CSE355 Homework Three Sample Solutions

Question 1

Let $\Sigma = \{0, 1\}$. A string $w \in \Sigma^\star$ of length $n = |w|$ is called *sparse* if every two 1s in the string have at least $\lceil \log_2 n \rceil$ 0s between them. Let $L_S \subseteq \Sigma^\star$ be the language of sparse strings. Show that L_S is not regular.

Suppose (to the contrary) L is regular: then by the Pumping Lemma for Regular Languages, there is a pumping length ℓ for L_S. Choose any $p \geq \ell + 4$.[1] Choose the string $w = 10^{2^p - p - 3}10^p1$ (I'll explain how I obtained this string below). We have that $|w| \geq p$, and $w \in L$ because $|w| = 2^p$, and $\log_2(2^p) = p$, and all pairs of 1's are separated by at least p 0's. We want to show that for any decomposition of w into uvx, there is a way to pump so as to leave the language. Note that we have the following cases:

- $u = \epsilon, v = 10^\alpha, x = 0^{2^p - p - 3 - \alpha}10^p1$ (with $\alpha \geq 0$). We are justified here to say v does not include the second 1 because we chose p to be large enough. Choose $i = 2$: $uv^2x = 10^\alpha 10^\alpha 0^{2^p - p - 3 - \alpha}10^p1 = 10^\alpha 10^{2^p - p - 3}10^p1$. Since $|v| \geq 1$, we have that $|uv^2x| \geq 2^p + 1$. Since $\alpha \leq p$ (it's actually $\leq p - 1$), and $p < \lceil \log_2(2^p + 1) \rceil$, we have that $uv^2x \notin L$. We actually get the same conclusion if we considered the third "run" of 0's at the end of uv^2x.
- $u = 10^\alpha, v = 0^\beta, x = 0^{2^p - p - 3 - \alpha - \beta}10^p1$. We can just analyze this case just on string length. If we choose any $i \geq 2$, then we will add at least one more 0 in the first "set" of 0's. Then the length of the string will be $\geq 2^p + 1$. And so each "set" of 0's must have length $\geq \lceil \log_2(2^p + 1) \rceil > p$. We get a contradiction because we have a set of 0's of length p (the second set).

Since we have exhausted all the cases, and found a contradiction in each, L_S is not regular.

So how did I find this string? A common technique for using the Pumping Lemma is to get "as close" to, or "match," a requirement (i.e., be in the language) as possible, then choosing a value of i to make the contradiction happen. In this case, I wanted a certain substring of 0's that has *exactly* $\lceil \log_2(n) \rceil$ 0's in length, and then choose a value of i for which the value of $\lceil \log_2(n) \rceil$ is bigger, but have this considered substring be unmodified, giving us the contradiction. Note, we don't want to pump this set of 0's, because we would make the analysis much more difficult (or might not even work).

A good strategy from this discussion (which will eventually give us w above) then is to look at the string 10^c10^p1 (where p is the pumping length from before); we want to find the value of c such that $p = \log_2(n)$ exactly, but at the same time guarantee that c is large enough so that the 0^p substring at the end is unmodified. We can accomplish both of these simultaneously: since the length of this string is $3 + p + c$, we want to have that $p = \log_2(3 + p + c)$. Exponentiating (with base 2) both sides gives $2^p = 3 + p + c$. Solving for c gives $c = 2^p - p - 3$, as was done in the proof.

Questions for thought:

1. What if instead of having *every* two 1s in the string, we only had the $\lceil \log_2 n \rceil$ requirement on at least k pairs of 1s (for a fixed k)? Are there any cases where this is regular?
2. Instead of having *at least* $\lceil \log_2 n \rceil$ 0's between every pair of 1s, we had *at most* this number?
3. Is L_S context-free?

[1] The reason for this is subtle. It may happen that ℓ is small, which makes the number of in which case $2^\ell - \ell - 3$ is small (say, less than ℓ). Performing algebraic manipulation and approximating the result shows that in order to avoid the scenario of $2^\ell - \ell - 3 < \ell$, we have that $\ell \geq 4$ (it's actually is ≈ 3.247, but ℓ has to be an integer). So we just add 4 to ℓ to avoid this degenerate case if ℓ happened to be small.

Question 2

Let $\Sigma = \{0, 1\}$. A string $w \in \Sigma^\star$ is called *prefix-dense* if every prefix of w contains at least twice as many 1s as 0s. Let $L_P \subseteq \Sigma^\star$ be the language of prefix-dense strings.

Question 2a

Show that L_P is not regular.

Suppose (to the contrary) that L_P were regular. Then there is a pumping length p for L_P. Choose $w = 1^{2p}0^p$ - it has length at least p and is in L_P because every prefix has at least twice as many 1's as 0's. Since $|uv| \leq p$, we can write $u = 1^\alpha, v = 1^\beta, x = 1^{2p-\alpha-\beta}0^p$. Choose $i = 0$: $ux = 1^{2p-\beta}0^p$. Since $\beta \geq 1$, there is a prefix (i.e., the entire string) that has fewer than twice as many 1s as 0s. Therefore, since $ux \notin L_P$, L_P is not regular.

Question 2b

Show that L_P is context-free.

We construct a CFG for L_P:

$$S \to \epsilon \mid S1S \mid S1S1S0S$$

This does not produce any strings that are not in L_P: The first rule is valid because ϵ has no 1's or 0's, so their counts are in the correct ratio. Inserting a 1 into any valid string cannot make it invalid, as every prefix has either the same counts as before, or else an additional 1 and no additional 0's, so the second rule is also valid. The last rule matches every 0 with two earlier 1's, ensuring that the number of 0's in a prefix never exceeds half the number of 1's, as required. Finally, the positions of the S's on the right-hand sides are all permissible because inserting a prefix-dense string into another always yields a prefix-dense string.

This does produce all strings in L_P: Given a prefix-dense string, each 0 can be matched off with two earlier 1's (making use of the third case) without reusing any 1's, or else the prefix ending with that 0 would have less than twice as many 1's as 0's. Then any additional 1's are taken care of by the second case. Finally, all the superfluous S's are replaced by ϵ.

Question 3

Let X be the language $\{a^n b^m a^\ell b^\ell a^m b^n : n, m, \ell \geq 0\}$.

Question 3a

Show that X is not regular. Suppose (to the contrary) that X is regular. Then X has a pumping length p. Choose the string $w = a^p b^p a^p b^p a^p b^p$. It certainly has length at least p, and is in the language by setting $n = m = \ell = p$. Since $|uv| \leq p$, we have that $u = a^\alpha, v = a^\beta, x = a^{p-\alpha-\beta}b^p a^p b^p a^p$. Choosing $i = 2$ gives the string $uv^2x = a^{p+\beta}b^p a^p b^p a^p b^p$. The only way that this string is in X is if $\beta = 0$ (to match the first set of a's with the last set of b's), which is a contradiction. Therefore, X is not regular.

Question 3b

Show that X is context-free. This goes off of the simple idea of the CFG for $\{0^n 1^n : n \geq 0\}$: $S \to 0S1 \mid \epsilon$. We can use a similar rule to match the first set of a's with the last set of b's, then have a rule that proceeds to a new variable to work on the first set of b's and the last set of a's. And finally, we can match the inner a's and b's:

$$S \to aSb \mid A$$
$$A \to bAa \mid B$$
$$B \to aBb \mid \epsilon$$

We can use this grammar because there is no dependence between n, m, ℓ. Question for thought: if we instead considered the language $\{a^n b^m a^n b^m a^m b^n : n, m \geq 0\}$, can you make a CFG for this language? How about $\{a^n b^m a^n b^n a^m b^n : n, m \geq 0\}$?

Question 4

Let L be a context-free language generated by the context-free grammar G. Let L' be the language that contains exactly the strings that both belong to L and have length a multiple of 3. Show that L' must be context-free by devising and describing a method for producing a context-free grammar for L' starting with the grammar G.

We form G' from G as follows. The variables consist of V_0, V_1, and V_2 for each variable V of G. The start variable is S_0, where S is the start variable of G. Finally, for each rule of G, let V be the variable on the left-hand side. For each variable W appearing on the right-hand side, consider replacing it with each of W_0, W_1, and W_2 in turn, where the choice is made independently for each such variable. For each resulting right-hand side, we count the number of terminals and add all the variables' subscripts; call the total t. Then we add a rule to G' with that right-hand side and with $V_{t \pmod 3}$ on the left-hand side. For example, if the right-hand side is ϵ, the rule added will be $V_0 \to \epsilon$.

The resulting grammar keeps track of the number of terminals (mod 3) to which a variable is permitted to expand, so that any string in its language must have a length of 0 (mod 3). The rules allow distributing the terminals between the variables on the right-hand side in any manner that is consistent with the remainder specified on the left-hand side. For any allowed length, the strings that are produced are the same as L. As a result, it contains all and only the strings of L'.