Section 07: Solutions

1. A Hint for the Homework

This problem uses a similar technique to the number theory problem on HW6.

Prove that if $18|(n-4)$ then $19|(2^n+3)$ for all integers $n \geq 0$.

Hint: You may use without proof the fact that $2^{18} \equiv_{19} 1$.

Solution:

Let n be an arbitrary integer ≥ 0 such that $18|n-4$. Then by definition of divides, $n = 18k+4$ for some integer k.

$$2^n \equiv_{19} 2^{18k+4} \equiv_{19} (2^{18})^k \cdot 2^4 \equiv_{19} 1^k \cdot 16 \equiv_{19} 16.$$

And so,

$$2^n + 3 \equiv_{19} 16 + 3 \equiv_{19} 0.$$

Therefore by definition of $\equiv_{19}$, $19|2^n+3$. Since n was arbitrary, the claim holds for all integers $n \geq 0$.

2. Regular Expressions

(a) Write a regular expression that matches base 10 numbers (e.g., there should be no leading zeroes).

Solution:

$$0 \cup ((1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)(0 \cup 1 \cup 2 \cup 3 \cup 4 \cup 5 \cup 6 \cup 7 \cup 8 \cup 9)^*)$$

(b) Write a regular expression that matches all base-3 numbers that are divisible by 3.

Solution:

$$0 \cup ((1 \cup 2)(0 \cup 1 \cup 2)^*0)$$

(c) Write a regular expression that matches all binary strings that contain the substring "111", but not the substring "000".

Solution:

$$(01 \cup 001 \cup 1^*)^*(0 \cup 00 \cup \varepsilon)111(01 \cup 001 \cup 1^*)^*(0 \cup 00 \cup \varepsilon)$$

(d) Write a regular expression that matches all binary strings that do not have any consecutive 0's or 1's.

Solution:

$$((01)^*(0 \cup \varepsilon)) \cup ((10)^*(1 \cup \varepsilon))$$

(e) Write a regular expression that matches all binary strings of the form $1^k y$, where $k \geq 1$ and $y \in \{0,1\}^*$ has at least k 1's.

Solution:

$$1(0 \cup 1)^* 1(0 \cup 1)^*$$

Explanation: While it may seem like we need to keep track of how many 1's there are, it turns out that we don't. Convince yourself that strings in the language are exactly those of the form $1x$, where x is any binary string with at least one 1. Hence, x is matched by the regular expression $(0 \cup 1)^* 1 (0 \cup 1)^*$.

3. CFGs

Write a context-free grammar to match each of these languages.

(a) All binary strings that end in 00.
Solution:

$$\mathbf{S} \to 0\mathbf{S} \mid 1\mathbf{S} \mid 00$$

(b) All binary strings that contain at least three 1's.
Solution:

$$\mathbf{S} \to \mathbf{TTT}$$
$$\mathbf{T} \to 0\mathbf{T} \mid \mathbf{T}0 \mid 1\mathbf{T} \mid 1$$

(c) All binary strings of the form xy, where $|x| = |y|$, but $x \neq y$.

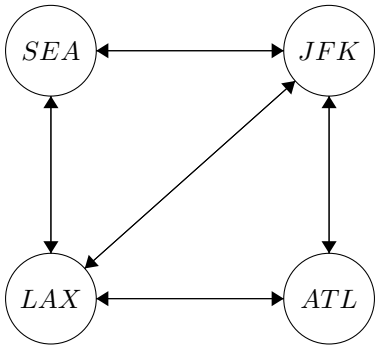
Solution:

$$\mathbf{S} \to \mathbf{AB} \mid \mathbf{BA}$$
$$\mathbf{A} \to 0 \mid 0\mathbf{A}0 \mid 0\mathbf{A}1 \mid 1\mathbf{A}0 \mid 1\mathbf{A}1$$
$$\mathbf{B} \to 1 \mid 0\mathbf{B}0 \mid 0\mathbf{B}1 \mid 1\mathbf{B}0 \mid 1\mathbf{B}1$$

Explanation: We will explain the forward direction (i.e. this grammar generates strings of the desired form); in particular, we will examine strings generated by the rule **AB**, as the other rule follows similarly. An arbitrary string generated by **AB** will look like $a_1 0 a_2 b_1 1 b_2$, where $a_1, a_2, b_1, b_2 \in \{0,1\}^*$, $|a_1| = |a_2| = k_1$, and $|b_1| = |b_2| = k_2$ for some $k_1, k_2 \in \mathbb{N}$. In particular, we can "repartition" the substring $a_2 b_1$ into $a_2' b_1'$ s.t. $|a_2'| = k_2$ and $|b_1'| = k_1$. Letting $x = a_1 0 a_2'$ and $y = b_1' 1 b_2$, observe that $|x| = |y| = k_1 + k_2 + 1$ and x and y differ at the $(k_1 + 1)$-th character.

4. Airports

Suppose you want to book a flight from Seattle (SEA) to New York (JFK), but you're not sure which route to take. In the diagram below, suppose that lines are drawn between two airports if and only if there exist regular flights between them.



Design an CFG which describes all the possible flight paths from SEA to JFK. A flight path is defined as a sequence of airport codes separated by $\rightsquigarrow$ to indicate a flight. For example, the following are valid:

- $SEA \rightsquigarrow LAX \rightsquigarrow ATL \rightsquigarrow JFK$
- $SEA \rightsquigarrow LAX \rightsquigarrow JFK$
- $SEA \rightsquigarrow JFK \rightsquigarrow ATL \rightsquigarrow LAX \rightsquigarrow JFK$

Note that a flight path may visit an airport any number of times, however it cannot traverse between airports which do not have a line between them on the map and it must begin with SEA and end with JFK. Your non-terminals should be the set $\{SEA, LAX, ATL, JFK, \rightsquigarrow\}$.

Solution:

Use sea as the starting non-terminal.

$$
\begin{aligned}
sea &\rightarrow SEA \rightsquigarrow jfk \mid SEA \rightsquigarrow lax \\
lax &\rightarrow LAX \rightsquigarrow sea \mid LAX \rightsquigarrow atl \mid LAX \rightsquigarrow jfk \\
atl &\rightarrow ATL \rightsquigarrow lax \mid ATL \rightsquigarrow jfk \\
jfk &\rightarrow JFK \mid JFK \rightsquigarrow sea \mid JFK \rightsquigarrow lax \mid JFK \rightsquigarrow atl
\end{aligned}
$$