NATIONAL UNIVERSITY OF SINGAPORE

CS 4232 – Theory of Computation

Semester 1; AY 2017/2018; Midterm Test 2

Time Allowed: 40 Minutes

INSTRUCTIONS TO CANDIDATES

1. Please write your Student Number. Do not write your name.
2. This assessment paper consists of FOUR (4) questions and comprises NINE (9) printed pages.
3. Students are required to answer **ALL** questions.
4. Students should answer the questions in the space provided.
5. This is a **CLOSED BOOK** assessment.
6. It is permitted to use calculators, provided that all memory and programs are erased prior to the assessment; no other material or devices are permitted.
7. Every question is worth FIVE (5) marks. The maximum possible marks are 20.

STUDENT NO: _____

This portion is for examiner's use only

Question	Marks	Remarks
Question 1:		
Question 2:		
Question 3:		
Question 4:		
Total:		

Question 1 [5 marks] **CS 4232 – Solutions**

For a homomorphism h from Σ^* to Γ^* and a language $L \subseteq \Gamma^*$, the inverse image of L under h is $h^{-1}(L) = \{u \in \Sigma^* : h(u) \in L\}$.

Let $\Gamma = \{0, 1, 2\}$ and $L = \{00\}^* \cup \{11\}^*$. Determine the maximum number n of states needed for an nfa of $h^{-1}(L)$ and provide an example homomorphism h where this number is taken – other homomorphisms might allow lower values for n, so the worst case n should be determined and the witnessing h provided.

Explain why the corresponding n is optimal.

Solution. The language L itself needs an nfa with five states; it has a start state and two branches from it which recognise $\{00\}^*$ and $\{11\}^*$; these two branches need to be kept different, as mixed words like 0011 are not in the language, therefore the nfa can also only leave but not come back to the start state. So the set of states is $Q = \{s, z, z', o, o'\}$ with s, z', o' being accepting and z, o rejecting. The nfa goes on 0 from s to z and on 1 from s to o. Furthermore, each 1 let the nfa swap between the states o, o' and each 0 let the nfa swap between the states z, z'. There are no other transitions.

Now given the nfa $(Q, \Gamma, \delta, \{s, z', o'\}, s)$, a new nfa for $h^{-1}(L)$ can take the same states Q and the same starting state s, but on a symbol $a \in \Sigma$ and a state $q \in Q$, the nfa can go to any state $p \in Q$ where the original nfa can go on the word $h(a)$ from q to p; the accepting states remain the same and the alphabet is Σ. This construction then allows to conclude that every $h^{-1}(L)$ has an nfa with 5 states. In the case that $\Sigma = \Gamma$ and $h(a) = a$ for all $a \in \Sigma$, this identity homomorhism satisfies $h^{-1}(L) = L$ and therefore needs as above an nfa with 5 states.

Thus the correct solution is $n = 5$.

Question 2 [5 marks] **CS 4232 – Solutions**

Consider the context-free grammar $(\{S, T, U\}, \{0, 1, 2, 3\}, P, S)$ with P containing the rules $S \to 0S1S2|TU|3$, $T \to STS$, $U \to U3|3$ and write an equivalent context-free grammar in Chomsky Normal Form without useless non-terminals; that is, every of its non-terminals can be reached and all non-terminals allow to derive some word.

Solution. As T cannot be terminalised and U can only be derived when a T is derived, the non-terminals T, U can be omitted from the grammar and also all rules which contain one of these non-terminals on either side. So the resulting rules are $S \to 0S1S2|3$.

Now the grammar is put into Chomsky Normal Form and besies the non-terminal S inherited from above, it also has non-terminals $U, V, W, 0', 1', 2'$. Here $0', 1', 2'$ stand for non-terminals for which there are only the derivations $0' \to 0$, $1' \to 1$ and $2' \to 2$ in the grammar. Furthermore, the other non-terminals S, U, V, W have the rules $S \to 0'U|3$, $U \to SV$, $V \to 1'W$, $W \to S2'$. The symbol S is the start symbol. The terminals remain unchanged as $0, 1, 2, 3$.

Question 3 [5 marks] **CS 4232 – Solutions**

Assume that L and H are context-free languages not containing the empty word ε. Is there a $O(n^3)$ algorithm to check the membership of a word of length n in the language $(L - H) \cdot (H - L)$, where $-$ denotes the set difference and $\cdot$ the concatenation?

☐ Yes, ☐ No.

If there is an algorithm, describe how the algorithm works and why it keeps the time-bound; if there is no such algorithm, explain why it cannot exist. Note that here the size of the two grammars is taken to be constant and absorbed in the $O(n^3)$-expression. So if one fixes the grammars, then the resulting algorithm should be in cubic time.

Solution. Assume that the grammars in Chomsky Normal Form are for L the grammar are (N_1, Σ, P_1, S_1) and for H the grammar (N_2, Σ, P_2, S_2), where N_1 and N_2 are disjoint. So one can now run the Cocke Kasami Younger algorithm with the non-terminals $N_1 \cup N_2$, as the union is also a grammar in Chomsky Normal Form with the only pathology that there are two start symbols. For input word $a_1 a_2 \ldots a_n$, one builds now in time $O(n^3)$ the pyramide-shaped table of all sets $U_{i,j}$ of non-terminals which can be derived into $a_i \ldots a_j$. Now the word is in $(L - H) \cdot (H - L)$ iff there is an $m \in \{1, 2, \ldots, n-1\}$ such that $U_{1,m}$ contains S_1 but not S_2 and $U_{m+1,n}$ contains S_2 but not S_1. This last test can be done with one loop over m and takes time $O(n^2)$, thus the overall running-time of the algorithm is $O(n^3)$.

Question 4 [5 marks] **CS 4232 – Solutions**

Is there a register machine which can compute the function F given as $n \mapsto 3^{2^n}$?

☐ Yes, ☐ No.

For the answer "yes", provide the corresponding program; for the answer "no", provide a proof why this program does not exist.

Solution. The answer is "yes". The program can be made as follows.

Line 1: Function $F(R_1)$;

Line 2: $R_2 = 0$; $R_3 = 1$;

Line 3: If $R_1 = R_2$ Then Goto Line 6;

Line 4: $R_2 = R_2 + 1$; $R_3 = R_3 + R_3$;

Line 5: Goto Line 3;

Line 6: $R_4 = 0$; $R_5 = 1$;

Line 7: If $R_4 = R_3$ Then Goto Line 10;

Line 8: $R_4 = R_4 + 1$; $R_5 = R_5 + R_5 + R_5$;

Line 9: Goto Line 7;

Line 10: Return(R_5).

END OF PAPER