Finite Automata and Formal Languages

TMV026/DIT321

Friday 27th of August 2010

CTH: Total 60 points: $\geq$ 26: 3, $\geq$ 38: 4, $\geq$50: 5
GU: Total 60 points $\geq$26: G, $\geq$42: VG

No help material.

Answers can be written in English or Swedish. Write as clear as possible.

All answers should be well motivated. Points will be deduced when you give an unnecessarily complicated solution or when you do not properly justify your answer.

1. (5pts) Prove the following statement using mathematical induction:

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$$

   Do not forget to clearly state the base case and the inductive hypothesis!

2. (3.5pts) Define a deterministic finite automata accepting the language over $\{0,1\}$ not containing the sub-word 0101.

3. (4.5pts) Convert the following non-deterministic finite automata with ϵ-transitions to an equivalent deterministic finite automata.

	a	b	ϵ
$\rightarrow q_0$	$\{q_1\}$	$\emptyset$	$\{q_2\}$
q_1	$\{q_1\}$	$\{q_2, q_3\}$	$\emptyset$
q_2	$\{q_3\}$	$\{q_2\}$	$\emptyset$
$^*q_3$	$\emptyset$	$\emptyset$	$\emptyset$

4. (5pts) Compute, by eliminating the states in the automaton, a regular expression that generates exactly the language accepted by the following deterministic finite automata.

	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_3	q_2
q_2	q_1	q_3
$^*q_3$	$-$	$-$

5. Give regular expressions that generate exactly the following languages. Justify your answer.

   (a) (2pts) $\mathcal{L}((a+b)a^*) \cap \mathcal{L}(baa^*)$.

   (b) (2pts) $\{0,1\}^* - \mathcal{L}((0+1)^*10)$.

   (c) (2pts) The words over $\{0,1\}$ that contain exactly one pair of consecutive 0's.

6. (5pts) Do these two regular expressions represent the same language? Justify your answer.

   (a) $b + ab^* + aa^*b + aa^*ab^*$ and $a^*(b + ab^*)$?

   (b) $(ab + a)^*ab$ and $(aa^*b)^*$?

7. (a) (1.5pts) What is formally a regular language?

   (b) (1.5pts) Explain informally why the language $\{0^n1^n2^n \mid n > 0\}$ is not regular.

   (c) (3pts) State the Pumping lemma for regular languages and use it to formally prove that the language $\{0^n1^n2^n \mid n > 0\}$ is not regular.

8. (a) (4.5pts) Write a context-free grammar that generates the language

$$\{a^mb^nc^k \mid (k = m + n, k > 0) \text{ or } (n = k + m, n > 0)\}$$

   Explain the grammar

   (b) (1.5pts) Is this grammar ambiguous? Justify.

9. Consider the following context-free grammar with starting symbol S:

$$\begin{array}{ll} S \to a \mid aA \mid B \mid C & \\ A \to aB \mid \epsilon & B \to Aa \\ C \to cCD & D \to ddd \end{array}$$

   (a) (7.5pts) Simplify the grammar by successively applying the following steps (that is, steps ii. to iv. should be performed to the grammar obtained in steps i. to iii. respectively):

      i. Identify and eliminate the ϵ-productions.

      ii. Identify and eliminate the unit productions.

      iii. Identify and eliminate the non-generating symbols.

      iv. Identify and eliminate the non-reachable symbols.

   (b) (2pts) Which is the language generated by this grammar?

   (c) (1.5pts) Is this grammar ambiguous? Justify.

10. (4pts) Consider the following context-free grammar G with starting symbol S:

$$\begin{array}{ll} S \to AB \mid BC & A \to BA \mid a \\ B \to CC \mid b & C \to AB \mid a \end{array}$$

   Apply the CYK algorithm to determine whether $baaab \in \mathcal{L}(G)$ or not. Show the resulting table and justify your answer.

11. (4pts) Consider the following Turing machine defined by the tuple $(\{q_0, q_1\}, \{0, 1, a, b\}, \delta, q_0, \square, \{q_1\})$, where the transition function δ is as following:

$$\begin{array}{l} \delta(q_0, 0) = (q_0, a, \mathsf{R}) \\ \delta(q_0, 1) = (q_0, b, \mathsf{R}) \\ \delta(q_0, \square) = (q_1, \square, \mathsf{R}) \end{array}$$

   What does this Turing machine do? When does it stop? Explain as much as you can.

Exam Solutions

In the exam, you need to explain a bit more your solutions.

1. Base case: n = 0. Here we need to prove that $\sum_{i=0}^{0} i = \frac{0(0+1)}{2}$. That is, we need to prove that $0 = 0$, which is certainly true.

   Inductive step: We assume the statement for a given n and then prove the statement for $n+1$.

   So our inductive hypothesis (IH) is that $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$ is true.

   We need to show that $\sum_{i=0}^{n+1} i = \frac{(n+1)(n+1+1)}{2}$, that is, $\sum_{i=0}^{n+1} i = \frac{(n+1)(n+2)}{2}$ is also true.

   We have that $\sum_{i=0}^{n+1} i = \sum_{i=0}^{n} i + (n+1)$.

   Using our IH we have that $\sum_{i=0}^{n+1} i = \frac{n(n+1)}{2} + (n+1)$.

   Applying math. properties we have

   $$\sum_{i=0}^{n+1} i = \frac{n(n+1)}{2} + \frac{2(n+1)}{2} = \frac{(n+2)(n+1)}{2} = \frac{(n+1)(n+2)}{2}$$

   as desired.

2.

	0	1
$\rightarrow^* q_0$	q_1	q_0
$^*q_1$	q_1	q_2
$^*q_2$	q_3	q_0
$^*q_3$	q_1	q_4
q_4	q_4	q_4

3.

	a	b
$\rightarrow q_0q_2$	q_1q_3	q_2
$^*q_1q_3$	q_1	q_2q_3
q_1	q_1	q_2q_3
q_2	q_3	q_2
$^*q_2q_3$	q_3	q_2
$^*q_3$	$-$	$-$

4. If we first eliminate q_2 we obtain an intermediate automata with 3 states (q_0, q_1 and q_3) such that:

   - We go from q_0 to q_1 with the RE $a + ba$.
   - We go from q_0 to q_3 with the RE bb.
   - We go from q_1 to q_1 with the RE ba.
   - We go from q_1 to q_3 with the RE $a + bb$.

   After we eliminate q_1 we are able to obtain the final RE: $bb + (a + ba)(ba)^*(a + bb)$

   Note: We could have eliminated q_1 first and then q_2 and then obtained the (equivalent) RE $aa + (b + ab)(ab)^*(b + aa)$.

5. (a) $\mathcal{L}((a+b)a^*)$ is the language $\mathcal{L}(a^+) \cup \mathcal{L}(ba^*)$. Then $(\mathcal{L}(a^+) \cup \mathcal{L}(ba^*)) \cap \mathcal{L}(baa^*) = \mathcal{L}(baa^*)$, hence the resulting RE is baa^*.

   (b) $\mathcal{L}((0+1)^*10)$ is the language over $\{0,1\}$ with words ending in 10. Hence we want the set of words over $\{0,1\}$ which do not end in 10. The RE is then $(1+0)^*1 + (1+0)^*00 + 0 + \epsilon$.

   (c) The REs $(1+01)^*$ and $(1+10)^*$ generate words with no consecutive 0's hence the resulting RE is $(1+01)^*00(1+10)^*$.

6. (a) (3.5pts) $b + ab^* + aa^*b + aa^*ab^*$ and $a^*(b + ab^*) = a^*b + a^*ab^*$ represent the same language. We show the equivalence by explaining that the words generated by the RE on the left are also generated by the RE on the right, and vise versa.

   Let $w \in b + ab^* + aa^*b + aa^*ab^*$. Each of the possible forms that w can get can also be generated by $a^*b + a^*ab^*$: b and aa^*b by a^*b, and ab^* and aa^*ab^* by a^*ab^*.

   Let $w \in a^*b + a^*ab^*$. a^*b is generated by b (if not a is present) or by aa^*b (if at least 1 a is present). a^*ab^* is generated by ab^* (if only 1 a is present) or by aa^*ab^* (if more than 1 a are present).

   (b) (1.5pts) $(ab + a)^*ab$ and $(aa^*b)^*$ do not represent the same language. The RE $(ab + a)^*ab$ generates words whose length is at least 2 and end with ab while the RE $(aa^*b)^*$ generate a language which admits the empty word.

7. (a) Formally a regular language is such that it exists a deterministic finite automata which accepts the language. Since we have shown that DFA, NFA, ϵ-NFA and RE are equivalent then a regular language is such that there is a FA which accepts it or a RE that generates it.

   (b) Intuitively, a regular language is a language that needs only a finite memory in order to see if a certain word belongs to the language or not.

   We need $3n$ states to see if the word $0^n1^n2^n$ for a *given* n. Since $n > 0$ (hence it could grow unlimited) then we need an infinite number of states to recognise the language.

   (c) See the book or the slides of the course for the statement of the Pumping lemma for regular languages.

   Let us assume the language is regular. So the Pumping lemma gives as a certain n. Let $w = 0^n1^n2^n$. Then y will contain only 0's and hence xy^kz will contain more 0's than 1's and 2's for $k > 1$.

8. (a)

$$\begin{aligned}
&S \rightarrow P \mid Q\\
&P \rightarrow aPc \mid ac \mid D\\
&D \rightarrow bDc \mid bc\\
&Q \rightarrow ED \mid E \mid D\\
&E \rightarrow aEb \mid ab
\end{aligned}$$

   - S is the start symbols of the grammar: P generates the part $k = m + n, k > 0$ and Q generates the part $n = k + m, n > 0$.
   - P generates the part $k = m + n, k > 0$.
     aPc and ac introduce a c for each a. When and if we want to add b's we move to D.
   - D introduces a c for each b.
   - Q generates the part $n = k + m, n > 0$.
   - E introduces a b for each a.

   (b) When $k = n$ (and hence $m = 0$) the grammar is ambiguous. Consider the 2 left-most derivations of b: $S \Rightarrow P \Rightarrow D \Rightarrow bc$ and $S \Rightarrow Q \Rightarrow D \Rightarrow bc$.

9. (a) i. (2pts) The only ϵ-production is $A \to \epsilon$. After we eliminate it we obtain

$$\begin{array}{ll} S \to a \mid aA \mid B \mid C & \\ A \to aB & B \to Aa \mid a \\ C \to cCD & D \to ddd \end{array}$$

ii. (2pts) The unit productions are $S \to B$ and $S \to C$. After we eliminate them we obtain

$$\begin{array}{ll} S \to a \mid aA \mid Aa \mid cCD & \\ A \to aB & B \to Aa \mid a \\ C \to cCD & D \to ddd \end{array}$$

iii. (2pts) The only non-generating symbol is C. After we eliminate it we obtain

$$\begin{array}{ll} S \to a \mid aA \mid Aa & \\ A \to aB & B \to Aa \mid a \\ & D \to ddd \end{array}$$

iv. (1.5pts) The only non-reachable symbol is D. After we eliminate it we obtain

$$\begin{array}{ll} S \to a \mid aA \mid Aa & \\ A \to aB & B \to Aa \mid a \end{array}$$

(b) The grammar generates sequences of a's with an odd number of a's, that is, $a(aa)^*$.

(c) Yes, the word aaa can be generated in the 2 following ways: $S \Rightarrow aA \Rightarrow aaB \Rightarrow aaa$ and $S \Rightarrow Aa \Rightarrow aBa \Rightarrow aaa$. Since both are left-most derivations it shows that the grammar is ambiguous.

10.

$\{S, C\}$				
$\{S, A, C\}$	$\{S, C\}$			
$\emptyset$	$\{S, A, C\}$	$\{B\}$		
$\{S, A\}$	$\{B\}$	$\{B\}$	$\{S, C\}$	
$\{B\}$	$\{A, C\}$	$\{A, C\}$	$\{A, C\}$	$\{B\}$
b	a	a	a	b

S belongs to the upper-most set, which means that the word is generated by the grammar since S is the starting symbol of the grammar.

11. The Turing machine transforms sequences of (only) 0's and 1's into sequences of a's and b's respectively.

Example: When the word 00101 is on the tape, the Turing machine returns the word $aabab$.

Example: If the word $00a10b1$ or any other word containing a's or b's is on the tape, the Turing machine gets stuck since there is no instruction that deals with this situation.

The first 2 instructions translate 0's into a's and 1's into b's, respectively. The third instruction recognises the end of the input (eventually the empty input) and moves the machine to the final state.

The machine stops when it reads a $\square$ from the initial state, hence, when the input is empty or when it finishes reading a sequence of only 0's and 1's.