NATIONAL UNIVERSITY OF SINGAPORE

CS 3231 – Theory of Computation

Semester 1; AY 2021/2022; Final Examination

Time Allowed: 120 Minutes

INSTRUCTIONS TO CANDIDATES

1. Please write your Student Number. Do not write your name.
2. This assessment paper consists of TEN (10) questions and comprises ELEVEN (11) printed pages.
3. Students are required to answer **ALL** questions.
4. Students should either answer the questions in the space provided on the pdf of the question paper or hand-write on A4 paper, with each question on an extra page having question number and student number on top and questions being ordered before scanning.
5. This is an **CLOSED BOOK** assessment **WITH HELPSHEET**.
6. You are not permitted to communicate with other people except inviligators during the exam and you are not allowed to post in forums or other such entries during the exam and you are not allowed to read the internet, the Luminus or other sources except the question paper.
7. Every question is worth SIX (6) marks. The maximum possible marks are 60.
8. The file should be submitted to LUMINUS FILES and should have the name STUDENTNUMBER-FE-CS3231.pdf, say A01234567X-FE-CS3231.pdf

STUDENT NO: ______________________________

Let $\Sigma = \{0, 1, 2, 3\}$ (so the base 4 alphabet for writing natural numbers). Construct a regular expression of the language of all numbers which start and end with the same digit from Σ. The numbers should not have leading zeroes, that is, they should be written in the shortest usual way. So 00110 and 0110 cannot be used but 110 can be used.

Solution. A valid expression is

$$\{0,1,2,3\} \cup (\{1\} \cdot \{0,1,2,3\}^* \cdot \{1\}) \cup (\{2\} \cdot \{0,1,2,3\}^* \cdot \{2\}) \cup (\{3\} \cdot \{0,1,2,3\}^* \cdot \{3\}).$$

The first part of the union is the list of all one-digit numbers and the further three parts have, for $a = 1, 2, 3$, respectively, the numbers of at least two digits which start and end with a. Numbers starting with 0 do not exist, except for 0 itself, as these are explicitly excluded in the question text.

Consider the language $L = \{v2w : v, w \in \{0,1\}^+ \text{ and } v \neq w\}$. Determine the first case of the below which applies to make a grammar $(\{0,1,2\}, N, P, S)$ for L where N is the set of non-terminals and P the set of rules and S the start symbol. Consider the following types of grammars (used for L or other languages not containing the empty word):

1. Regular: Every rule in P is of the form $A \to v$ or $A \to wB$ where $v, w \in \Sigma^*$ and $A, B \in N$;
2. Linear: Every rule in P is of the form $A \to vBw$ or $A \to v$ with $v, w \in \Sigma^*$ and $A, B \in N$;
3. Context-free: Every rule in P is of the form $A \to w$ where $A \in N$ and $w \in (N \cup \Sigma)^*$;
4. Context-sensitive: Every rule in P is of the form $l \to r$ where l contains at least one non-terminal and $|l| \leq |r|$.

Construct the grammar according to the first of the four options which is possible and explain why none of the options before is possible.

Solution. The best possible choice is linear. The language L is the union of three languages I, J and K, where $I = \{t1v2u0w : t, u, v, w \in \{0,1\}^* \text{ and } |u| = |t|\}$, $J = \{t0v2u1w : t, u, v, w \in \{0,1\}^* \text{ and } |u| = |t|\}$ and $K = \{v2w : v, w \in \{0,1\}^+ \text{ and } |v| \neq |w|\}$.

The non-terminals are $\{S, T, U, V, W, X, Y, Z\}$, the terminals are $\{0,1,2\}$, the start symbol is S and the rules consist of the rules $S \to X|Z$ which transforms the symbol S into the start symbols for the below subgrammars of I, J, K plus the rules of all three subgrammars.

Now the rules of the subgrammar for I are as follows: $Z \to Z0|Z1|T0$, $T \to 0T0|0T1|1T0|1T1|12|1U2$, $U \to U0|U1|0|1$.

Now the rules of the subgrammar for J are as follows: $Z \to Z0|Z1|V1$, $V \to 0V0|0V1|1T0|1V1|02|0W2$, $U \to W0|W1|0|1$.

Now the rules of the subgrammar for K are as follows: $X \to 0X0|0X1|1X0|1X1|0Y20|0Y21|1Y20|1Y21|02Y0|02Y1|12Y0|12Y1$, $Y \to Y0|Y1|0|1$.

It can be seen that $\{0\}^* \cdot \{2\} \cdot \{0\}^* - L = \{0^n 2 0^n : n \geq 0\}$ which is a famous example for a set which is not regular, thus also L cannot be regular, as the set differences of regular sets is regular.

Question 3 [6 marks]

Consider the following regular expression: $L = \{0,1,2\}^* \cdot \{0\}^+ \cdot \{1\}^+ \cdot \{2\}^+ \cdot \{0,1,2\}^*$. Describe the set L in words and provide a deterministic finite automaton. This deterministic finite automaton should have as few states as possible.

Solution. The set L consists of all words which have a subword containing all of $0, 1, 2$ such that the zeroes are first followed by the ones followed by the twos. Before and after the subword can be arbitrary digits.

The dfa has the states s, z, o, t, where the state t is accepting and the others are rejecting. The transition table is as follows:

$$\begin{tikzcd}
\text{start} \arrow[r] & s \arrow[loop above, "1,2"] \arrow[r, "0"] & z \arrow[loop above, "0"] \arrow[r, "1"] \arrow[l, bend left, "2"] & o \arrow[loop above, "1"] \arrow[r, "2"] \arrow[l, bend left, "0"] & t \arrow[loop above, "0,1,2"]
\end{tikzcd}$$

Here s is the start state. When the most recent input is a sequence of zeroes and t was not yet reached then the state is z, when it is a sequence of zeroes first and ones second and t was not yet reached then the state is o, when it is a sequence of zeroes followed by a sequence of ones followed by a sequence of twos then the state is t and there the dfa stays forever. In particular the dfa accepts a word iff it has a subword of the form 01^+2. Before the 0 and after the 2 there can be arbitrarily many symbols and the 1 appears in this subword at least once and also can appear in any larger quantity.

To see that the four states are needed, note that the three rejecting states cannot be equal to the accepting state. Furthermore, from o one can reach the accepting word with the word 2, but not from s, z. From z one can reach the accepting word with the word 12, but not from s. Thus the behaviour of the three rejecting states is different on the pair on the set $\{12, 2\}$ of words. Therefore the three rejecting states are not equivalent. Furthermore, all states are reacheable from the start state and all allow to go into an accepting state, thus one cannot omit any state.

Consider the language L of all words ua where a is a symbol not appearing in u and let Σ be the alphabet and n be the size of Σ, $n \geq 2$. Consider the following possibilities of the size of a description in dependence of the alphabet size n: (a) linear size, (b) quadratic size but not linear size; (c) polynomial size but neither linear nor quadratic size; (d) exponential size. For the following three items say which of the options (a), (b), (c) and (d) applies:

1. Size of regular expression built from finite sets, union, concatenation and Kleene star where the size of a regular expression is the length of it when written down as a string;

2. Size of deterministic finite automaton in terms of number of states in dependence of n;

3. Size of non-deterministic finite automaton in terms of number of states in dependence of n.

For each of the three tasks, give a short explanation for the reason for the corresponding choice from (a), (b), (c) and (d).

Solution. For 1., the option is (b). The regular expression is quadratic in the number of states. One has, for each digit a, an expression σ_a containing a set of all digits in Σ different from a and then forms the expression $(\sigma_a^* \cdot \{a\})$ for all a. These expressions are connected by unions. This construction has quadratic size. So for $\Sigma = \{0, 1, 2\}$ the expression would be $(\{0, 1\}^* \cdot \{2\}) \cup (\{0, 2\}^* \cdot \{1\}) \cup (\{1, 2\}^* \cdot \{0\})$. Note that each σ_a is already of linear length and as there are not much compacter ways to write down the expression, the overall complexity which is the union of n linear expressions is then of quadratic and thus of polynomial length.

For 2., the option is (d). There are exponentially many states needed. The dfa must remember all the symbols it has seen so far. In the case that it reads in some state the symbol a, it must go to an accepting state if a has not yet seen before and to a rejecting state otherwise. Thus it must store the whole set of different symbols seen before and for each subset of Σ there must be at least one state.

For 3., the option is (a). On the first symbol a, the nfa goes from the start state s into a state q_b with $b \neq a$ where $a, b \in \Sigma$ or directly into the accepting state for one-symbol words. From then onwards, it stays in q_b until the symbol b is read upon which it goes into one common accepting state t. In the accepting state, the nfa cannot go on and the symbol b or the symbol a at direct transfer into the accepting state has to be the last symbol. Thus there are $n + 2$ states and the quantity of states is linear in n.

Let h be a homomorphism with $h(0) = 3$, $h(1) = 33$ and $h(2) = 3333$. Furthermore, let $H = \{33\} \cdot (\{333\}^* \cup \{33333\}^*)$ and let $L = \{x \in \{0, 1, 2\}^* \, [h(x) \in H]\}$. Is there an nfa with up to 10 states recognising the language L? If so, then construct this nfa else explain why the nfa does not exist.

Here a homomorphism h is a mapping which satisfies $h(v \cdot w) = h(v) \cdot h(w)$ for all words v, w where $\cdot$ denotes the concatenation. It is sufficient to specify h on all letters of the alphabet of the domain on which h is defined. An nfa is a non-deterministic finite automaton and it has one start state, perhaps multiple accepting (or final) states and transition function which maps pairs (state,terminal symbol) to sets of possible new states. An nfa accepts a word iff there is a run on the word starting in the start state and ending in an accepting state.

Solution. The nfa exists; a dfa of that size would not exist. The idea is to go from the start state non-deterministically directly into one of two loops of length 5 and 3, respectively, which count modulo 5 and 3, respectively. So there will be states $s, q_0, q_1, q_2, q_3, q_4, r_0, r_1, r_2$. Let $f(0) = 1, f(1) = 2, f(2) = 4$. Now the transition function δ is as follows:

$$\begin{aligned} \delta(s, a) &= \{q_{f(a) \; mod \; 5}, r_{f(a) \; mod \; 3}\} \\ \delta(q_b, a) &= \{q_{(b+f(a)) \; mod \; 5}\} \\ \delta(r_b, a) &= \{r_{(b+f(a)) \; mod \; 3}\} \end{aligned}$$

So the only non-deterministic choice is the first move. Here b ranges over the possible indices of the states q_b and r_b, a ranges over the set $\{0, 1, 2\}$. The state s is the start state and rejecting, the states q_b and r_b are accepting iff $b = 2$. The so constructed nfa has nine states, one below the required bound.

Ogden's Pumping Lemma can be thought of a game between two players, Anke and Boris. Anke wants to prove that the language L behaves like a context-free language, that is, satisfies the pumping lemma; Boris wants to disprove this. The game goes as follows:

1. Anke selects a pumping constant k for L;
2. Boris selects a long enough word $u \in L$ and colours at least k symbols in the word in red (so called marked symbols), if L does not contain a word of length k or more, then Anke has won and the game ends here;
3. Anke splits the word into five parts v, w, x, y, z with $u = vwxyz$ such that at least one of the letters in the word wy is red and at most k of the letters in the word wxy are red;
4. Boris selects a number $h \in \mathbb{N}$;
5. If $vw^hxy^hz \in L$ then Anke wins else Boris wins.

Consider $L = \{u \in \{0,1\}^* :$ For $n \in \mathbb{N}$, if $n^2 \leq |u| \leq n^2 + 2n$ then there are at most n zeroes in $u\}$. Does L satisfy Ogden's Lemma? That is, determine which player (Anke saying YES or Boris saying NO) has a winning strategy in the above algorithm. Describe the winning strategy.

Solution. The language L does not satisfy Ogden's Lemma; in other words, player Boris has a winning strategy. After Anke selected the constant k, Boris produces the word $0^k1^{(k-1)k}$ and Boris colours all ones in red. Then Anke selects the two pumps v and y. v and y together contain $i \in \{0, 1, 2, \ldots, k\}$ many zeroes and $j \in \{1, 2, \ldots, k\}$ ones. If $i > 0$ then Boris pumps up and selects $h = 2$; it follows that the word is of the form $0^{k+i}1^{k(k-1)+j}$ and $i + j \leq 2k$. Thus the word has at most length $k^2 + 2k$ and for being in L, it can contain at most k zeroes; however, as $i > 0$, the word is not in L. If $i = 0$ then Boris pumps down and selects $h = 0$, so the word gets by j symbols shorter. As $j \geq 1$, the resulting word can only contain up to $k - 1$ zeroes, but by pumping down none of the k zeroes were removed, so the word is again not in L. Thus Boris wins in both cases the game and has a winning strategy. It follows that L does not satisfy Ogden's Lemma and in particular, L is not context-free.

Consider the following grammar: $(\{S,T\},\{0,1,2,3\},P,S)$ where the set P contains the rules $S \to 0S|1ST|2STT|3$ and $T \to 3$. In which normal form is the grammar: (a) Chomsky Normal Form; (b) Greibach Normal Form; (c) Linear Grammar Normal Form?

Furthermore, can the language be recognised by a deterministic push-down automaton which accepts by empty stack and which reads exactly one terminal symbol in every cycle and which has exactly one stack symbol U? Please answer with YES or NO and give some reasons for the answer.

For this, note that the deterministic push-down automaton asked for has in every situation at most one option it can apply. Furthermore, it accepts a word iff it does not get stuck during processing the input, the input is fully read, it is in an accepting state after processing the input and the stack is empty.

Solution. The grammar is in Greibach Normal Form: On every right side of the rule stands one terminal symbol followed by a perhaps empty word of non-terminal symbols.

The answer is YES. One can indeed find a deterministic pushdown automaton to recognise the language by empty stack. The stack symbol is just an U and the states are s and t. The rules are as follows: $(s,U,0) \to (s,U)$; $(s,U,1) \to (s,UU)$; $(s,U,2) \to (s,UUU)$; $(s,U,3) \to (t,\varepsilon)$; $(t,U,3) \to (t,\varepsilon)$. The grammar produces words from $\{0,1,2\}^* \cdot \{3\}^+$ and if the first part contains i 1s and j 2s then the second part contains $i+j+j+1$ 3s. The state s is rejecting and the state t is accepting. The grammar pushes for each $a \in \{0,1,2\}$ in addition to the top symbol U in the stack a further symbols U onto the stack. When it sees a 3, it transfers from the rejecting start state into an accepting state t and, furthermore, with every 3 read, the top symbol U of the stack will be consumed without replacement. The state t makes sure that only 3s are processed and the PDA has to reach state t in order to make the stack empty.

Consider the following function: If the ternary representation $a_1a_2\ldots a_n$ of x has as many 1 as 2 then $f(x) = 1$ else $f(x) = 0$.

Write a register program which computes $f(x)$. The register machine is allowed to add, to subtract and to compare registers and constants with $<, >, =, \neq, \leq, \geq$ and multiplication with constants can be expressed by repeated addition: For example, $R_2 = 3 \cdot R_2$ can be implemented by $R_2 = R_2 + R_2 + R_2$ and similarly for other constants. Assignments to registers by other registers or constants are also allowed. Goto commands can be used, but it is permitted to use If-Then-Else statements for better readability and to group several statements into one line or between "Begin" and "End" in If-Then-Else statements. If you use macros, please provide the full code of the macros.

Solution.

Line 1 Function Ternary(R1); $R2 = 1$; $R_3 = 0$; $R_4 = 0$;

Line 2 If $R_2 > R_1$ Then Goto 3; $R_2 = R_2 + R_2 + R_2$; Goto 2;

Line 3 $R_1 = R_1 + R_1 + R_1$;

Line 4 If $R_2 \leq R_1$ Then Begin $R_1 = R_1 - R_2$; $R_3 = R_3 + 1$ End;

Line 5 If $R_2 \leq R_1$ Then Begin $R_1 = R_1 - R_2$; $R_3 = R_3 - 1$; $R_4 = R_4 + 1$ End;

Line 6 If $R_1 > 0$ Then Goto 3;

Line 7 If $R_3 = R_4$ Then Return(1) Else Return(0).

Here R_1 is the input, R_2 is the smallest power of 3 larger than the input and R_3 and R_4 count the number of 1s and 2s, respectively, in the ternary representation of the input. The digits are read out at the top and subtracted off the number in this process, multiplying R_1 with three shifts the next ternary digit into the read position. When the number R_1 is 0, the algorithm can stop.

Recall that the set

$$\{(a,b,c) : \exists x,y\, [a,b,c,x,y \in \mathbb{N} \wedge a \cdot x^2 + b \cdot y = c]\}$$

is **NP**-complete. What about the following sets (where $a,b,c,x,y,z \in \mathbb{N}$):

1. $A = \{(a,b,c) : \exists x,y,z\, [a \cdot x^2 \cdot z + b \cdot y \cdot z = c \cdot z]\}$;
2. $B = \{(a,b,c) : \exists x,y\, [a \cdot x^2 \cdot (y+2) + b \cdot (y+1)^2 = c \cdot (y+2) + b]\}$;
3. $C = \{(a,b,c) : \exists x\, [a \cdot x^2 + b \cdot x = c]\}$.

Assuming that $\mathbf{P} \neq \mathbf{NP}$ and $\mathbf{NP} \neq \mathbf{PSPACE}$, determine the complexities of these sets in terms of a register machine by using the choices for the sets A,B,C: (a) Constant time; (b) Polynomial but not constant time; (c) **NP**-complete; (d) **PSPACE**-complete; (e) Primitive recursive but not in **PSPACE**. Give short explanations for the choices, but register machine programs are not required.

Solution. The set A has constant time complexity, so choice (a). The reason is that A contains all tuples (a,b,c), as the variable z is multiplied with all terms and z can be 0. So the register machine just returns 1 for accept independently on what natural numbers a,b,c are in the input.

The set B is **NP**-complete, so choice (c). As y is a natural number, $y+2$ is greater or equal 2. Thus when considering $a \cdot x^2 + b \cdot y = c$, one can multiply the equation with $(y+2)$ without changing the solvability. To this one adds b on both sides to get $a \cdot x^2 \cdot (y+2) + b \cdot y \cdot (y+2) + b = c \cdot (y+2) + b$. Then rearranging the bracketing gives the definition of B. Thus the set B is exactly the **NP**-complete set from the example in the question.

The set C is in **P** but not constant time, so choice (b). If one sets $x = 0, y = 0$ the the left side is 0 and thus less equal c. If at least one of a,b is non-zero, then letting $x = c$ gives the upper bound $a \cdot c^2 + b \cdot c$ for c. By halving search one can in polynomial determine an x such that $a \cdot x^2 + b \cdot x \leq c < a \cdot (x+1)^2 + b \cdot (x+1)$. Now one checks whether the first inequality is actually an equality in order to decide whether the equation by parameters (a,b,c) has a solution or not.

Consider the Diophantine set

$$A = \{x \in \mathbb{N} : \exists y_1, y_2, y_3 \in \mathbb{N}\, [p(x, y_1, y_2, y_3) = 0]\}$$

with $p(x, y_1, y_2, y_3) = (x - y_1 \cdot y_1 \cdot y_1 - 1 - y_2)^2 + (x + y_3 - y_1^3 - 3 \cdot y_1^2 - 3 \cdot y_1)^2$. Here, for terms t, t^2 is an abbreviation of $t \cdot t$ and t^3 is an abbreviation of $t \cdot t \cdot t$.

(a) Describe the numbers x in this set A in every day language so that it is clear what set it is. Such descriptions could be "The set of prime numbers", "The set of numbers which are not powers of 5", "The set of factorials" and so on. Explain the description.

(b) An equivalent definition of Diophantine sets is the following: A set B is Diophantine iff there is a polynomial q with integer coefficients and a number n of variables such that

$$B = \{q(y_1, \ldots, y_n) : y_1, \ldots, y_n \in \mathbb{N}\} \cap \mathbb{N}.$$

Find a polynomial with at most three variables which shows that the set A from above is Diophantine, that is, choose q such that A is the set of all natural numbers inside the range of q. Explain the formula.

Solution. For **(a)**, the set A contains all natural numbers which are not a cube, that is, which are not of the form $y_1 \cdot y_1 \cdot y_1$ for any natural number. The formula of A consists of two squared conditions which are 0 when satisfied and greater equal 1 otherwise. The first condition is that x is the sum of $y_1^3 + 1$ and some natural number y_2, the second condition is that the sum $x + y_3$ is equal to the natural number $y_1^3 + 3 \cdot y_1^2 + 3 \cdot y_1$ which is $(y_1 + 1)^3 - 1$. In other words, the condition says that there is an y_1 such that $y_1^3 < x < (y_1 + 1)^3$ and therefore x is not a cube number.

For **(b)**, one starts with the idea to use p as a black box. More precisely consider $r(x, y_1, y_2, y_3) = x \cdot (1 - 2 \cdot p(x, y_1, y_2, y_3)^2)$. Now if $p(x, y_1, y_2, y_3) = 0$ then $r(x, y_1, y_2, y_3) = x$. If $p(x, y_1, y_2, y_3) \neq 0$ then x will be multiplied with a negative number and thus the output of $r(x, y_1, y_2, y_3)$ is negative. This formula can be simplified to use less variables by the following polynomial q in place of r:

$$q(y_1, y_2, y_3) = (y_1^3 + 1 + y_2) \cdot (1 - 2 \cdot (y_2 + 1 + y_3 - 3 \cdot y_1 \cdot (y_1 + 1))^2).$$

The simplification is to use the expression $y_1^3 + 1 + y_2$ in place of x and so to eliminate the variable x. Then only the condition that $y_2 < 3y_1(y_1 + 1)$ must be satisfied, as $y_2 = x - 1 - y^3$. This condition is equivalent to $y_2 + 1 + y_3 = 3 \cdot y_1 \cdot (y_1 + 1)$.

Here some examples of natural numbers x in the range of q: For $x = 7$, choose $y_1 = 1$, $y_2 = 5$ and $y_3 = 0$; For $x = 12$, choose $y_1 = 2$, $y_2 = 3$ and $y_3 = 14$; For $x = 28$, choose $y_1 = 3$, $y_2 = 0$ and $y_3 = 35$.

END OF QUESTION PAPER.