanation. If a relation does not have a property, state a counterexample, but do not explain your counterexample further.

a) Let $R = \{(x, y) \ : \ x = y + 1\}$ on $\mathbb{N}$.

b) Let $R = \{(x, y) \ : \ x^2 = y^2\}$ on $\mathbb{R}$.

c) Let $R = \{(x, y) \ : \ \text{len}(xy) \text{ is even}\}$ on $\{0, 1\}^*$.

Task 5 – Relation Proof I

Let R be a relation on set A. Recall from lecture that "R is *reflexive*" iff

$$\forall a \in A\,((a, a) \in R)$$

i.e. every pair (a, a) is in R.

Now, let R and S be relations on a set A, and consider the following claim:

> Given that both R and S are reflexive, it follows that $R \cap S$ is reflexive.

Write an **English proof** that the claim holds.

Note: even though we want you to write your proof directly in English, it must still look like the translation of a formal proof. In particular, you must include all steps that would be required of a formal proof, excepting only those that we have explicitly said are okay to skip in English proofs (e.g., Elim $\exists$).

Task 6 – Relation Proof II

Let R be a relation on a set A. Recall from lecture the definition of the *composition* of two relations:

$$R \circ R := \{(a, b) \mid \exists \mathbf{c}\, ((a, \mathbf{c}) \in R \wedge (\mathbf{c}, b) \in R)\}$$

Now, consider the following claim:

Given that R is reflexive, it follows that $R \subseteq R \circ R$.

Write an **English proof** that the claim holds.

Follow the structure of our template for subset proofs.

Note: even though we want you to write your proof directly in English, it must still look like the translation of a formal proof. In particular, you must include all steps that would be required of a formal proof, excepting only those that we have explicitly said are okay to skip in English proofs (e.g., Elim $\exists$).