Name: Student number:

Exam 2, Module 7, Study Unit 202001361 Languages & Machines

Friday, April 1, 2021, 9:00 - 11:00

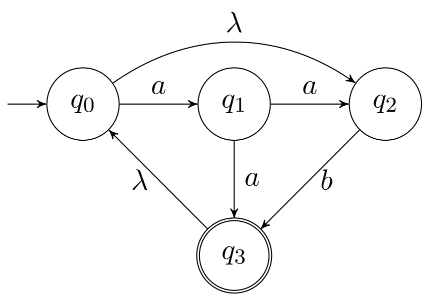
This second exam of Module 7 consists of the **Languages & Machines part** only, and is a **2h exam**.

You are allowed to use a handwritten cheat sheet (A4, double sided) during the exam.

After you finished your exam, take pictures of all pages and upload them as **one** PDF to the canvas assignment for this exam.
If you intend to leave before 11:00, then quietly come with your exam to the front and take the pictures there. Otherwise, please wait for our announcement at the end of the exam.

Exam sheet contains sample solution!

1. (8 points) Consider the following NFA-λ, M (only q_3 is accepting):



(a) Does the automaton accept the word $aabaa$?
If yes, provide a sequence of states with which the automaton can accept the word.
If no, provide a sequence of states with which the automaton rejects the word.

[ ] No. A potential rejecting state sequence is:

[✔] Yes. The sequence of states is: $q_0, q_1, q_2, q_3, q_0, q_1, q_3$

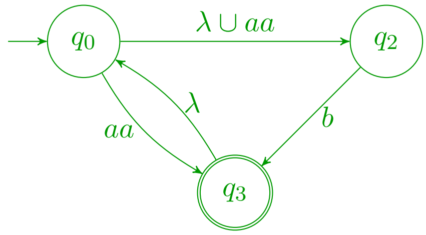
Does the automaton accept the word aaa?
If yes, provide a sequence of states with which the automaton can accept the word.
If no, provide a sequence of states with which the automaton rejects the word.

[✔] No. A potential state rejecting sequence is: q_0, q_1, q_3, q_0, q_1

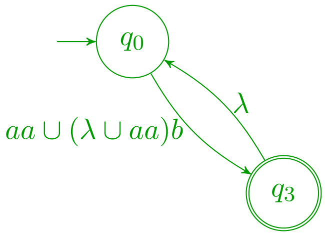
[ ] Yes. The sequence of states is:

(b) Transform the automaton M step by step to a regular expression.
States must be eliminated in the order q_1, q_2.

After elimination of q_1:



After elimination of q_2



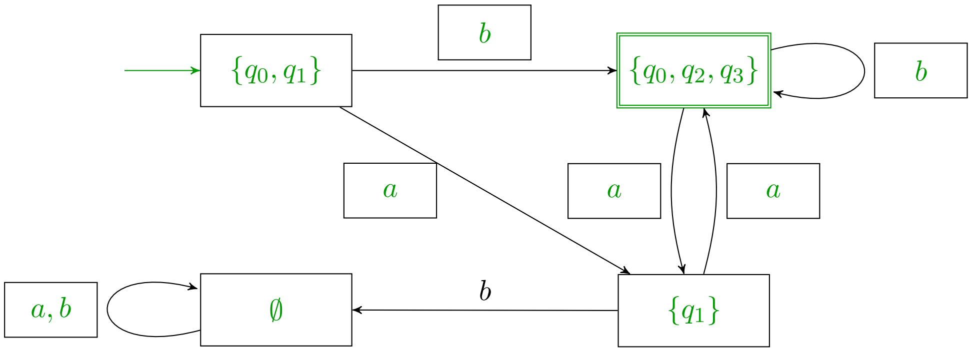
Thus, the regular expression obtained is:

$(aa \cup (\lambda \cup aa)b)^+$

(c) Fill in the following table with the λ-closure and input-transition function of M.

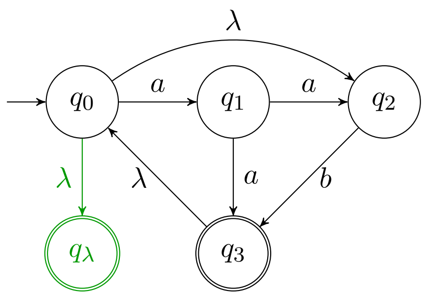
	λ-closure	a	b
q_0	$\{q_0, q_2\}$	$\{q_1\}$	$\{q_0, q_2, q_3\}$
q_1	$\{q_1\}$	$\{q_0, q_2, q_3\}$	$\emptyset$
q_2	$\{q_2\}$	$\emptyset$	$\{q_0, q_2, q_3\}$
q_3	$\{q_0, q_2, q_3\}$	$\{q_1\}$	$\{q_0, q_2, q_3\}$

(d) The following incomplete DFA accepts the same language as M.



i. Mark the initial state.
ii. Mark the accepting state/the accepting states.
iii. Label the states with the set of λ-NFA states they represent.
iv. Complete the white boxes at the transitions.

(e) Modify the λ-NFA in the figure below such that the empty word is also in its language (but no further new words).



2. (8 points) For each of the following languages, indicate whether it is

   - A: regular
   - B: context-free but not regular
   - C: recursive but not context-free
   - D: recursively enumerable but not recursive
   - E: not recursively enumerable

   For each correct answer, you will get 1 point. For each incorrect answer, you will loose 1 point. For no answer, you get 0 points. The total points of this exercise will not become negative.

   No further explanation is required.

   [B] Language $L_1 := \{a^{i+2}\,c\,b^{i+1} \mid 0 \leq i\}$
   [D] Language $L_2 := \{w \mid w$ encodes a Turing machine which writes the sequence $abc\}$
   [A] Language $L_3 := \{b^j\,c^i\,b^j\,(cb)^i \mid 0 \leq i \leq 23$ and $0 \leq j \leq 3i\}$
   [A] Language $L_4 := \mathcal{L}(a^*b^*)$
   [E] Language $L_5 := \{w \mid w$ encodes a Turing machine which never writes the sequence $abc\}$
   [C] Language $L_6 := \{a^i b^j a^{2i} b^{3j} \mid i \geq 0, j \geq 0\}$
   [C] Language $L_7 := \{a^i b^j c^k \mid 0 \leq i \leq j \leq k\}$
   [A] Language $L_8 := \{a^i b^j c^k \mid i$ even, j odd$\}$

3. (5 points) Consider the following languages

   - $L_1 = \mathcal{L}(a^*b^*)$
   - $L_2 = \mathcal{L}(c^*d^*)$
   - $L_3 = \{w \in \{a,b\}^* \mid |w| = 2^n \text{ for some } n \geq 0\}$

   Which of the following languages are regular? Mark languages for which this is the case with Y and mark those for which this does not hold with N. Leave out the answer if you do not know. Correct answers yield 1 point, for incorrect answers you loose 1 point. For no answer, you get 0 points. In total, you cannot get less than 0 points for this question.

   [Y] The language of words of L_1 which do not end with ab

   [N] $L_1 \cap L_3$

   [Y] $L_2 \cap L_3$

   [Y] $L_1 \cup L_2$

   [N] $L_1 \cup L_3$

4. (10 points) Consider the following context-free grammar G:

$$G = \begin{cases} S \to A\,B\,C \mid \lambda \mid A\,B \\ A \to a\,A\,a \mid a \\ B \to a\,b\,C \\ C \to C\,c \mid a \end{cases}$$

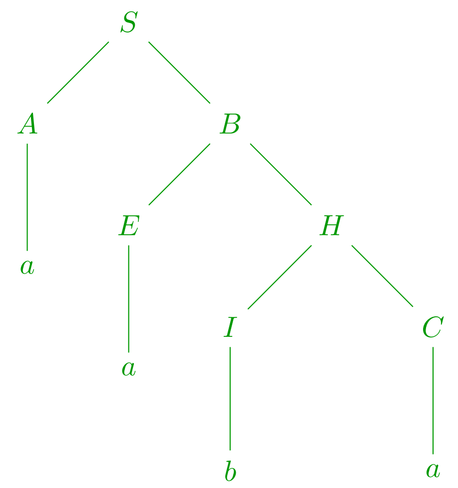
(a) Complete following grammar G' in Chomsky Normal form such that it is equivalent to G.

$$G' = \begin{cases} S \to \boxed{A\,D} \mid \lambda \mid A\,B \\ D \to \boxed{B\,C} \\ A \to \boxed{E\,F} \mid a \\ E \to a \\ F \to \boxed{A\,E} \\ \boxed{B} \to E\,H \\ H \to \boxed{I\,C} \\ I \to \boxed{b} \\ C \to \boxed{C\,K} \mid a \\ K \to c \end{cases}$$

(b) Let $w = aaba$. Use the table below to apply the CYK-algorithm (after Cocke-Younger-Kasami) to decide whether $w \in \mathcal{L}(G')$.

	1	2	3	4
1	$\{A, E, C\}$	$\{F\}$	$\emptyset$	$\{S\}$
2		$\{A, E, C\}$	$\emptyset$	$\{B\}$
3			$\{I\}$	$\{H\}$
4				$\{A, E, C\}$

Use the results of the CYK-algorithm to provide a derivation tree of G' for w:

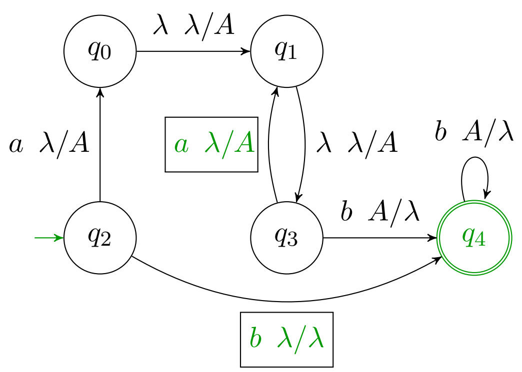


5. (8 points) Consider the context-free language $L = \{a^i\, b^{2i+1} \mid i \geq 0\}$.

(a) Give a context-free grammar G in Greibach Normal form for L.

$$G = \begin{cases} S & \to & b \mid a\,B\,B\,B \mid a\,A\,B\,B\,B\,B\,B \\ A & \to & a\,A\,B\,B \mid a \\ B & \to & b \end{cases}$$

(b) Consider the following incomplete non-extended, deterministic PDA.



Complete the automaton such that it accepts L. Note that we require this to be a deterministic, non-extended PDA. Only the symbol A may be pushed on the stack.

i. Mark the initial state

ii. Mark the accepting state or states.

iii. Fill in the two white boxes.

6. (5 points) Prove or disprove: the language $L := \{a^i b^i c^i \mid i \geq 0, i \text{ is prime}\}$ is context-free.

L is *not* context-free. We can prove this in almost the same way as for the language

$$X := \{a^i b^i c^i \mid i \geq 0\}$$

seen in the lecture. The reason that this works is basically that there are infinitely many prime numbers.

- Let k be arbitrarily given
- Choose $z = a^{k'} b^{k'} c^{k'}$ where k' is the smallest prime number larger or equal to k
- Let $uvwxy = z$ such that $|vx| > 0$ and $|vwx| \leq k$
- We distinguish the following possible cases:
  (a) v and x both contain one (repeated) symbol from $\{a, b, c\}$
    - Then $v \in \{a^m, b^m, c^m\}$ and $x \in \{a^n, b^n, c^n\}$ with $m + n = |vx| > 0$
    - Let $\{\alpha, \beta, \gamma\} \subseteq \{a, b, c\}$ such that $v = \alpha^m$ and $x = \beta^n$ and $\gamma \notin \{\alpha, \beta\}$.
    - Then uv^2wx^2y contains the same number of γ's as z does, but more α's and/or β's
    - Choose $i = 2$; then $uv^iwx^iy \notin L$

  (b) v or x contains (at least) 2 different symbols from $\{a, b, c\}$
    - Then v^2 or x^2 contains $a\,b$ or c before an a or a c before $a\,b$;
    - Choose $i = 2$; then $uv^2wx^2y \notin L$

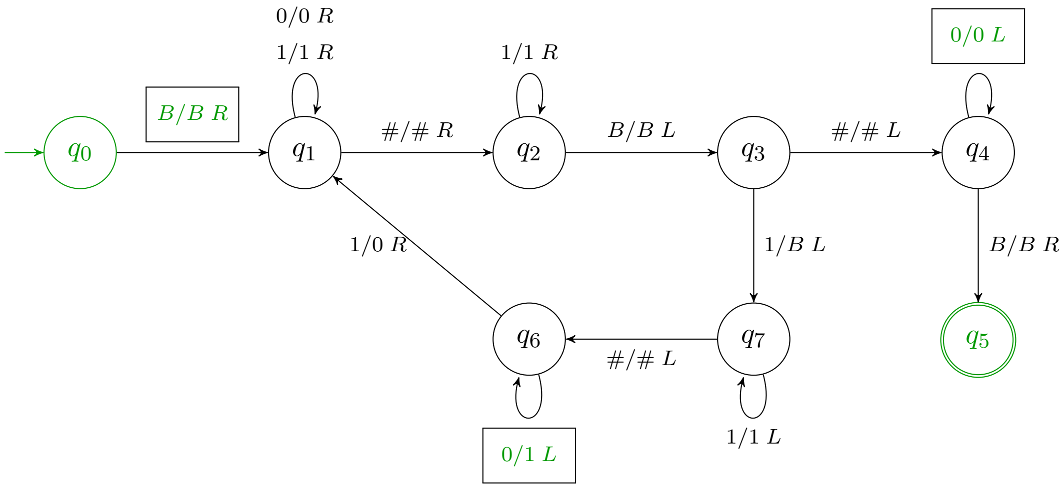
7. (6 points) Consider the language L over the alphabet $\Sigma = \{0, 1, \#\}$ with

$$L := \{w\#1^i \mid w \in \{0,1\}^* \text{ and } \mathrm{val}(w) = i\}$$

where $\mathrm{val}(w)$ is the value of w interpreted as a binary number. Thus, for instance $\mathrm{val}(1101) = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 13$. We have for instance

- $101\#11111 \in L$
- $111\#11 \notin L$
- $101\#11111\# \notin L$
- $1101\#1111111111111 \in L$
- $000\# \in L$
- $\# \in L$

Consider the following incomplete Turing machine to decide this language.



(a) Mark the state which should be initial.

(b) Mark the state or the states which should be accepting.

(c) Fill in the white boxes in the automaton figure.

(d) Decide what the following states do. Choose from the list of provided options

- state q_0: iv
- state q_1: vi
- state q_6: i
- state q_7: iii

Possible options:

i. We want to decrement w by 1. As long as we see 0s, we replace them by 1s and go to the left. As soon as we see a 1, we replace it by a 0 and go to the next round.

ii. All 1 on the right side of the $\#$ have been removed. Thus, check whether there are only 0s on the left side of the $\#$ remaining.

iii. We have just removed the rightmost 1. Thus, we search for the $\#$ symbol in order to decrement the w on its right side.

iv. Read blank character the machine head initially points at to to start reading the word.

v. Search for the right end of the sequence of 1 at the right side of $\#$.

vi. Read over the binary encoding part w of the input and search for the $\#$.

vii. The structure of the word is fine, we have left a blank left of the word and accept.

viii. Remove rightmost 1 and move on. If all 1 at right side of $\#$ have been removed, prepare checking that there are only 0 on the left side of the $\#$.

Notes:

Notes: