Theoretical Computer Science Exam, February 7, 2023

The exam consists of **4 exercises**. Available time: 1 hr 30 min.; you may write your answers in Italian, if you wish.

All exercises must be addressed for the written exam to be considered sufficient.

F.NAME …….. L.NAME …… PERSON CODE ……

Exercise 1 (8 pts)

Consider the language

$$L = \{ a^m \; e^j \; b^n \mid n, m > 0 \;\wedge\; m < n \;\wedge\; j \geq 0 \}$$

For instance, strings *aeebbb, aabbb* $\in L$, while *aaeb*, *ebb* $\notin L$.

a) Design an automaton of the least powerful type that accepts language L. Write a computation of the automaton that accepts the string *aeebb* and one that rejects the string *aaebb.*

b) Write a grammar, of the least general type, that generates language L. For this grammar write a derivation of the string *aeebbb*.
c) Design an automaton of the least powerful type that accepts language L^+.
d) Write a grammar, of the least general type, that generates language L^+.

Exercise 2 (8 pts)

Specify, by means of a sentence of Monadic First-Order logic (or Second-Order only if necessary) the language L over the alphabet $\{a, b, c\}$ defined by the following expression

$$L = a\, b^* c$$

Explain why the following strings do not satisfy the specification sentence. Please refer as precisely as possible to the structure of the sentence.

- *abca*
- *bbc*
- *abb*
- *aa*

Consider now the following variant L_1 of language L

$$L_1 = a\, (bb)^* c$$

and say how the specification sentence for language L could be modified to specify L_1.

Exercise 3 (8 pts)

Please answer the following questions providing suitable explanations for your answers.

1) Consider the problem of determining if the language accepted by a (generic) Turing machine is empty.
   a) Is this problem decidable?

   b) Is it semidecidable?
2) Consider next the problem of determining if, given two generic Turing machines M_1 and M_2, there is a string accepted by both of them.
   a) Is this problem decidable?

   b) Is it semidecidable?

Exercise 4 (8 pts)

Consider the language $L = \{ w\, w^R \mid w \in \{ a, b \}^+ \}$.

1. Describe a ***deterministic*** k-tape Turing machine M that recognizes language L. You do not need to provide a precise description of M in terms of its transition function, but you should provide an informal description sufficiently clear and detailed so that the time complexity $T_M(n)$ (where $n = |x|$ is the length of the input string) can be estimated. Efficient solutions are preferred and the time complexity should be $\Theta(n)$. Provide an estimation of the time complexity function as precise as possible; notice that you should evaluate the complexity *function*, not only its Θ class.

2. Say if a ***nondeterministic*** k-tape Turing machine N can recognize language L with a time complexity $T_N(n)$ such that $T_N(n) < T_M(n)$ for all n. In the positive case, describe N in a sufficiently clear and detailed way so that the time complexity $T_N(n)$ (where $n = |x|$ is the length of the input string) can be estimated, and estimate $T_N(n)$ as precisely as possible. In the negative case, explain why such a nondeterministic Turing machine does not exist.

Solutions

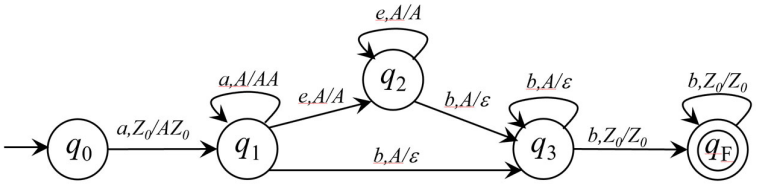
Exercise 1

Consider the language

$$L = \{ a^m\ e^j\ b^n \mid n, m > 0 \ \wedge \ m < n \ \wedge \ j \geq 0 \}$$

For instance, strings *aeebbb, aabbb* $\in L$, while *aaeb*, *ebb* $\notin L$.

a) Design an automaton of the least powerful type that accepts language L. Write a computation of the automaton that accepts the string *aeebb* and one that rejects the string *aaebb.*



$\langle q_0\ aeebb\ Z_0\rangle \vdash \langle q_1\ eebb\ AZ_0\rangle \vdash \langle q_2\ ebb\ AZ_0\rangle \vdash \langle q_2\ bb\ AZ_0\rangle \vdash \langle q_3\ b\ Z_0\rangle \vdash \langle q_F\ \varepsilon\ Z_0\rangle$ **accept**

$\langle q_0\ aaebb\ Z_0\rangle \vdash \langle q_1\ aebb\ AZ_0\rangle \vdash \langle q_1\ ebb\ AAZ_0\rangle \vdash \langle q_2\ bb\ AAZ_0\rangle \vdash \langle q_3\ b\ AZ_0\rangle \vdash \langle q_3\ \varepsilon\ Z_0\rangle$ **reject**

b) Write a grammar, of the least general type, that generates language L. For this grammar write a derivation of the string *aeebbb*.

Grammar: (axiom S)

$S \rightarrow S\ b \ \mid A\ b$

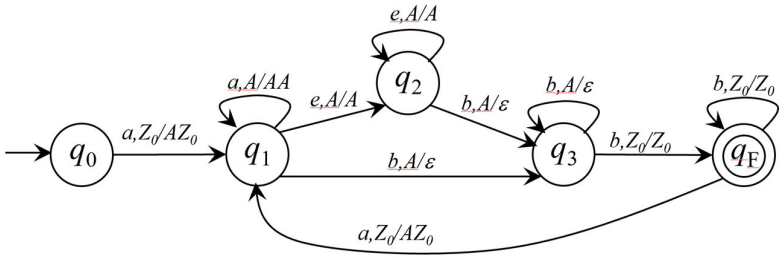
$A \rightarrow a\ A\ b \ \mid a\ E\ b$

$E \rightarrow e\ E \ \mid \varepsilon$

Derivation:

$S \Rightarrow S\ b \Rightarrow A\ b\ b \Rightarrow a\ E\ b\ b\ b \Rightarrow a\ e\ E\ b\ b\ b \Rightarrow a\ e\ e\ E\ b\ b\ b \Rightarrow a\ e\ e\ b\ b\ b$

c) Design an automaton of the least powerful type that accepts language L^+.



d) Write a grammar, of the least general type, that generates language L^+.

Grammar: (axiom X)

$X \to S\,X \mid S$

$S \to S\,b \mid A\,b$

$A \to a\,A\,b \mid a\,E\,b$

$E \to e\,E \mid \varepsilon$

Exercise 2

Specify, by means of a sentence of Monadic First-Order (or Second-Order only if necessary) the language L over the alphabet $\{a, b, c\}$ defined by the following expression

$$L = a\,b^*\,c$$

$a(0) \wedge$

$\forall x \big(\; (\, a(x) \to b(x{+}1) \vee c(x{+}1) \,) \wedge$

$\quad (\, b(x) \to b(x{+}1) \vee c(x{+}1) \,) \wedge$

$\quad (\, c(x) \to last(x) \,)$

$\big)$

An alternative specification formula: $a(0) \wedge \exists y \, (\, last(y) \wedge c(y) \wedge \forall x \, (\, 0 < x < y \to \; b(x) \,) \quad)$

Explain why the following strings do not satisfy the specification sentence.

- *abca* falsifies $\forall x \ldots (\, c(x) \to last(x) \,)$
- *bbc* falsifies $a(0)$
- *abb* falsifies $\forall x \ldots (\, b(x) \to b(x{+}1) \vee c(x{+}1) \,)$ at position 2
- *aa* falsifies $\forall x \ldots (\, a(x) \to b(x{+}1) \vee c(x{+}1) \,)$

Consider now the following variant L_1 of language L

$$L_1 = a\,(bb)^*\,c$$

and say how the specification sentence for language L could be modified to specify L_1.

The strings of L_1 have the same sequence of letters as those of L but the number of b's is even, and so is the length of any string. Therefore it suffices to add, as a conjunct, the typical sub-formula (of monadic second order logic) specifying that the string length is even.

$\exists X \, \forall x \, (\, (\, x{=}0 \to X(x) \,) \wedge (\, last(x) \to \neg X(x) \,) \wedge \forall y \, (\, y{=}x{+}1 \to (\, X(x) \leftrightarrow \neg X(y) \,) \,) \,)$

Exercise 3

1) Consider the problem of determining if the language accepted by a (generic) Turing machine is empty.
   a) Is this problem decidable?

The problem is not decidable, because the set of functions computed by a Turing machine accepting the empty language satisfies the hypotheses of the Rice theorem.

   b) Is it semidecidable?

The problem is not even semidecidable, because the complement problem is semidecidable: given a generic Turing machine, it is possible di determine, by means of a dovetail simulation, if it accepts some string.

2) Consider next the problem of determining if, given two generic Turing machines M_1 and M_2, there is a string accepted by both of them.
   a) Is this problem decidable?

It is not decidable. Consider in fact a simplified version of this problem where machine M_2 is fixed to any machine accepting exactly one string. Then this new problem has just one machine parameter, and consists of determining if a generic Turing machine accepts a given input string. Now the Rice theorem applies directly to this problem (because the set of functions computed by these machines is not trivial, i.e., it is not empty nor it is equal to the set of all computable functions), which is therefore undecidable. Therefore the original problem, which is a generalization of this new simplificed problem, is not decidable either.

   b) Is it semidecidable?

It is semidecidable, by means of a (dovetail) simulation in which, for all possible input strings, the two machines are executed, on the same string, for an increasing number of steps. If there exists a string accepted by both machines, soon or late their executions both terminate successfully.

Exercise 4

Consider the language $L = \{ w\, w^R \mid w \in \{ a, b \}^+ \}$.

1. Describe a ***deterministic*** k-tape Turing machine M that recognizes language L. You do not need to provide a precise description of M in terms of its transition function, but you should provide an informal description sufficiently clear and detailed so that the time complexity $T_M(n)$ (where $n = |x|$ is the length of the input string) can be estimated. More efficient solutions are preferred and the time complexity should be $\Theta(n)$. Provide an estimation of the time complexity function as precise as possible; notice that you should evaluate the complexity *function*, not only its Θ class.

A 2-tape deterministic Turing machine M reads the entire input string x and stores it in the memory tape T_1 , while it writes $n/2$ in unary on tape T_2 ($n=|x|$). This requires $n+c_1$ steps, where c_1 is a small constant.

Then M moves back the memory head of T_1 to the initial position. This requires $n+c_2$ steps, where c_2 is a small constant.

Then using the number $n/2$ stored in unary on T2, it compares the elements of the second half of the input string, read from the input from right to left, with the elements of the first half, read from T_1 from left to right. This requires $(n/2)+c_3$ steps, where c_3 is a small constant.

Overall the time complexity function is $2n + n/2 + c = 5 \cdot n/2 + c$, where c is a small constant.

2. Say if a ***nondeterministic*** k-tape Turing machine N can recognize language L with a time complexity $T_N(n)$ such that $T_N(n) < T_M(n)$ for all n. In the positive case, describe N in a sufficiently clear and detailed way so

that the time complexity $T_N(n)$ (where $n = |x|$ is the length of the input string) can be estimated, and estimate $T_N(n)$ as precisely as possible. In the negative case, explain why such a nondeterministic Turing machine does not exist.

The nondeterministic Turing machine N (with one memory tape) can nondeterministically guess the center of the string and work very much as a nondeterministic pushdown automaton that accepts the same language: it reads the first half of the input and stores it in the memory tape. Then it checks that the characters of the second part of the string, read from the input from left to right, are equal to characters in a symmetric position in the first part of the string, stored in the memory tape and read from right to left. Overall this requires $n + c$ steps, where c is a small constant.