COT 4210

Spring 2012

Final Exam Solution

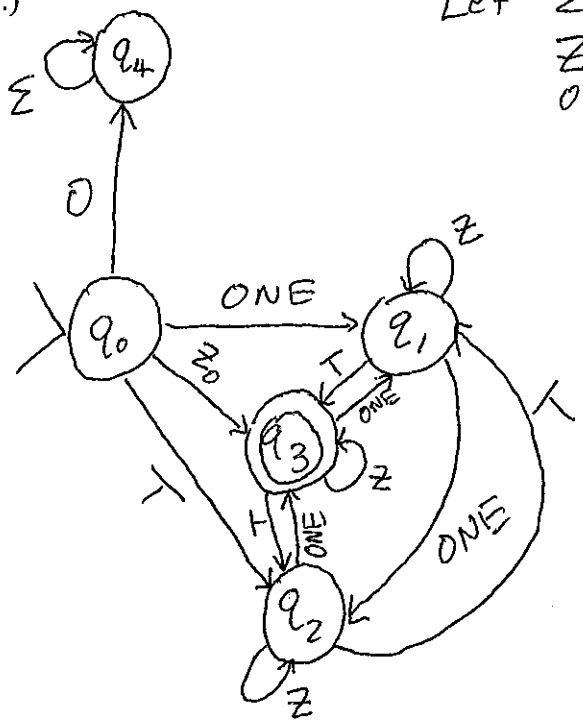
Date: 4/24/2012

Lecturer: Arup Guha

1) (10 pts) Regular Languages

(a) (5 pts) Draw an DFA over the alphabet {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} that accepts all non-empty strings without any leading zeros that represent positive integers that have a value divisible by 3. (Note: for example, 3, 27, 84 and 101121 should be accepted but ε, 0, 0036 and 17 should not be accepted.)

Let $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$,
$Z = \{0,3,6,9\}$, $Z_0 = \{3,6,9\}$
ONE = {1, 4, 7}
T = {2, 5, 8}



GRADING
1pt - Start state ≠ ACCEPT
1pt - 0 goes to dead state
1pt - Having 3 other states corresponding to 0 mod 3, 1 mod 3, 2 mod 3
2pts - all 9 transitions

(b) (5 pts) Let L be the language over the alphabet {a, b, c, d} that is the set of strings that have an even number of a's. (For example, bccdc, aababcdab and dbbcdabacd are all in L, but bdcabcd and badcaa are not.) Write down a regular expression for L.

$$(b \cup c \cup d)^*\left((b \cup c \cup d)^* a (b \cup c \cup d)^* a (b \cup c \cup d)^*\right)^*$$

GRADING
1pt - allows ε
1pt - allows all non-empty strings w/0 a's.
1pt - allows strings w/even # of a's that don't start with a.
1pt - allows strings w/even # of a's that don't end with a.
1pt - doesn't allow any strings with an odd number of a's.

2) (10 pts) Context-Free Languages

(a) (4 pts) Consider all of the arithmetic expressions that could be created by the variable x, the operator + and both open and close parentheses. Create a Context Free Grammar with the alphabet { x, +, (,) } that generates all such valid expressions.

Let S be the Start Variable in our grammar. A grammar that satisfies the requirements is as follows:

$S \rightarrow x \mid (S) \mid S + S$

Basically, we can add any two valid expressions or group any single valid expression. Atomically speaking, x is the only arithmetic expression allowed in this alphabet.

Grading: 1 pt for having a start variable, 1 pt for allowing a basic x and no epsilon, 1 pt for expressions with an arbitrary number of +s, 1 pt for expressions that can be parenthesized. Take off 1 pt if you find a string that is incorrectly classified.

(b) (6 pts) If C is a Context-Free Language and R is a regular language, then $C \cap R$ is a Context-Free Language. (This is true and you don't need to prove it for this question. Rather, you'll need to use this fact to answer the following question.) Consider the language A defined over the alphabet {a, b, c} as follows: A = { w | w contains an equal number of a's, b's and c's}. Prove that A is NOT a Context-Free Language.

Assume, to the contrary, that A is context free. Let L the langauge described by the regular expression $a^*b^*c^*$. Based on the theorem stated above, since A is context free and L is regular, it follows that $A \cap L$ is context free. This intersection is clearly equal to { $a^n b^n c^n$ | n $\geq$ 0 }, since A forces all strings to have the same number of strings and L forces all a's to precede all b's and all b's to precede all c's. But, as was proved in class and the textbook, this particular language is not context-free, which provides our contradiction. It follows that our initial assumption, that A is context free is incorrect. Thus, A is NOT context free, as desired.

Grading: 1 pt for setting up proof by contradiction. 2 pts for picking A and $a^*b^*c^*$ as the two languages to use, 1 pt for deducing that this intersection should be context free, 1 pt for figuring out that this intersection is really $a^n b^n c^n$, and 1 pt for the conclusion.

3) (10 pts) Let L = { <M> | M is a Turing Machine such that L(M) only contains even-length strings}. Prove that L is undecidable without using Rice's Theorem.

Assume to the contrary, that L is decidable. Then there exists a Turing Machine that decides membership in L. Let this Turing Machine be X.

We will now create Y, a decider for A_{TM} as follows:

Y(TM M, input w)

1. **Create a new Turing Machine M' such that if its input is an odd number of characters greater than one, it automatically rejects. If its input is a single character, erase the contents of the tape and write w, and then run the steps of M. If its input is an even length string, just run the steps of M on it.**

2. **Call X(M'). Since X is a decider, it's guaranteed to return an answer.**

3. **If X returns true, then return false, indicating that M does not accept w. If X returns false, return true, indicating that M does accept w.**

Basically, if M accepts w, then L(M') will include all inputs of length 1. Thus, <M'> will NOT be a member of L. Alternatively, if M does not accept w, all inputs of odd length will NOT get accepted by M', meaning that <M'> is an element of L.

But, we know that A_{TM} is undecidable. This means that our initial assumption must be false and L is undecidable.

Grading:

1 pt - Assume to the contrary...
1 pt - Give a name to the decider of L
2 pts - set up a decider for A_{TM} or another known undecidable language.
3 pts - creating a new TM to pass as an argument to the decider of L
2 pts - determining how to call the decider for L and how to use its answer to solve A_{TM} or the language chosen
1 pt - wrapping up the proof

4) (10 pts) Utilizing the polynomial-time reduction of 3SAT $\leq_P$ SUBSET-SUM shown in the text book, show the result of that reduction on the 3SAT instance shown below:

$$(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee \overline{x_1} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_4} \vee x_2) \wedge (x_1 \vee \overline{x_3} \vee x_4) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4})$$

In particular, your answer should be a set of integers and a target value, T.

	1	2	3	4	c1	c2	c3	c4	c5	c6		Set Values
y1	1	0	0	0	1	0	0	1	0	0		1000100100
z1	1	0	0	0	0	1	0	0	1	0		1000010010
y2	0	1	0	0	0	1	1	0	0	0		0100011000
z2	0	1	0	0	1	0	0	0	1	1		0100100011
y3	0	0	1	0	1	1	1	0	0	0		0010111000
z3	0	0	1	0	0	0	0	1	0	1		0010000101
y4	0	0	0	1	0	0	0	1	1	0		0001000110
z4	0	0	0	1	0	0	1	0	0	1		0001001001
g1	0	0	0	0	1	0	0	0	0	0		0000100000
h1	0	0	0	0	1	0	0	0	0	0		0000100000
g2	0	0	0	0	0	1	0	0	0	0		0000010000
h2	0	0	0	0	0	1	0	0	0	0		0000010000
g3	0	0	0	0	0	0	1	0	0	0		0000001000
h3	0	0	0	0	0	0	1	0	0	0		0000001000
g4	0	0	0	0	0	0	0	1	0	0		0000000100
h4	0	0	0	0	0	0	0	1	0	0		0000000100
g5	0	0	0	0	0	0	0	0	1	0		0000000010
h5	0	0	0	0	0	0	0	0	1	0		0000000010
g6	0	0	0	0	0	0	0	0	0	1		0000000001
h6	0	0	0	0	0	0	0	0	0	1		0000000001
tar	1	1	1	1	3	3	3	3	3	3	TARGET	1111333333

Grading: 1 pt - having some sort of chart with rows and columns
3 pts - having 20 rows
2 pts - having 10 columns
3 pts for the values in the table
1 pt - for the target value

5) (15 pts) The Amazing-Race problem is as follows:

You are given a trek of n segments, s_i ($1 \leq i \leq n$), for each segment, you may choose two possible paths, one which takes a_i minutes and another which takes b_i minutes. The goal of the Amazing-Race is to choose one of the two possible paths for each segment such that the total amount of time it takes to traverse each of these paths is exactly equal to a target time T minutes. The input to the Amazing-Race problem can be represented as and ordered list of n ordered pairs (a_i, b_i) and a target value T where each value is a positive integer, $0 < a_i < b_i$, and $\sum_{i=1}^{n} a_i \leq T \leq \sum_{i=1}^{n} b_i$. An instance of n ordered pairs (a_i, b_i) and a target value T belongs in Amazing-Race if and only if there exists a sequence of values t_1, t_2, t_3, ..., t_n where $t_i \in \{a_i, b_i\}$ for all i, $1 \leq i \leq n$ where $\sum_{i=1}^{n} t_i = T$.

Prove that the Amazing-Race problem is NP-Complete by reducing Subset Sum to it.

First, note that Amazing-Race is in NP. Given a certificate with a list of which path to choose for each segment, one can verify in polynomial time that the sum of the times it takes to traverse each of the selected paths adds up to exactly T minutes. (0 pts)

Now, we will show Amazing-Race is NP-Complete by showing that Subset Sum $\leq_P$ Amazing-Race. Consider an arbitrary instance of Subset Sum: $\{s_1, s_2, s_3, ..., s_n\}$ and a target X. (2 pts) From this instance, we will create an Amazing-Race instance that is satisfiable if and only if the original Subset Sum instance was sastisfiable as follows:

Let $a_i = 1$, and let $b_i = 1 + s_i$ for all i, $1 \leq i \leq n$. Finally set $T = n + X$. (7 pts)

Consider any situation where the Subset Sum instance is satisfiable. Namely, let s_{n1}, s_{n2}, .. s_{nk} sum to X. Now, consider choosing the analogous values for path b, and the rest path a. Namely, choose b_{n1}, b_{n2}, ..., b_{nk}, and choose a_i, for all $i \notin \{n1, n2, ..., nk\}$. By substituting, we find the sum of these paths to be $1 + 1 + 1 + ... + 1 + s_{n1} + 1 + s_{n2} + ... 1 + s_{nk} = n + s_{n1} + s_{n2} + .. + s_{nk} = n + X$, showing that the Amazing-Race instance is satisfiable. (4 pts) Similarly, if an Amazing-Race instance created in this manner is satisfiable, we can use each of the b paths chosen to determine a satisfiable instance of the corresponding subset sum problem. This completes the reduction and proves that Amazing-Race is NP-Complete. (2 pts)

In layman's terms, we have made a one to one correspondence between the items in the subset sum instance and the DIFFERENCE in times of the two options for each path in the Amazing-Race instance. Each choice of path b corresponds to choosing a number for the subset in subset sum and vice versa.

6) (5 pts) How many calories are in a single serving of Nabisco 100 calorie snap packs? **100**