15-453 Formal Languages, Automata, and Computation

Homework 5, due March 3

Problem 1 (35 points)

For each of the following languages, classify them as one of:

- regular;
- context-free but not regular;
- Turing recognizable but not context free.

Argue why you classified them this way. Your explanations should be clear, but they do not need to be detailed proofs.

1. $\{w \mid w$ has 1,425,315 more a's than b's$\}$. $\Sigma = \{a, b\}$. context-free
2. $\{a^p \mid p$ is prime$\}$. Here $\Sigma = \{a\}$. turing
3. $\{w \mid$ There is the same number of 01s and 10s in $w\}$. For example 1001 is in this language, because we have one pair of each, but 1010 is not, because it has two 10s and only one 01. $\Sigma = \{0, 1\}$. regular

Example of a correct argument: The language $L = \{ww^R \mid w \in \{a, b\}^*\}$ is a context-free language, but not a regular language. It is a context free language because it can be generated by the grammar $S \to aSa \mid bSb \mid \epsilon$. It can be shown not to be regular, by applying the pumping lemma for regular languages to the word $0^p110^p \in L$, with p the pumping length.

Example of an insufficient argument: The language L is context-free because we can build a PDA accepting it, but it is not regular, because NFAs cannot remember symbols.

basically for a's delete the first a then skip the second and so on(rounding down for odd no. of a's). then erase a b when you encounter one.

Problem 2 (30 points)

Show the state diagram of a Turing Machine which accepts the following language: $L = \{\#a^n b^m \mid m, n > 1, \lfloor \log_2 n \rfloor = m - 1\}$. We have marked the left end of the tape for your convenience. Also explain your construction in words. **Hint:** this problem is very closely related to Example 3.4 from the book. The relation can be written $2^{m-1} \leq n < 2^m$. You can count how many times you can halve the number of a's, and this should give the number of b's. However, you should draw the complete state diagram, and not just give the explanation in words.

Problem 3 (35 points)

for union, we'll construct a turing machine m3 which has two tracks, one for m1 and the other for m2. m3 will first execute m1 on the first track. if m1 accepts, m3 accepts. If m1 rejects, m2 is simulated. if m2 accepts, m3accepts. otherwise m3 rejects.

[From problem π (3.14) in the book.] Show that the Turing-decidable languages are closed under the following operations:

1. Union.
2. Complementation.

m3 is the same as m1, except that the accept and reject states are swapped

Prove carefully your answers. **Hint:** use the fact that TMs with multiple tapes are equivalent to deterministic TMs.

FYI: the Turing-decidable languages are also closed under concatenation, star and intersection, but you are not required to prove these facts now.

Problem 4 (extra credit 30 points)

We define a new type of DFA, which has besides the input tape a work-tape of finite length. For $k > 0$, a machine in the class DFA(k) has the input string on a read-only tape, and it has a read/write work tape of fixed length k. The DFA(k) can move the head on the work tape both ways, (the input tape can only be read from left to right). Notice that the usual DFAs are just the class DFA(0).

1. Formalize the notion of a DFA(k) machine clearly, by defining the states, alphabets and transition function. **Note:** there may be multiple ways to do this; just choose one of them which seems reasonable; for instance, we did not specify what happens when the DFA(k) reaches either end of the work tape.

2. Prove that for any k, the languages recognized by the machines in DFA(k) are exactly the regular languages. **Note:** You **must** give a rigorous proof in order to receive the extra credit.

We can conclude that giving to a DFA just a constant amount of scratch space is not useful for increasing its computational power.

formal definition:
(Q, A, T,k,left end marker, right end marker, D, S, F, B)
Q: finite set of states, A: input alphabet, T: tape alphabet, such that the input alphabet is its subset
k:scratch length, left end marker: symbol of tape alphabet such that it is present at the left end of the tape,
right end marker: symbol of tape alphabet such that it is present at the right end of the tape, and the difference
between positions is k+1(there are k blank spaces in the middle.
S: start state
F: accept states
B: blank symbol belonging to tape alphabet - input alphabet
D: transition function that takes the input state, input alphabet, tape alphabet: writes something on that tape cell,
moves either left or write, moves one step ahead on the input, changes state

we claim that the information stored in states + tape can be stored in the finite control itself. This is because the tape alphabet can have a limited no. of configurations : (LEF)*(TAPE ALPHABET)^k*(REM)*{0,1,2,...,k+1}. Thus a new DFA may have the actual states Q' as (Q, tape configuration) with the start state being (S, (lem)(B)^k(rem)(0))