CSE105 Homework 3 Solutions Fall, 2001

3.9.a

Consider the language $L=\{a^nb^nc^n:n\geq 0\}$. We already know that this is not a CFL by Example 2.20; hence, the standard 1-PDA cannot accept L. However, a 2-PDA can accept L by storing *a*'s on one stack, *b*'s on the other, and then popping one *a* and one *b* for each remaining *c* of the input. If both stacks are empty after the last *c* is read, then w is accepted.

3.9.b

We show that two stacks can simulate a TM, an extra stack does not lead to a more powerful automaton. The extra stack can easily be presented by another tape on the TM. By theorem 3.8, any k-tape TM is equivalent to a single tape TM, therefore 3-PDAs and 2-PDAs are equivalent in power.

The TM by 2-PDA simulation is done as follows:
Stack 1 would represent the tape contents to the left of the current head position of the TM, while stack 2 would be the tape contents to the right of the current head position with the current symbol on the top of the stack. The two stack contents would change accordingly as the head moved across the tape left or right.

3.14

Notation for a), b) and e): for any two decidable languages L_1 and L_2, let M_1 and M_2 be the TMs that decide them.
Notation for c) and d): for any decidable language L, let M be the TM that decides it.

a. We construct a TM M' that decides the union of L_1 and L_2:
"On input w:
1.Run M_1 on w, if it accepts, accept.
2.Run M_2 on w, if it accepts, accept. Otherwise, reject."
M' accepts w if either M_1 or M_2 accepts it. If both reject, M' rejects.

b. We construct a NTM M' that decides the concatenation of L_1 and L_2:
"On input w:
1. For each way to cut w into two parts $w=w_1w_2$:
2. Run M_1 on w_1.
3. Run M_2 on w_2.
4. If both accept, accept. Otherwise, continue with the next w_1,w_2.
5. All cuts have been tried without success, so reject."
We try every possible cut of w. If we ever come across a cut such that the first part is accepted by M_1 and the second part is accepted by M_2, w is in the concatenation of L_1 and L_2. So M' accept w. Other wise, w does not belong to the concatenation of the language and is rejected.

c. We construct a NTM M' that decides the star of L:
"On input w:
1.For each way to cut w into parts so that $w=w_1w_2\ldots wn$:
2.Run M on wi for i=1,2,…,n. If M accepts each of these string w_i, accept.
3.All cuts have been tried without success, so reject."
If there is a way to cut w into different substrings such that every substring is accepted by M, w belongs to the star of L and thus M' accepts w. Otherwise, w is rejected.

Since there are finitely many possible cuts of w, M' will halt after finitely many steps.

d. We construct a TM M' that decides the complement of L:
"On input w:
Run M on w. If M accepts, reject; if M rejects, accept."
Since M' does the opposite of whatever M does, it decides the complement of L.

e. We construct a TM M' that decides the intersection of L_1 and L_2:
"On input w:
1.Run M_1 on w, if it rejects, reject.
2.Run M_2 on w, if it accepts, accept. Otherwise, reject."
M' accepts w if both M_1 and M_2 accept it. If either of them rejects, M' rejects w, too.

3.15
For any two Turing-recognizable language L_1 and L_2, let M_1 and M_2 be the TMs that recognize them.

a. We construct a TM M' that recognizes the union of L_1 and L_2:
"On input w:
Run M_1 and M_2 alternatively on w step by step. If either accept, accept. If both halt and reject, then reject."
If any of M_1 and M_2 accept w, M' will accept w since the accepting TM will come to its accepting state after a finite number of steps. Note that if both M_1 and M_2 reject and either of them does so by looping, then M will loop.

b. We construct a NTM M' that recognizes the concatenation of L_1 and L_2:
" On input w:
1.Nondeterministically cut w into two parts $w=w_1w_2$.
2.Run M_1 on w_1. If it halts and rejects, reject. If it accepts, go to stage 3.
3.Run M_2 on w_2. If it accepts, accept. If it halts and rejects, reject."
If there is a way to cut w into two substrings such M_1 accepts the first part and M_2 accepts the second part, w belongs to the concatenation of L1 and L2 and M' will accept w after a finite number of steps.

c. For any Turing-recognizable language L, Let M be the TM that recognizes it. We construct a NTM M' that recognizes the star of L:
" On input w:
1.Nondeterministically cut w into parts so that $w=w_1w_2\ldots w_n$.
2.Run M on wi for all i. If M accepts all of them, accept. If it halts and rejects any of them, reject."
If there is a way to cut w into substrings such M accepts all the substrings, w belongs to the star of L and M' will accept w after a finite number of steps.

d. We construct a TM M' that recognizes the intersection of L_1 and L_2:
"On input w:
1.Run M_1 on w. If it halts and rejects, reject. If it accepts, go to stage 3.
2.Run M_2 on w. If it halts and rejects, reject. If it accepts, accept."
If both of M_1 and M_2 accept w, w belongs to the intersection of L_1 and L_2 and M' will accept w after a finite number of steps.

4.7
Suppose B is countable and a correspondence f: $N \rightarrow B$ exists. We construct x in B that is not paired with anything in N. Let $x=x_1x_2...$ Let $x_i=0$ if $f(i)_i=1$, and $x_i=1$ if $f(i)_i=0$ where $f(i)_i$ is the ith bit of $f(i)$. Therefore, we ensure that x is not $f(i)$ for any i because it differs from $f(i)$ in the ith symbol, and a contradiction occurs.

4.10 Show that A is decidable, where
A = {<M> | M is a DFA which does not accept any string containing an odd number of 1's}.

Prove by construction.
Construction: the following TM M_A decides A.
On input <M> where M is a DFA:
1. Construct a DFA G that accepts strings containing an odd number of 1's.
2. Construct a DFA F such that $L(F) = L(M) \cap L(G)$.
3. Run TM T from Theorem 4.4 on input <F>, where T decides E_{DFA}
4. If T accepts, accept. If T rejects, reject.

M_A decides A:
a). If $x \in A$, then x is a DFA which does not accept any string containing an odd number of 1's. Then $L(x) \cap L(G) = \phi = L(F)$. Therefore TM T on input <F> will accept, so M_A accepts.
b). If $x \notin A$, then x is a DFA which accept some string containing an odd number of 1's. Then $L(F) = L(x) \cap L(G) \neq \phi$. Therefor TM T on input <F> rejects, so M_A rejects.

From a) and b) above, we have shown that M_A decides A.

5.4
If $A \leq_m B$ and B is a regular language, does that imply that A is a regular language? Why or why not?

No, that does not imply that A is regular.
For example, $\{a^nb^n \mid n \geq 0\} \leq_m \{a^n \mid n \geq 0\}$.
The reduction first tests whether its input is a member of $\{a^nb^n \mid n \geq 0\}$. If so, it outputs the string a, and if not it outputs the string b.

5.12 Let S={<M>| M is a TM that accepts w^R whenever it accepts w}.
Show that S is undecidable.

We show that $A_{TM} \leq_m S$ by mapping <M, w> to <M'> where M' is the following TM:
On input x:
1. If $x = 01$ then accept.
2. If $x \neq 10$ then reject.
3. If $x = 10$, then simulate M on w. If M accepts w then accept. If M halts and rejects w, then reject.

If $< M, w> \in A_{TM}$, then M accepts w, and $L(M') = \{01,10\}$, so $<M'> \in S$.
If $< M, w> \notin A_{TM}$, then M rejects w, and $L(M') = \{01\}$, so $<M'> \notin S$.
Therefore, $< M, w> \in A_{TM} \Leftrightarrow <M'> \in S$.
Since A_{TM} is undecidable, so is S.

7.6

P is closed under union.

For any two P-language L_1 and L_2, let M_1 and M_2 be the TMs that decide them in polynomial time. We construct a TM M' that decides the union of L_1 and L_2 in polynomial time:

M'= "On input <w>":

1. Run M_1 on w. If it accepts, accept.
2. Run M_2 on w. If it accepts, accept. Otherwise, reject."

M' accepts w if and only if either M_1 and M_2 accept w. Therefore, M' decides the union of L_1 and L_2. Since both stages take polynomial time, the algorithm runs in polynomial time.

P is closed under concatenation.

For any two P-language L_1 and L_2, let M_1 and M_2 be the TMs that decide them in polynomial time. We construct a TM M' that decides the concatenation of L_1 and L_2 in polynomial time:

M'= "On input<w>:

1. For each way cut w into two substrings w=w1w2:
2. Run M_1 on w1 and M_2 on w2. If both accept, accept.
3. If w is not accepted after trying all the possible cuts, reject."

M' accepts w if and only if w can be written as w_1 w_2 such that M_1 accepts w_1 and M_2 accepts w_2. Therefore, M' decides the concatenation of L_1 and L_2. Since stage 2 runs in polynomial time and is repeated at most O(n) times, the algorithm runs in polynomial time.

P is closed under complement.

For any P-language L, let M be the TM that decides it in polynomial time. We construct a TM M' that decides the complement of L in polynomial time:

M'= "On input <w>:

1. Run M on w.
2. If M accepts, reject. If it rejects, accept."

M' decides the complement of L. Since M runs in polynomial time, M' also runs in polynomial time.

7.7

NP is closed under union and concatenation.

We refer to the two languages as A and B, and TM M_A and M_B are the Non-Deterministic Turing Machines that decide them in poly time.

Union:

We will construct a non-deterministic Turing machine C, that decides A union B.

TM C = "On input x

1. Non-Deterministically decide whether to check if x is a member of A or B.
2. If A was decided, run M_A on x, and output M_A 's decision.
   If B was decided, run M_B on x, and output M_B 's decision."

TM C will run in $O(1) + MAX (O(M_A), O(M_B)) = MAX (O(M_A), O(M_B))$, which must by poly time, since M_A and M_B run in poly-time. Because TM C decides A union B in poly time, A union B is in NP.

Concatenation:
The concatenation of Languages A and B will contain strings that start with a string that is an element of A and end with a string that is an element of B. We will construct a non-deterministic Turing machine C, that decides A concatenate B.

TM C = "On input x
1. Non-Deterministically divide x into y and z.
2. Run M_A on y.
3. Run M_B on z.
4. If both M_A and M_B accepted, accept; else reject."

TM C will run in $O(1) + O(M_A) + O(M_B) = MAX (O(M_A), O(M_B))$, which must by poly time, since M_A and M_B run in poly-time. Because TM C decides A concatinate B in poly time, A concatinate B is in NP.

Problem 10.
L = {<M1,M2>| M1 and M2 are TMs such that for some input x, both M1 and M2 halt on x}.

a) Prove that L is Turing-recognizable by constructing enumerators EM1 and EM2 that enumerates all the strings in L(M1) and L(M2). The instructions on how to create this machine are on page 141.

We will now construct a TM M, that recognizes L.
M = "On input <M1, M2>,
1. Repeat the following for i = 1, 2, 3,...
2. Simulate EM1 for i steps, record any output strings on the tape.
3. Simulate EM2 for i steps, record any output strings on the tape.
4. Compare the output of EM1 and EM2. If the same string appears on both lists, accept."

b) Show that L is undecidable.
We show the complement of E_{TM}, namely $\bar{E}_{TM}$, can be mapping reduced to L. (E_{TM} is the language of all Turing Machines whose language is the empty set, page 173.)

Assume thare is a TM *N* that decides *L*. We will construct a TM *S* that decides $\bar{E}_{TM}$. Let *Acc* be the Turing machine that accepts all its inputs.

S = "On Input <*M*>,
1. Construct <*M', Acc*>, where *Acc* is a TM accepting everything and *M'* is the following TM:
"On input *x*, simulate *M* on *x*.
If *M* accepts *x*, then accept.
Otherwise, if M halted, then enter an infinite loop."
2. Run *N* on <*M, Acc*>.
3. If *N* accepts, reject; else accept."

If $<M, Acc> \in L$, then *M* and *Acc* halt on the some input, $L(M) \neq \phi$, so $<M> \in \bar{E}_{TM}$.
If $<M, Acc> \notin L$, then *M* and *Acc* do not halt the same input, $L(M) = \phi$, $<M> \notin \bar{E}_{TM}$.
Therefore, $<M, Acc> \in L \Leftrightarrow <M> \in \bar{E}_{TM}$.
Since $\bar{E}_{TM}$ is undecidable, so is L (by Corollary 5.17).