# Finite Automata Theory and Formal Languages

TMV026/TMV027/DIT321 – Responsible: Ana Bove

Tuesday 28 of May 2013
Total: 60 points

| TMV027/DIT321 registration VT13<br>Exam valid 6hp | TMV026/DIT321 registration before VT13<br>Exam valid 7.5 hp |
|---|---|
| CTH: $\geqslant$ 27: 3, $\geqslant$ 40: 4, $\geqslant$50: 5 | CTH: $\geqslant$ 33: 3, $\geqslant$ 43: 4, $\geqslant$53: 5 |
| GU: $\geqslant$27: G, $\geqslant$45: VG | GU: $\geqslant$33: G, $\geqslant$50: VG |

No help material but dictionaries to/from English or Swedish.

Write in English or Swedish, and as readable as possible (think that what we cannot read we cannot correct).

*OBS:* 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.

*Good luck!*

---

1. (5pts) Prove that the words generated by the following grammar have always one more triangle (either up or down) than squares (either black or white):

   $$S \to \blacktriangledown \mid \blacktriangle \mid S \blacksquare S \mid \square S\, S$$

   Do not forget to clearly state which kind of induction you are using, the property you will prove, the base case(s) and the inductive hypothesis(es)!

2. (3.5pts) Construct a DFA which recognises the language generated by the regular expression $0(1+0)^*1 + 0(11+00)^*1$, without going via an $\epsilon$-NFA.

3. Consider the following $\epsilon$-NFA:

   | | 0 | 1 | $\epsilon$ |
   |---|---|---|---|
   | $\to q_0$ | $\{q_0\}$ | $\emptyset$ | $\{q_1\}$ |
   | $q_1$ | $\{q_2\}$ | $\emptyset$ | $\{q_3\}$ |
   | $q_2$ | $\emptyset$ | $\{q_1\}$ | $\emptyset$ |
   | $^*q_3$ | $\{q_3\}$ | $\emptyset$ | $\emptyset$ |

   (a) (1.5 pts) Use your intuition to give a regular expression generating exactly the same language as the one accepted by the automaton.

   (b) (4.5pts) Convert the $\epsilon$-NFA into a DFA.

4. (5pts) Minimise the following automaton. Show the intermediate table and justify the construction of the new automaton.

   | | $a$ | $b$ |
   |---|---|---|
   | $\to q_0$ | $q_1$ | $q_2$ |
   | $q_1$ | $q_3$ | $q_4$ |
   | $q_2$ | $q_5$ | $q_6$ |
   | $q_3$ | $q_3$ | $q_4$ |
   | $q_4$ | $q_5$ | $q_6$ |
   | $^*q_5$ | $q_4$ | $q_3$ |
   | $q_6$ | $q_5$ | $q_6$ |

5. (a) (3pts) Show that $(aa^*b^*b)^* = \epsilon + a(a+b)^*b$.

   (b) (3pts) Explain why the languages $(0+1)^*01(0+1)^* + 1^*0^*$ and $(0+1)^*$ are the same.

   Hint: A possible way to go is to analyse what each language represents rather than to show double inclusion.

6. (a) (1pts) When is a language regular? Explain as much as you can.

   (b) (4.5pts) For each of the following languages, give a regular expression which generates the language or prove that the language is not regular.

   i. $\{0^i1^j2^i \mid i, j \geqslant 0\}$;

   ii. $\{0^i1^j2^k \mid i, j, k \geqslant 0\}$.

7. (a) (6pts) Give a context-free grammar that generates the language $\{a^ib^jc^k \mid j \neq i+k \text{ with } i+j+k > 0\}$.

   (b) (1pt) When is a grammar ambiguous?

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

8. Consider the following grammar with start symbol $S$:

$$\begin{array}{ll} S \to aAB \mid AbB \mid F & \\ A \to aA \mid C & C \to cC \mid \epsilon \\ B \to bB \mid D & D \to dD \mid \epsilon \\ F \to fF \mid Ff & E \to eE \mid \epsilon \end{array}$$

   (a) (2pts) Identify the nullable variables and eliminate $\epsilon$-productions;

   (b) (2pts) Identify and eliminate unit productions in the grammar from (a);

   (c) (1.5pts) Identify and eliminate useless symbols in the grammar from (b);

   (d) (2.5pts) Use your intuition and describe, as formal as you can, the language generated by the grammar in (c);

   (e) (1pts) Is the language described in (d) regular? Justify;

   (f) (1.5pts) Put the grammar from (c) in Chomsky Normal Form.

9. (4pts) Consider the following grammar with start symbol $S$:

$$S \to AB \qquad A \to BB \mid a \qquad B \to AB \mid b$$

   Apply the CYK algorithm to determine if the string *aabba* is generated by this grammar. Show the resulting table and justify your answer.

10. (a) For TMV026/DIT321 registration before VT2013

    (6pts) Construct a Turing machine for the language $\{0^i1^j \mid i > j\}$ by giving its transition function. Explain it.

    NOTE: You may chose to do part (b) instead if you prefer.

    (b) For TMV027/DIT321 registration VT2013

    i. (4pts) Give a high-level description of a Turing machine for the language $\{0^i1^j \mid i > j\}$.

    ii. (1pt) Explain what a Turing decider is.

    iii. (1pt) State whether your Turing machine is also a Turing decider or not. Justify.

# Solutions Exam 130528

Here we only give a brief explanation of the solution. Your solution should in general be more elaborated than these ones.

1. The property to prove is $P(w)$ : If $S \Rightarrow^* w$ then $\#_T(w) - 1 = \#_S(w)$ where $\#_T$ and $\#_S$ are the functions that give the nr. of triangles and of squares, respectively, in a word.

   We will use course of value induction on the length of the derivation (number of steps) $S \Rightarrow^* w$.

   Base case: $S \Rightarrow w$ in one step, hence the rules applied should have been $S \to \blacktriangle$ or $S \to \blacktriangledown$.

   Then either $w = \blacktriangle$ or $w = \blacktriangledown$ respectively.

   In both cases $\#_T(w) = 1$ and $\#_S(w) = 0$. Hence the property holds.

   Step case: IH: if $S \Rightarrow^* w$ in at most $n > 0$ steps then $\#_T(w) - 1 = \#_S(w)$.

   Let $S \Rightarrow^* w'$ in $n + 1$ steps with $n > 0$.

   Since $n > 0$ then $w'$ is derived in at least 2 steps and then the first rule applied should have been $S \to S \blacksquare S$ or $S \to \square\, S\, S$.

   It should also be that either $w' = w_1 \blacksquare w_2$ or $w' = \square w_1 w_2$ with both $S \Rightarrow^* w_1$ and $S \Rightarrow^* w_2$. Observe that each of these derivations should take at least 1 step which means that each of them should be done in less than $n$ steps. Hence the IH applies to those 2 derivations and we have that $\#_T(w_1) - 1 = \#_S(w_1)$ and $\#_T(w_2) - 1 = \#_S(w_2)$.

   In both cases we have that $\#_S(w') = \#_S(w_1) + \#_S(w_2) + 1 =$ by IH $= \#_T(w_1) - 1 + \#_T(w_2) - 1 + 1 = \#_T(w_1) + \#_T(w_2) - 1 = \#_T(w') - 1$.

2. Since $0(11 + 00)^*1$ is included in $0(1 + 0)^*1$ you can simply have

| | 0 | 1 |
|---|---|---|
| $\to q_0$ | $q_1$ | $q$ |
| $q_1$ | $q_1$ | $q_2$ |
| $^*q_2$ | $q_1$ | $q_2$ |
| $q$ | $q$ | $q$ |

   Otherwise, you could start with a NFA and convert to a DFA.

NFA:

| | 0 | 1 |
|---|---|---|
| $\to q_0$ | $\{q_1\}$ | $\emptyset$ |
| $q_1$ | $\{q_1, q_3\}$ | $\{q_1, q_2, q_4\}$ |
| $^*q_2$ | $\emptyset$ | $\emptyset$ |
| $q_3$ | $\{q_1\}$ | $\emptyset$ |
| $q_4$ | $\emptyset$ | $\{q_1\}$ |

DFA:

| | 0 | 1 |
|---|---|---|
| $\to q_0$ | $q_1$ | $q$ |
| $q_1$ | $q_1q_3$ | $q_1q_2q_4$ |
| $q_1q_3$ | $q_1q_3$ | $q_1q_2q_4$ |
| $^*q_1q_2q_4$ | $q_1q_3$ | $q_1q_2q_4$ |
| $q$ | $q$ | $q$ |

3. (a) $0^*(01)^*0^*$

   (b)

| | 0 | 1 |
|---|---|---|
| $\to\ ^*q_0q_1q_3$ | $q_0q_1q_2q_3$ | $q$ |
| $^*q_0q_1q_2q_3$ | $q_0q_1q_2q_3$ | $q_1q_3$ |
| $^*q_1q_3$ | $q_2q_3$ | $q$ |
| $^*q_2q_3$ | $q_3$ | $q_1q_3$ |
| $^*q_3$ | $q_3$ | $q$ |
| $q$ | $q$ | $q$ |

4. First we run the algorithm that identifies equivalent states (you should explain a bit its construction):

| | $q_0$ | $q_1$ | $q_2$ | $q_3$ | $q_4$ | $q_5$ |
|---|---|---|---|---|---|---|
| $q_6$ | $X$ | $X$ | | $X$ | | $X$ |
| $q_5$ | $X$ | $X$ | $X$ | $X$ | $X$ | |
| $q_4$ | $X$ | $X$ | | $X$ | | |
| $q_3$ | | | $X$ | | | |
| $q_2$ | $X$ | $X$ | | | | |
| $q_1$ | | | | | | |

The resulting automaton is (you should explain a bit its construction):

| | $a$ | $b$ |
|---|---|---|
| $\rightarrow q_0q_1q_3$ | $q_0q_1q_3$ | $q_2q_4q_6$ |
| $q_2q_4q_6$ | $q_5$ | $q_2q_4q_6$ |
| $^*q_5$ | $q_2q_4q_6$ | $q_0q_1q_3$ |

5. (a) The left-to-right inclusion: If the word is empty then it is obvious.
Otherwise the word consists of several parts and in each part one or more $a$'s are followed by one of more $b$'s. Observe that a non empty word starts with $a$ and finished with $b$ and in the middle there are $a$'s and $b$'s in a certain pattern. This middle part can certainly be generated by $(a+b)^*$ since this generates all words with $a$'s and $b$'s.
The right-to-left-inclusion: If the word is empty then it is obvious.
Otherwise the word starts with $a$, it follows any number of $a$'s and $b$'s and finishes with a $b$. Note that this word consists of $n \geqslant 1$ blocks of $a$'s (each block with 1 or more $a$'s) followed by at least 1 $b$ (for example *aaabbaabab* consists of 3 blocks). Each block contains 1 or more $a$'s and 1 or more $b$'s. Each of these blocks is then generated by $(aa^*b^*b)$.
Alternative one could construct a DFA for each expression and see that they are equal.

(b) Let $\Sigma = \{0, 1\}$.
$(0+1)^*01(0+1)^*$ represents all string in $\Sigma^*$ with 01 as substring.
$1^*0^*$ represents the strings in $\Sigma^*$ where all the 1's precede all the 0's. Hence, no string where a 0 precedes a 1 is part of this language, that is, no string with 01 as substring is part of the language.
If we put both parts together we have the language over $\Sigma$ where either 01 is a substring or 01 is not a substring. Hence we get all words over $\Sigma$.
And this is exactly what $(0+1)^*$ represents.

6. (a) A language is regular if it can be accepted by a finite automaton (DFA, NFA, or epsilon-NFA) or generated by a regular expression or by a regular grammar.

(b) i. (3.5pts) Let $L = \{0^i1^j2^i \mid i, j \geqslant 0\}$. Let us assume $L$ is regular, then the Pumping lemma should apply. Let $n$ be the variable given by the lemma. Let $w = 1^n01^n \in L$. We have that $|w| \geqslant n$ so then we know that there are $x, y, z$ such that $w = xyz$, $|xy| \leqslant n$, $y \neq \epsilon$ and $\forall k.xy^kz \in L$.
Since $|xy| \leqslant n$ then $xy$ should contain only 1's (from the beginning of the word) and since $y \neq \epsilon$ then $y$ should contain at least one 1. Hence, when $k = 0$ then we are actually removing 1's from the beginning part of the word which now will contain less 1's than the ending part of the word.
Then $xy^0z \notin L$ and hence $L$ is not regular.
(Actually any $k \neq 1$ helps here, for $k > 1$ we add 1's at the beginning and break the balance also.)

ii. (1pt) The RE is $0^*1^*2^*$.

7. (a) If $j \neq i + k$ then either $j < i + k$ or $j > i + k$. $L$ will generate words with the left condition (less-than) and $M$ with the right condition (more-than).

$$\begin{array}{ll} S \to L \mid M & \\ L \to aAC \mid ACc & M \to DbE \\ A \to aAb \mid aA \mid a \mid ab \mid \epsilon & D \to aDb \mid Db \mid b \mid ab \mid \epsilon \\ C \to bCc \mid Cc \mid c \mid bc \mid \epsilon & E \to bEc \mid bE \mid b \mid bc \mid \epsilon \end{array}$$

$A$ generates words $a^i b^j, j \leqslant i$.
$C$ generates words $b^j c^k, j \leqslant k$.
$AC$ generates words $a^i b^j c^k, j \leqslant i + k$.
The productions of $L$ guarantee that at least there is an extra $a$ or an extra $c$ and hence $j < i+k$.
$D$ generates words $a^i b^j, j \geqslant i$.
$E$ generates words $b^j c^k, j \geqslant k$.
$DE$ would generate words $a^i b^j c^k, j \geqslant i + k$, by adding an extra $b$ we guarantee that $j > i + k$.

(b) A grammar is ambiguous when there exists a word $w$ for which there are at least 2 different parse trees or 2 different left/right-most derivations.

(c) Yes, *aabcc* have 2 different left-most derivations:
$S \Rightarrow L \Rightarrow aAC \Rightarrow aabC \Rightarrow aabCc \Rightarrow aabcc$ and $S \Rightarrow L \Rightarrow ACc \Rightarrow aACc \Rightarrow aabCc \Rightarrow aabcc$.

8. (a) Nullable variables: $C, D, E, A, B$.
After eliminating $\epsilon$-productions we get:

$$\begin{array}{ll} S \to aAB \mid a \mid aA \mid aB \mid AbB \mid b \mid Ab \mid bB \mid F & \\ A \to aA \mid a \mid C & C \to cC \mid c \\ B \to bB \mid b \mid D & D \to dD \mid d \\ F \to fF \mid Ff & E \to eE \mid e \end{array}$$

(b) Unit productions: $S \to F, A \to C, B \to D$.
After eliminating unit-productions we get:

$$\begin{array}{ll} S \to aAB \mid a \mid aA \mid aB \mid AbB \mid b \mid Ab \mid bB \mid fF \mid Ff & \\ A \to aA \mid a \mid cC \mid c & C \to cC \mid c \\ B \to bB \mid b \mid dD \mid d & D \to dD \mid d \\ F \to fF \mid Ff & E \to eE \mid e \end{array}$$

(c) First we need to eliminate the non-generating symbol $F$ and all productions involving $F$.
Then we need to eliminate the non-reachable symbol $E$ and all productions involving $E$.
After eliminating useless symbols we get:

$$\begin{array}{ll} S \to aAB \mid a \mid aA \mid aB \mid AbB \mid b \mid Ab \mid bB & \\ A \to aA \mid a \mid cC \mid c & C \to cC \mid c \\ B \to bB \mid b \mid dD \mid d & D \to dD \mid d \end{array}$$

(d) $C$ generates the language $c^+$, $D$ generates $d^+$.
$A$ generates $a^+c^* + a^*c^+$ and $B$ generates $b^+d^* + b^*d^+$.
$S$ generates then the language $a^+c^*b^*d^* + a^*c^*b^+d^*$.

(e) Yes, see the regular expression above.

(f) Put the grammar from (c) into Chomsky Normal Form.

$$\begin{array}{ll} S \to VU \mid a \mid VA \mid VB \mid UX \mid b \mid AX \mid BX & U \to AB \\ A \to VA \mid a \mid YC \mid c & V \to a \\ B \to XB \mid b \mid ZD \mid d & X \to b \\ C \to YC \mid c & Y \to c \\ D \to ZD \mid d & Z \to d \end{array}$$

9.

| | | | | |
|---|---|---|---|---|
| $\emptyset$ | | | | |
| $\{A\}$ | $\emptyset$ | | | |
| $\{S, B\}$ | $\{A\}$ | $\emptyset$ | | |
| $\emptyset$ | $\{S, B\}$ | $\{A\}$ | $\emptyset$ | |
| $\{A\}$ | $\{A\}$ | $\{B\}$ | $\{B\}$ | $\{A\}$ |
| $a$ | $a$ | $b$ | $b$ | $a$ |

Since $S$ does NOT belong to the upper set then the string is NOT generated by the grammar.

10. (a) You will need to give the transition function for the description below and explain it a bit.

(b) i. If in the initial state we read a blank or a 1 then the word is empty or starts with a 1, so we reject.
If the initial state reads a Y it means we have as many 0's as 1's and the word needs to be rejected (this could happen only after we have read at least one 0 and one 1).
If in the initial state we read a 0's, we replace it with an X, move to the left and change to the state that will look for the corresponding 1, if it exists.
The state that will look for the corresponding 1 might first need to move to the left across some 0's and then even to the left across some Y's.
If in the state that looks for the corresponding 1 we read a blank it means that there are more 0's than 1's so we need to accept.
If in the state that looks for the corresponding 1 we read a 1 then we mark it with an Y, we move to the right and we change to the state that will look for the first 0 that has not been changed to an X.
The state that looks for the first 0 that has not been changed to an X might first need to move to the right over some Y's and even to the right over some 0's until we get to an X. Then we move to the left and change to the initial state.
Note: I assume the tape only contains 0's, 1's and blank symbols.

ii. A Turing machine $M$ is a Turing decider if for all input words $w$ from our alphabet, all runs of $M$ over $w$ are halting, that is, they all either accept or reject the word.

iii. Yes since it always terminates with an accept or reject answer.