Name: Student number:

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

Thursday, April 21, 2022, 13:45 - 15:45

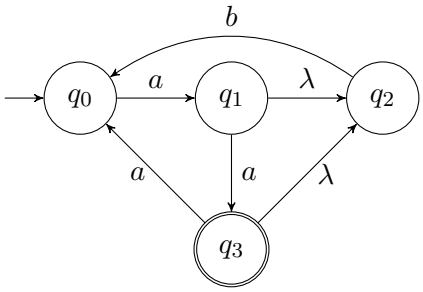
Carefully read the following instructions:

- 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. No other utilities (calculator, cell phone, books, etc.) are allowed, except writing utilities.
- Use an indelible pen with blue or black colour to write the exam. In particular, do not use a pencil. Pens which write with red or green colour are strictly forbidden.
- There are 5 pages of scrap paper attached to this exam. You are allowed to detach these pages from the exam sheet. In this case, we will not grade anything written on the loose pages. If you require more scrap paper, please inform us.
- Do not detach pages from the exam other than scrap paper.
- Write your name and student number on the top of each page of the exam.

Marks:

Exercise	1	2	3	4	5	6	7	$\sum$
Maximum	8	8	5	10	8	5	6	50
Achieved								

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



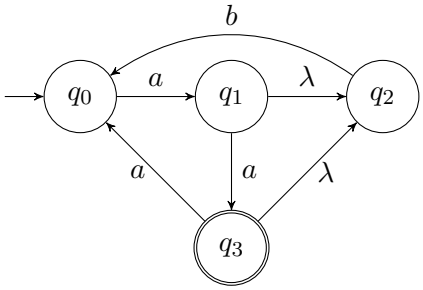
   (a) For each of the following words w, decide wether the automaton accepts it, by ticking (✔) the according box.
   If yes, also provide a sequence of states with which the automaton can accept the word.
   If no, also provide a sequence of states with which the automaton rejects the word.
   If the automaton rejects because there is no transition to a successor state with the according input, add "deadend" to the end of the sequence.

   i. $w := abaa$: [ ] No. [ ] Yes. Sequence of states:

   ii. $w := aab$: [ ] No. [ ] Yes. Sequence of states:

   iii. $w := aabaa$: [ ] No. [ ] Yes. Sequence of states:

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

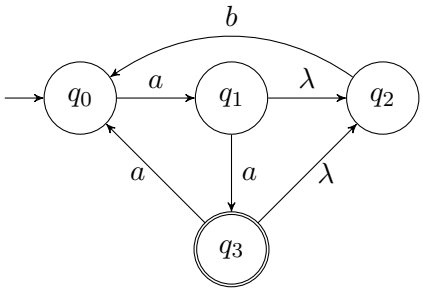


After elimination of q_1:

After elimination of q_2

Thus, the regular expression obtained is:

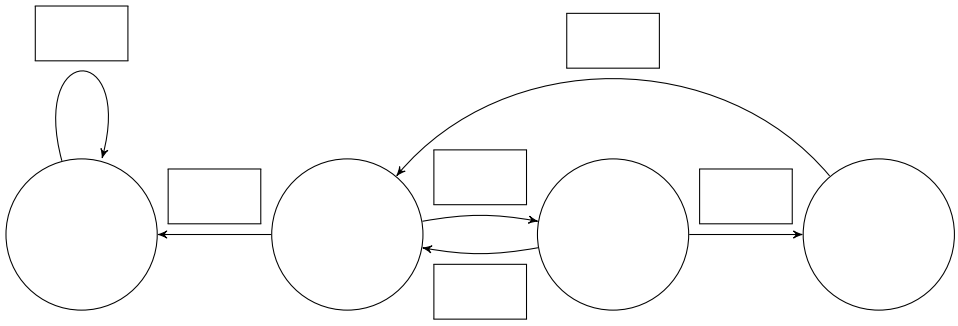
(c) Consider the following NFA-λ, M (only q_3 is accepting):



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

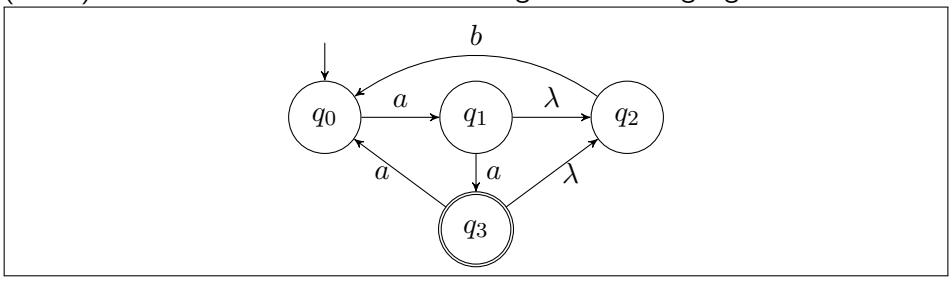
	λ-closure	a	b
q_0			
q_1			
q_2			
q_3			

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



i. Mark the initial state (by adding an incoming arrow)
ii. Mark the accepting state/the accepting states (with double lines)
iii. Label the states with the set of NFA-λ states they represent.
iv. Complete the white boxes at the transitions.

(e) Modify the NFA-λ in the figure below such that the word "aaa" is in its language. The only modification modification allowed is to reverse the direction of a single transition (arrow). We do not care about other changes to the language of the automaton.



2. (8 points)

(a) 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.

No further explanation is required.

[ ] Language $L_1 := \{w\#n \mid n$ encodes a number and w encodes a Turing machine which halts within n steps when started with empty word as input$\}$

[ ] Language $L_2 := \{b^j\, c^{3i}\, b^j\, (cb)^i \mid i \geq 62$ and $0 \leq j \leq 30\}$

[ ] Language $L_3 := \{w \mid w$ encodes a nondeterministic Turing machine which writes sequence $baab$ for input $w^R\}$

(b) For each of the following statements, indicate whether it is

- T: true
- F: false

For each correct answer, you will get 1 point. For each incorrect answer, you will loose 2 points. For no answer, you get 0 points.

No further explanation is required.

[ ] If we transform an NFA to a regular grammar with the method from the lecture, the resulting grammar will be unambiguous.

[ ] If we transform a DFA to a regular grammar with the method from the lecture, the resulting grammar will be unambiguous.

[ ] Languages for which only ambiguous grammars exist can only be decided by nondeterministic Turing machines but not by deterministic Turing machines.

[ ] It is possible to construct a Turing machine which decides whether a (binary) encoding of a Python programme is syntactically correct and that it is shorter than 5000 characters.

[ ] Let $G := \langle V, \Sigma, P, S\rangle$ be a context-free grammar with $|\Sigma| = 1$ such that for each rule of P there is at most one variable on the right side. The language of G is regular.

The total points of this exercise will not become negative.

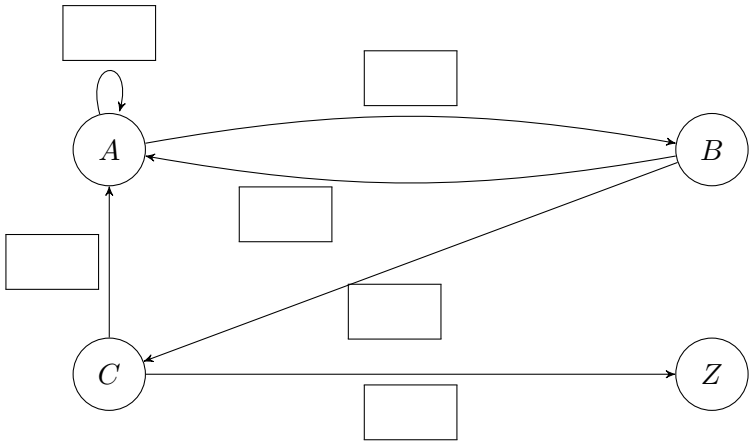
3. (5 points)

(a) Consider the following regular grammar $G_1 := \langle \{A, B, C\}, \{a, b\}, P_1, A \rangle$ with

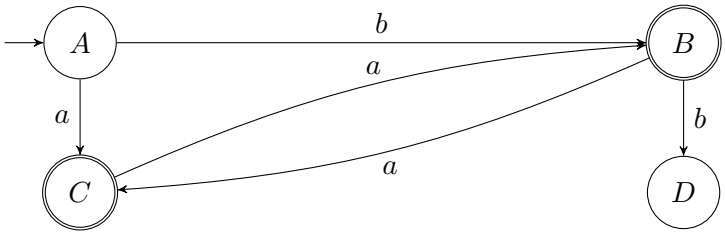
$$P_1 := \begin{cases} A & \to & a\,A \mid a\,B \mid b\,A \\ B & \to & a\,A \mid a\,C \\ C & \to & b\,A \mid \lambda \mid a \end{cases}$$

Complete the NFA N_1 below such that it accepts the same language as G.

i. Mark the initial state (by adding an incoming arrow)
ii. Mark the accepting state/the accepting states (with double lines)
iii. Complete the white boxes at the transitions.



(b) Consider the following NFA N_2:
(Correction: C accepting rather than D)



Complete the regular grammar G_2 such that it accepts the same language as N_2

i. Write the initial variable of the grammar
ii. Fill out the boxes in the rules

$G_2 := \langle \{A, B, C, D\}, \{a, b\}, P_2, \square \rangle$ with

$$P_2 = \begin{cases} \square & \to & b\,B \mid a\,C \\ B & \to & \square \mid \square \mid \square \\ \square & \to & \square \mid \square \end{cases}$$

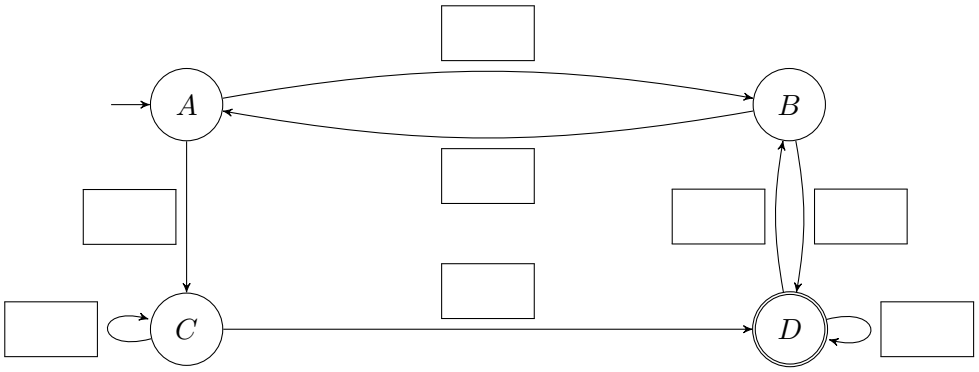
(c) Let a λ-regular grammar be such that its rules have the forms allowed for a regular grammar or (for $A, B \in V$): $A \to B$.

Consider the λ-regular grammar $G_3 := \langle \{A, B, C, D\}, \{a, b\}, P_3, A \rangle$ with

$$P_3 := \begin{cases} A & \to & b\,C \mid C \mid a\,B \\ B & \to & A \mid a\,D \\ C & \to & a\,D \mid b\,D \mid D \mid a\,C \\ D & \to & b\,D \mid \lambda \mid b\,B \end{cases}$$

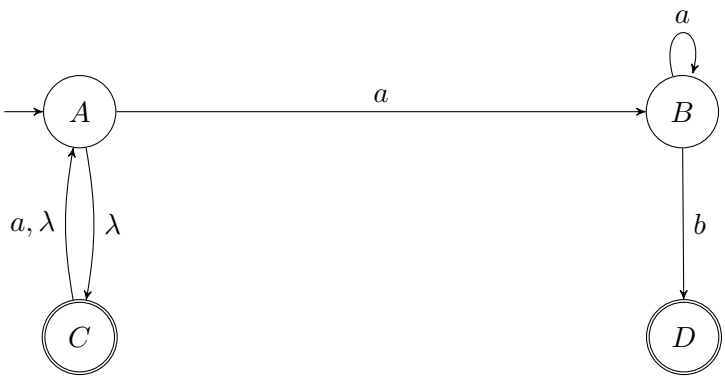
Correction: Added $D \to b\,B$

Complete the following NFA-λ N_3 such that it accepts the same language as G_3:



(d) Let a λ-regular grammar be such that its rules have the forms allowed for a regular grammar or (for $A, B \in V$): $A \to B$.

Consider the following NFA-λ N_4:



Complete the λ-regular grammar G_4 such that it accepts the same language as N_4

i. Fill out the boxes in the rules

$G_4 := \langle \{A, B, C, D\}, \{a, b\}, P_4, A \rangle$ with

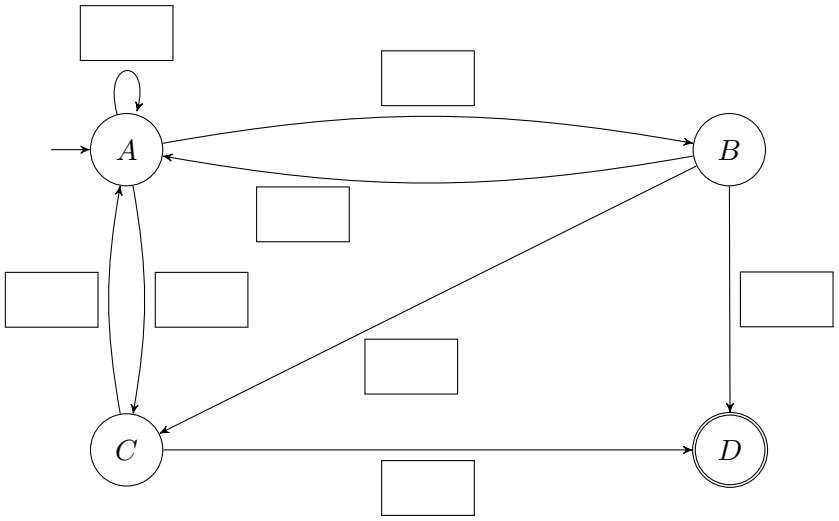
$$P_4 := \begin{cases} A & \to & \square \mid \square \\ B & \to & \square \mid \square \\ C & \to & \square \mid \square \mid \square \\ D & \to & \square \end{cases}$$

(e) Let a sequence-regular grammar be such that its rules have the forms allowed for a regular grammar or for $A, B \in V, w \in \Sigma^*$: $A \to w\,B$.

Consider the sequence-regular grammar $G_5 := \langle \{A, B, C, D\}, \{a, b\}, P_5, A \rangle$ with

$$P_5 := \begin{cases} A & \to & a\,a\,b\,A \mid a\,B \mid a\,b\,C \\ B & \to & a\,b\,A \mid a\,a\,a\,C \mid D \\ C & \to & b\,A \mid a\,a\,b\,b\,D \\ D & \to & \lambda \end{cases}$$

Complete the following NFA-λ N_5 such that it accepts the same language as G_5:



4. (10 points) Consider the context-free grammar $G := \langle \{S, T, U, V\}, \{a, b, c, d\}, P, S \rangle$ with

$$P := \begin{cases} S & \to & d\,T \mid a \\ T & \to & a\,T\,a \mid b\,T\,b \mid U \\ U & \to & c\,U \mid V \\ V & \to & d\,V \mid d \end{cases}$$

(a) Complete rules of the grammar $G' := \langle \{S, T, U, V, W, X, A, B, C, D\}, \{a, b, c, d\}, P', S \rangle$ in Chomsky Normal Form (CNF) such that it is equivalent to G. It is only required that G' is in CNF, we do not require that you provide exactly what the transformation algorithm would result in.

$$P' := \begin{cases} S & \to & D\,T \mid \square \\ T & \to & \square \mid B\,X \mid \square \mid \square \mid \square \\ U & \to & C\,U \mid D\,V \mid \square \\ \square & \to & D\,V \mid d \\ W & \to & T\,A \\ \square & \to & T\,B \\ A & \to & a \\ B & \to & b \\ C & \to & c \\ D & \to & d \end{cases}$$

(b) Let $w := dada$. Use the table below to apply the CYK-algorithm (after Cocke-Younger-Kasami) to decide whether $w \in \mathcal{L}(G')$. In case of empty sets, explicitly write "$\emptyset$".

	1	2	3	4
1			$\emptyset$	
2			$\emptyset$	
3				
4				

(c) Complete the following left-derivation of the grammar G' for the word $w := dada$.

S

$\vdash DT$

$\vdash$ $\square$

$\vdash$ $\square$

$\vdash$ $\square$

$\vdash$ $\square$

$\vdash dadA$

$\vdash dada$

Complete the following right-derivation of the grammar G' for the word $w := dada$.

S

$\vdash DT$

$\vdash$ $\square$

$\vdash$ $\square$

$\vdash$ $\square$

$\vdash$ $\square$

$\vdash Dada$

$\vdash dada$

5. (8 points) Consider the languages (over $\Sigma = \{a, b, \#\}$)

$$L := \{a^i \# b^j \# w \mid w \in \{a, b\}^*, i, j \geq 0, \mathrm{num}_a(w) = i, \mathrm{num}_b(w) = j\}.$$

and

$$L' := \{a^i \# b^j \# b^j a^i \mid i, j \geq 0\}$$

where $\mathrm{num}_a(w)$ ($\mathrm{num}_b(w)$) denotes the number of as (bs) in w.

(a) Complete the rules of the following general grammar $G := \langle \{S, T, A\}, \{a, b, \#\}, P, S \rangle$ such that $\mathcal{G} = L$.

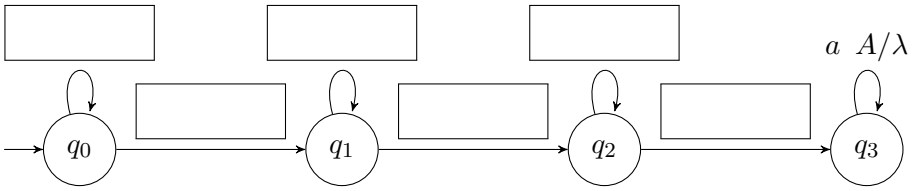
$$P := \left\{ \begin{array}{lcl} S & \to & \#\,T \\ T & \to & A\,T\,a \mid b\,T\,b \mid \# \\ A\,b & \to & b\,A \\ \square & \to & \square \\ \square & \to & \square \end{array} \right.$$

(b) Provide a regular expression R such that

$$\mathcal{L}(R) \cap L = L'.$$

$R :=$ ☐

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



Complete the automaton such that it accepts L' by

i. ~~marking the initial state (by adding an incoming arrow),~~
ii. marking the final state or states (with double lines), and
iii. by filling the six white boxes.

Note that this is not the automaton which would be obtain from the method from the lecture slides. We require this to be a deterministic, non-extended PDA. Only the stack symbols A and B may be pushed on the stack.

6. (5 points) Prove or disprove that the following language is context-free:

$$L := \{a^i b^i w \mid i \geq 0, w \in \{c, a\}^*, \mathrm{num}_c(w) = i\}$$

where $\mathrm{num}_c(w)$ denotes the number of cs in w.

Do not use the pumping lemma directly.

You are allowed to refer to languages from the lecture of which we already know whether or not they are context-free.

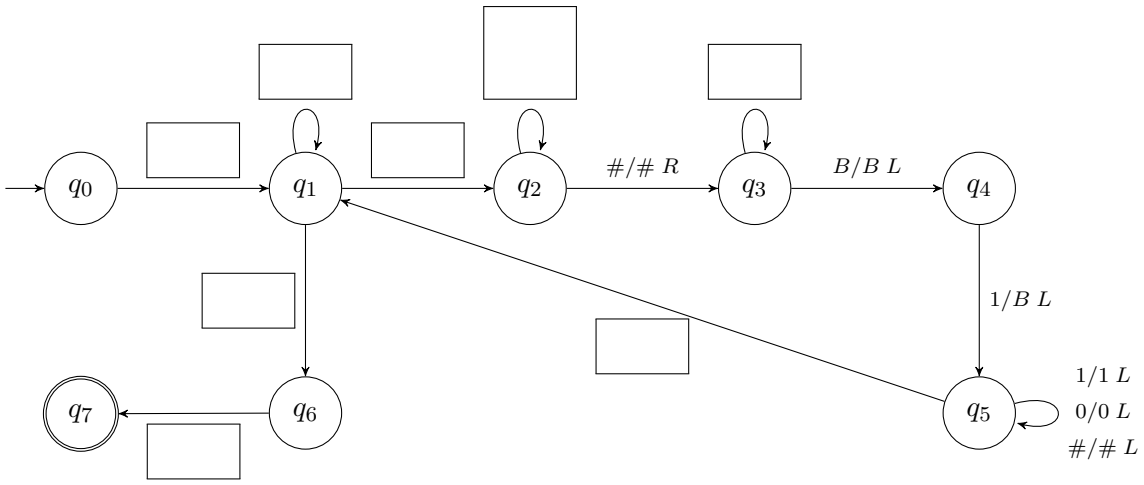
7. (6 points) Consider the language L over the alphabet $\Sigma = \{0, 1, \#\}$ with

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

where $\mathrm{val}^R(w)$ is the value of w^R interpreted as a binary number. Thus, for instance $\mathrm{val}^R(1110) = 1 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 + 0 \cdot 2^3 = 7$. We have for instance

- $101\#11111 \in L$
- $101\#11111\# \notin L$
- $000\# \in L$
- $111\#11 \notin L$
- $1011\#1111111111111 \in L$
- $\# \in L$

Consider the following incomplete Turing machine to decide this language.



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

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

- state q_1: ☐
- state q_3: ☐
- state q_4: ☐
- state q_6: ☐

Possible options:

i We have just removed the rightmost 1 of the sequence of 1 right of the $\#$. We now move to the left until we find the blank left to the word.

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

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

iv The structure of the word is fine, we accept.

v We want to decrement w by 1. As long as we see 0s, we replace them by 1s and go to the right. As soon as we see a 1, we replace it by a 0 and go to the next phase. If we see $\#$, we prepare to finish the computation.

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

vii We have just found the $\#$ while still decreasing w. We are almost done and just have to ensure that there are no 1 right to the $\#$.

viii We have found the blank at the right end of the sequence of 1 at the right side of $\#$. Thus, we remove it and move one step to the left to start over again.

Notes:

Notes:

Notes:

Name: Student number:

Notes:

Name: Student number:

Notes: