Honors Theory, Spring 2002, Yap
Homework 2
Due: Wed Feb 13 in class

MOSTLY ABOUT CONTEXT FREE LANGUAGES.

1. No credit work, as this exercise is self-rewarding :-). Recall the example of translating "`The spirit is willing but the flesh is weak`" into Russian, and back again, producing "`The vodka is strong but the meat is rotten`". The sentence "`Out of sight, out of mind`" was translated into "`Blind idiot`". Try to outdo these machine translations produced in the early days of MT. [It only goes to prove that this problem is too hard for machines.]

2. (10+15 Points)
For $n \geq 1$, let $\Sigma_n = \{a_1, \ldots, a_n\}$ be an alphabet with n letters. Define $L_n \subseteq \Sigma_n^*$ to comprise those words w for which there is some $i = 1, \ldots, n$ such that a_i does not occur in w.
(i) Show that there is an nfa N_n with $n+1$ states that accept L_n.
(ii) Show that every dfa tha accepts L_n has at least 2^n states. HINT: why 2^n?

SOLUTION (i): We construct a NFA M with $n+1$ states that accepts L_n.

FIGURE 2 HERE

$M = (Q, \Sigma, \delta, q_0, F)$
$Q = \{q_0, q_1 ... q_n\}$
$\Sigma = \{a_1, a_2 ... a_n\}$
$F = \{q_1, q_2 ... q_n\}$
$\delta(q_0, \epsilon) = q_i$, $i \neq 0$
$\delta(q_i, a_j) = q_i$, $i \neq j$, $i \neq 0$.

SOLUTION (ii): Assign to each string w a binary characteristic number $f(w) = b_n ... b_0$, where $b_i = 0$ if a_i does not appear in w, and $b_i = 1$ if a_i appears in w. There are 2^n such numbers. Only the strings that have the characteristic number with all bits set to 1 are rejected.
Show that every DFA that accepts L_n has at least 2^n states.
Suppose not, then there is a DFA M with less than 2^n states that accepts L_n. According to Pigeonhole Principle, there are at least two strings w and w' with different characteristic numbers x and y that reach the same state q_1. We have:
$\delta(q_0, w) = q_1$
$\delta(q_0, w') = q_1$.
As $x \neq y$, they have at least one bit b_k that is different. WLOG, assume $b_k = 1$ in x, and $b_k = 0$ in y. Choose a string $s = a_1 ... a_{k-1} a_{k+1} ... a_n$, then $f(ws)$ has all bits equal to 1, and $f(w's)$ has all bits except b_k equal to 1. So ws should be rejected, while $w's$ should be accepted. But
$\delta(q_0, ws) = \delta(q_1, s) = \delta(q_0, w's)$.
We obtain a contradiction.

3. (15+10 Points)
Let $A, B \subseteq \Sigma^*$. The **right quotient** of A by B is defined to be

$$A/B := \{w \in \Sigma^* : (\exists u \in B)[wu \in A]\}.$$

(i) Show that if A is context free and B is regular, then A/B is context free.
(ii) Use part (i) to show that the language $\{0^p 1^n : p \text{ is prime}, n > p\}$ is not context free.

SOLUTION (i): As A is context free, there is a pda $M_1 = \{Q_1, \Sigma_1, \Gamma_1, \delta_1, S, F_1\}$ that recognizes it. As B is regular, there is a dfa $M_2 = \{Q_2, \Sigma_2, \delta_2, q_0, F_2\}$ that recognizes it. We may assume $\Sigma_1 = \Sigma_2$. We construct a pda M that recognizes A/B:
$M = \{Q, \Sigma, \Gamma, \delta, S, F\}$
$Q = Q_1 \cup (Q_1 \times Q_2)$

$\Sigma = \Sigma_1$
$\Gamma = \Gamma_1$
$F = F_1 \times F_2$.
There are three steps. (a) The basic idea is that M will initially simulate the actions of M_1 until the end of input is reached. To do this, we put all the transitions of M_1 to M, but make all the accepting states of M_1 nonaccepting in M.

Once the end of input is reached, M will try to simulate the actions of M_2 on some unknown (hidden) input sequence. This is achieved by letting M perform ϵ-transitions based on this hidden input sequence. But, as M_2 acts on this hidden sequence, M_1 is also acting on exactly the same sequence. If both M_1 and M_2 halts at the same time, then we accept. Hence M must keep track of both the states of M_1 and M_2.
(b) First, we ϵ-transitions from Q_1 to $Q_1 \times Q_2$: for each $q_i \in Q_1$ add a transition $q_i, \epsilon, \epsilon \rightarrow q_{i0}, \epsilon$.

(c) Now we add ϵ-transitions to simulation the simultaneous actions of M_1 and M_2. Add a new state q_{ij} for each pair (q_i, q_j), where $q_i \in Q_1, q_j \in Q_2$. Add transitions among the states in $Q_1 \times Q_2$: for each symbol a such that $q_i, a, A \rightarrow q_I, B$ in M_1, and $q_j, a \rightarrow q_J$ in M_2
add a transition $q_{ij}, \epsilon, A \rightarrow q_{IJ}, B$
The accepting states are those q_{ij}'s, with $q_i \in F_1$, and $q_j \in F_2$.

We leave it as an exercise for the student to prove that this construction is correct.

SOLUTION (ii):
Let $A = \{0^p 1^n, p \text{ is prime}, n \geq p\}$. By way of contradiction, assume A is context free. Let $B = \{01^*\}$ and $C = \{0^{p-1}, p \text{ is prime }\}$. As B is regular, then $C = A/B$ is context free by part (i). But we will show that C is not context free, which is a contradiction. Let l be the pumping length for C. Select the string $s = 0^{m-1}$ where m is a prime greater than l. By the pumping lemma, $s = uvxyz$ where $|vxy| \leq l$ and $|vy| > 0$, and $uv^ixy^iz \in C$ for all $i \geq 0$. Pick $i = m+1$. Then $|uv^ixy^iz| = |uvxyz| + |vy| \times (i-1) = m - 1 + m \times |vy| = m \times (|vy|+1) - 1$. As $m \geq 2$ and $|vy| \geq 1$, $|uv^ixy^iz| + 1$ can not be a prime. Contradiction.

4. (10 Points Each) Prove or disprove that the following are context free:
(i) $L_1 = \{w \in \{a,b,c\}^* : \#_a(w) = \#_b(w) \text{ or } \#_b(w) = \#_c(w) \text{ or } \#_a(w) = \#_c(w)\}$.
(ii) $L_2 = \{w \in \{a,b,c\}^* : \#_a(w) \neq \#_b(w) \text{ or } \#_b(w) \neq \#_c(w) \text{ or } \#_a(w) \neq \#_c(w)\}$.
(iii) $L_3 = \{w \in \{a,b,c\}^* : \#_a(w) = \#_b(w) \text{ and } \#_b(w) = \#_c(w) \text{ and } \#_a(w) = \#_c(w)\}$.
NOTE: $\#_a(w)$ counts the number of occurences of a in w.

SOLUTION (i): L_1 is context free, as there is a PDA that recognizes it.

FIGURE 1 HERE

SOLUTION (ii): L_2 is context free, as there is a PDA that recognizes it.

FIGURE 2 HERE

SOLUTION (iii): Use the pumping lemma to show that L_3 is not context free. If L_3 is context free, let p be its pumping length. Let $s = a^p b^p c^p$. Clearly $s \in L_3$. Let $s = uvxyz$ be the decomposition of s given by the pumping lemma. As $|vxy| \leq p$, vy can not contain all three types of alphabet symbols. The string uv^2xy^2z can not contain equal numbers of a's, b's, and c's. Contradiction.

5. (15+15 Points) (i) Construct an efficient algorithm that, on input $\langle G, w\rangle$ where $G = (V, T, S, R)$ is a grammar in Chomsky Normal Form and w a string, decide whether $w \in L(G)$.
HINT: Use dynamic programming. For $1 \leq i \leq j \leq n$, let w_{ij} denote the substring $a_i \cdots a_j$ where $w = a_1, \ldots, a_n$. Define $V_{ij} = \{A \in V : A \Rightarrow^* w_{ij}\}$. How do you compute V_{ij} if you know the sets V_{ik}, V_{kj} for all $k = i, \ldots, j$?
(ii) What is the worst-case complexity of your algorithm, as a function of input sizes $m = |G|, n = |w|$? There is an interesting low-level issue here: $|w|$ and $|G|$ must be suitably interpreted. Note that V, T are arbitrary alphabets, but your algorithm must accept input with a fixed alphabet (say Σ). Hence use the following convention: assume Σ contains the special symbols $A, a, 0, 1$ (among others), and each symbol of V is encoded as a string of the form $A(0+1)^*$, and each symbol of T and w is encoded

as a string of the form $a(0+1)^*$. The definition of $|G|$ can be taken to be the number of symbols in writing down all the rules of G plus $|V \cup T|$. Then each symbol in x has length equal to its encoding as a string in $L(a(0+1)^*)$! Similarly, you need specify your encoding G over the fixed alphabet Σ and tell us how to determine its length $|G|$.

SOLUTION (i) (see Sipser's book p.241)
The dynamic programming technique here is standard. The idea is to construct a table $table(i,j)$ that contains the set of all variables A such that A can generate the substring $w[i,j]$. Here $w = w_1 \cdots w_n$ is the input and $w[i,j] = w_i \cdots w_j$. Recall that the empty string is handled in a special way in a Chomsky normal form grammar. Here is the algorithm M:

On input $w = w_1$ ... w_n:
1. If $w = \epsilon$ and $S \to \epsilon$ is a rule, accept.
2. For $i = 1$ to n,
3. For each variable A,
4. Test whether $A \to w_i$ is a rule.
5. If so, place A in $table(i,i)$.
6. For $l = 2$ to n,
7. For $i = 1$ to $n - l + 1$,
8. Let $j = i + l - 1$,
9. For $k = i$ to $j - 1$,
10. For each rule $A \to BC$,
11. If $table(i,k)$ contains B and $table(k+1,j)$ contains C, put A in $table(i,j)$.
12. If S is in $table(1,n)$, accept. Otherwise, reject.

SOLUTION (ii): Let $v = |V|, t = |T|$ and $r = |R|$. Clearly, $m = |G| \geq v + t + r$. The input length is $O(n \log v)$ since each symbol is encoded in length $O(\log v)$. Stages 4 and 5 run in $O(vn)$ times. Stage 6 runs at most n times. Each time stage 6 runs, stage 7 runs at most n times. Each time stage 7 runs, stages 8 and 9 run at most n times. Each time stage 9 runs, stage 10 runs r times. Thus stage 11, the inner loop of the algorithm, runs $O(r \times n^3)$. Summing the total shows that M executes in time $O(f(m) \times n^3)$ where $f(m)$ is the time in the inner loop. We can easily make sure that $f(m) = O(m^3)$, and with a little case $f(m) = O(m^2)$.