Mock-Exam Questions

Question 1 carries 50 marks, questions 2 - 6 are shorter and carry 25 marks each. Answer a set of questions carrying a total of 100 marks (namely, either QUESTION 1 and TWO other questions or FOUR of questions 2 - 6).

Question 1. (*50 points*)
Part 1: Consider the language $\mathcal{L}$ on the alphabet $A = \{0, 1\}$:

$$\mathcal{L} = \{0^k 1 0^k | k \geq 0\}$$

(a) Give context-free grammars G that generates precisely the language $\mathcal{L}$ (you need only one non-terminal symbol S, which is also the start symbol).

7 points.

(b) Is there a *Finite State Automaton* accepting (precisely) language $\mathcal{L}$? Give a reason for your answer.

10 points.

(c) Give the definition of the set $\mathcal{E}$ of all *regular expressions* over the aphabet $\{0, 1\}$. Is $\mathcal{L}$ a regular language, i.e., can $\mathcal{L}$ be denoted by some expression in $\mathcal{E}$? Give a reason for your answer.

8 points.

Part 2: Consider the following Turing Machine $M = (A, S, \nu, \zeta, \delta)$:
- the alphabet A is $\{0, 1\} \cup \{\triangleright, \sqcup\}$, where the auxiliary symbol $\triangleright$ marks the beginning of the tape and $\sqcup$ is the blank symbol;
- the set of states is $\{s, s_0, s_0', s_1, q\} \cup \{\text{“}yes\text{”}, \text{“}no\text{”}\}$ where s is the initial symbol;
- the next state function ν, the output function ζ and the direction function δ are given in list notation as follows:

$$\begin{array}{llll}
(s\ 0\quad s_0\ \triangleright\ \rightarrow); & (s\ 1\quad s_1\ \triangleright\ \rightarrow); & (s\ \triangleright\quad s\ \triangleright\ \rightarrow); & (s\ \sqcup\quad no\ \sqcup\ -); \\
(s_0\ 0\quad s_0\ 0\ \rightarrow); & (s_0\ 1\quad s_0\ 1\ \rightarrow); & & (s_0\ \sqcup\quad s_0'\ \sqcup\ \leftarrow); \\
(s_0'\ 0\quad q\ \sqcup\ \leftarrow); & (s_0'\ 1\quad no\ 1\ -); & (s_0'\ \triangleright\quad no\ \triangleright\ -); & \\
(s_1\ 0\quad no\ 0\ -); & (s_1\ 1\quad no\ 1\ -); & & (s_1\ \sqcup\quad yes\ \sqcup\ -); \\
(q\ 0\quad q\ 0\ \leftarrow); & (q\ 1\quad q\ 1\ \leftarrow); & (q\ \triangleright\quad s\ \triangleright\ \rightarrow). &
\end{array}$$

The input to M always begins with the symbol $\triangleright$ followed by a string $x \in \{0, 1\}^*$ followed only by symbols $\sqcup$ (i.e., no blank symbols $\sqcup$ occur before the end of the input).

(d) Write the outcome (i.e., $M(x) =$ “yes” or “not”) and the *sequences of configurations* representing the computations of M (i) with input $\triangleright 010$ and (ii) with input $\triangleright 0100$.

10 points.

(e) Describe (in English) the behaviour of the machine M given in part (d).

8 points.

(f) What is the language $\mathcal{L} \subset \{0, 1\}^*$ accepted by M? Is the language $\mathcal{L}$ decidable?

7 points.

Answers:

(a) $\mathcal{L}$ is generated by the grammar $G = (\{S, 0, 1\}, \{0, 1\}, P, S)$ with start symbol S and the following set of production rules P:

$$\{S \to 0S0, \qquad S \to 1\}$$

(b) Let M be an automaton with n states accepting $\mathcal{L}$: consider the string $x1x$ where x is the string 0^n (n occurrences of 0). M accepts $x1x$, since $x1x$ is in $\mathcal{L}$. After reading all of x the machine has been twice in the same state reading the same input 0. Indeed, let s^k (for $0 \leq k \leq n$) be the state of M after reading a prefix of x of length k: since there are $n+1$ prefixes of 0^n (namely, ϵ, 0^1, 0^2, …, 0^n) and only n states in M, by the *pigeon-hole principle* we must have $s^i = s^j$ for some $0 \leq i < j \leq n$. It follows that M does also accept a string $y = 0^k10^n$ where $k = n-j+i < n$. But y is not in $\mathcal{L}$. (See Hopctoft, Motwani and Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 2000, p.126.)

(c) The set $\mathcal{E}$ is the set of all expressions built up from $\emptyset$ (denoting the empty language), ϵ (denoting the language containing only the empty string), **0** and **1** (denoting languages containing only a one-letter word) and closed under the operations of concatenation $L_1 \cdot L_2$, union $L_1 + L_2$ and Kleene's closure L_1^*.
Since a language is regular if and only if it is accepted by a Finite State Automata, by part (b) there is no expression in $\mathcal{E}$ denoting the language $\mathcal{L}$.

(d)

(i) $M(010) =$ "yes":

$$\begin{array}{rcl}
(\sqcup & s & \triangleright 010) \\
(\triangleright & s & 010) \\
(\triangleright\triangleright & s_0 & 10) \\
(\triangleright\triangleright 1 & s_0 & 0) \\
(\triangleright\triangleright 10 & s_0 & \sqcup) \\
(\triangleright\triangleright 1 & s_0' & 0\sqcup) \\
(\triangleright\triangleright & q & 1\sqcup\sqcup) \\
(\triangleright & q & \triangleright 1\sqcup\sqcup) \\
(\triangleright\triangleright & s & 1\sqcup\sqcup) \\
(\triangleright\triangleright\triangleright & s_1 & \sqcup\sqcup) \\
(\triangleright\triangleright\triangleright & yes & \sqcup\sqcup).
\end{array}$$

(ii) $M(010) =$ "no":

$$\begin{array}{rcl}
(\sqcup & s & \triangleright 0100) \\
(\triangleright & s & 0100) \\
(\triangleright\triangleright & s_0 & 100) \\
(\triangleright\triangleright 1 & s_0 & 00) \\
(\triangleright\triangleright 10 & s_0 & 0) \\
(\triangleright\triangleright 100 & s_0 & \sqcup) \\
(\triangleright\triangleright 10 & s_0' & 0\sqcup) \\
(\triangleright\triangleright 1 & q & 0\sqcup) \\
(\triangleright\triangleright & q & 10\sqcup) \\
(\triangleright & q & \triangleright 10\sqcup) \\
(\triangleright\triangleright & s & 10\sqcup) \\
(\triangleright\triangleright\triangleright & s_1 & 0\sqcup) \\
(\triangleright\triangleright\triangleright & no & 0\sqcup)
\end{array}$$

(e) Let x be the input of a computation of M. A *subroutine* in the computation of $M(x)$ is as follows: M starts in state s reading a symbol $\triangleright$, then moves to the right.
• If the first symbol after $\triangleright$ is the blank symbol $\sqcup$, then M enters state "no" and halts rejecting.
• If the input after $\triangleright$ is 1, M erases it, by replacing it with $\triangleright$, enters state s_1 and moves to the right. If M currently reads $\sqcup$, then M enters state "yes" and accepts. Otherwise, M enters state "no" and rejects.

• If the input after $\triangleright$ is 0, then M erases it, by replacing it with the symbol $\triangleright$ and entering state s_0. Next it moves to the extreme right of the tape, still in state s_0 until it finds a blank symbol $\sqcup$; then it moves left in state s_0' and reads the last non-blank symbol of its current input.

(i) if the current input is empty, so M reads the symbol $\triangleright$, then the machine enters state "no" and halts rejecting;

(ii) If this is 1 then M enters state "no" and halts;

(iii) otherwise, the symbol read is an occurrence of 0, so M replaces it with $\sqcup$ and moves left in state q until it reaches an occurrence of $\triangleright$, and starts a new subroutine.

(f) $\mathcal{L}$ is the language

$$\mathcal{L} = \{0^k 1 0^k | k \geq 0\}$$

of Part 1 and machine M decides $\mathcal{L}$. Indeed in any computation of $M(x)$ every subroutine either terminates in a state "yes" or "no" or yields a new subroutine with a shorter current input. Therefore every computation terminates and for every x in A^* we have $M(x) =$ "yes" or "no".

Question 2. (*25 points*) Let $N = (A, S, s_0, \{s_1\}, \Delta)$ be the Non-Deterministic Finite State Automaton with alphabet $A = \{0, 1\}$, set of states $S = \{s_0, s_1, s_2\}$ where s_0 is the initial state, s_2 is the only accepting state and $\Delta = \{(s_0, 0, s_0), (s_0, 1, s_0), (s_0, 0, s_1)\}, (s_1, 1, s_2)\}$ is the transition relation.

(a) What does it mean to say that a string x on the language A^* is accepted by N?

(5 points)

(b) Use the powerset construction to convert N into an equivalent Deterministic Finite State Automaton M. (See Homework 5 question 3.)

(10 points)

(c) If possible, simplify the automaton M you obtained in (b) (e.g., by eliminating states unreachable from the initial state).

(5 points)

(d) Describe in English the language $\mathcal{L}$ accepted by N or give a regular expression denoting it.

(5 points)

Answer: (a) A string $x = a_0 a_1 \ldots a_{n-1}$ on the language A^* is accepted by N if and only if there is a sequence of states q_0, q_1, ..., q_{n-1}, q_n such that $q_0 = s_0$, $q_n = s_1$ and for each $i < n$, $(s_i, a_i, s_{i+1}) \in \Delta$ (i.e., with input x there is a sequence of transitions admissible by the relation Δ from the initial state s_0 to the finale state s_1).

(b) Using the powerset construction $M = (A, \wp(S), \{s_0\}, \{X \subseteq S | s_1 \in X\}, \nu)$ where ν is given by the table

	0	1
$\emptyset$	$\emptyset$	$\emptyset$

$\{s_0\}$	$\{s_0, s_1\}$	$\{s_0\}$
$\{s_1\}$	$\emptyset$	$\{s_2\}$
$\{s_2\}$	$\emptyset$	$\emptyset$
$\{s_0, s_1\}$	$\{s_0, s_1\}$	$\{s_0, s_2\}$
$\{s_0, s_2\}$	$\{s_0, s_1\}$	$\{s_0\}$
$\{s_1, s_2\}$	$\emptyset$	$\{s_2\}$
$\{s_0, s_1, s_2\}$	$\{s_0, s_1\}$	$\{s_0, s_2\}$

(the transition function ν can also be given as a flow-graph).

(c) In M the states $\emptyset$, $\{s_1\}$, $\{s_2\}$, $\{s_1, s_2\}$ and $\{s_0, s_1, s_2\}$ are never reachable from the initial state $\{s_0\}$, as it is obvious from the flow-graph of ν. Therefore M can be simplified to $M' = (A, \{X, Y, Z\}, X, \{Z\}, \nu')$ where $X = \{s_0\}$, $Y = \{s_0, s_1\}$, $z = \{s_0, s_2\}$ and ν' is given by the table

	0	1
$\{s_0\}$	$\{s_0, s_1\}$	$\{s_0\}$
$\{s_0, s_1\}$	$\{s_0, s_1\}$	$\{s_0, s_2\}$
$\{s_0, s_2\}$	$\{s_0, s_1\}$	$\{s_0\}$

(or by the corresponding flow-graph).

(d) $\mathcal{L}$ is the set of strings of zeros and ones that end with a substring of the form $\mathbf{01}$. A regular expression denoting $\mathcal{L}$ is the following: $(\mathbf{0} + \mathbf{1})^*\mathbf{01}$.

Question 3. (*25 points*) (a) Outline an elementary proof of the theorem: *there are infinitely many prime numbers of the form* $4k - 1$.

(10 points)

(b) Implicit in the proof there is an algorithm to define the following function:

$$p(n) = p_n, \qquad \text{the } n\text{-th prime number of the form } 4k - 1,$$

starting from $p_1 = 3$. Show that the function $p(n)$ is primitive recursive.
You can use the fact that the factorial function $x!$ is primitive recursive and that the relations $x < y$ and $x|y$ (x divides y) are primitive recursive.

(15 points)

(Hint: Consider the predicate $\mathrm{Pr}(x)$ (x *is prime*) defined as

$$\mathrm{Pr}(x) \equiv 1 < x \ \&\ \neg(\exists c. 1 < c < x \ \&\ c|x).$$

First show that $\mathrm{Pr}(x)$ is primitive recursive; then define the function $p(x)$ by the recursion scheme, using the bounded μ-operator.)

Answer: (a) The number 3 is the first prime number of the form $4k - 1$, $p_1 = 3$. Let p_n be the n-th prime number of the form $4k - 1$ and consider $c = 4(p_n!) - 1$.

Suppose that *there is no prime number p of the form $4k-1$ such that $p_n < p < c$ and $p|c$.* We claim that under this hypothesis c is prime. Indeed the number c is greater than 1 and it is not divisible by 2.

Suppose c was divisible by a prime p of the form $4k-1$, i.e., $p \cdot d = c$ for some d. Then by our hypothesis $p \leq p_n$ and so $p|4(p_n!)$, namely, $p \cdot e = p_n!$ where $e = 4 \cdot p_n \cdot (p_n - 1) \cdot \ldots \cdot (p+1) \cdot (p-1) \cdot \ldots \cdot 1 = 4(p_n!)/p$. But then $1 = (p \cdot e) - (p \cdot d) = p \cdot (e - d)$ and this is a contradiction, as no prime divides 1.)

Finally c cannot be divided only by primes of the form $4k+1$: indeed it is easy to show that any product of primes of the form $4k+1$ is still of this form. Therefore our hypothesis implies that c is prime.

In conclusion, either our hypothesis is false or c is prime. In either case *there is a prime number p of the form $4k-1$ such that $p_n < p \leq c$.* We let p_{n+1} be the least such p.

(b) The relation $\Pr(x) \equiv 1 < x \ \& \ \neg(\exists c.1 < c < x \ \& \ c|x)$ is primitive recursive, because it is defined by conjunction, negation and bounded quantification from the primitive recursive relations $x < y$ and $x|y$. Now $p(i)$ is primitive recursive because it is defined by the scheme of primitive recursion using also the composition scheme:

$$
\begin{array}{rcl}
p(1) & = & 3; \\
p(n+1) & = & \chi(p(n)), \qquad \text{where } \chi(z) = \mu x_{z<x<z!} \Pr(x) \& rm(x,4) = 3.
\end{array}
$$

Question 4. (*25 points*)

(a) Prove that the set of all reals r such that $0 < r < 1$ is uncountable.

(8 points)

(b) Prove that the set of all Turing-computable functions $f : \mathbf{N} \to \mathbf{N}$ is countable.

(12 points)

(c) Say that a real r such that $0 < r < 1$ is *recursive* if there is a total recursive function f such that $r = 0.f(1)f(2)f(3)\ldots$. Conclude that there are non-recursive reals.

(5 points)

Hints for (b): You may assume that every Turing machine is defined using the same alphabet A and a unique set of symbols for states. Recall that every Turing Machine can be uniquely identified by a sequence of quintuples $\{(a_1, b_1, c_1, d_1, e_1), \ldots, (a_n, b_n, c_n, d_n, e_n)\}$ (lexicographically ordered). Moreover, you can assume that there exists a *Gödel numbering*, i.e., an *injective* mapping $G : A \cup S \to \mathbf{N}$ for the symbols of the alphabet, from which you will define a Gödel numbering for the strings of symbols and for sequences of strings as explained in class. More precisely, you must do the following:

(i) explain what the quintuples mean;

(2 points)

(ii) define a Gödel numbering of quintuples, given the Gödel numbering $G : A \to \mathbf{N}$ for the symbols of the alphabet (Hint: use prime numbers);

(2 points)

(iii) define a Gödel numbering of sequences of quintuples, given the Gödel numbers of quintuples (Hint: use prime numbers);

(2 points)

(iv) show that the Gödel numbering given by (ii) and (iii) is an *injective* map from the set of quintuples to the natural numbers (Hint: use the fundamental theorem of arithmetic).

(6 points)

This answers the question, because the mapping G gives an enumeration (with gaps and repetitions) of all Turing computable functions and we can conclude that the set of all Turing computable functions is countable.

Answer: (a) Lecture notes, page 1.14.

(b) (i) A quintuple (a, b, c, d, e) means: if in state a the machine reads the input symbol b, then it goes to state c, prints symbol d and goes left or right depending on whether $e = L, H, R$ (or $e = \leftarrow, -, \rightarrow$).

(ii) Given the Gödel numbering $G(x)$ of the symbols, the Gödel numbering of the quintuples is given by

$$G(a, b, c, d, e) = p_1^{G(a)} \cdot p_2^{G(b)} \cdot p_3^{G(c)} \cdot p_4^{G(d)} \cdot p_5^{G(e)};$$

where p_i is the i-th prime number (starting from $p_1 = 2$).

(iii) If $Q_1, \ldots, Q_n$ are all the quintuples that define a Turing machine M (in the lexicographical order or in the order given by their Gödel numbering), then

$$G(Q_1, \ldots, Q_n) = p_1^{G(Q_1)} \cdot \ldots \cdot p_n^{G(Q_n)}$$

is the Gödel number of the machine.

(iv) Suppose g is the Gödel numbering of two machines M and M'. By the fundamental theorem of arithmetic $g = p_1^{\alpha_1} \cdot \ldots \cdot p_n^{\alpha_n}$ for some n and the factorization is unique. Since g is the Gödel number of a Turing Machine, $\alpha_1, \ldots, \alpha_n$ are Gödel numbers of quintuples, hence $\alpha_i = p_1^{G(a_i)} \cdot p_2^{G(b_i)} \cdot p_3^{G(c_i)} \cdot p_4^{G(d_i)} \cdot p_5^{G(e_i)}$ for some quintuple $(a_i, b_i, c_i, d_i, e_i)$; again by the fundamental theorem of arithmetic, this factorization is unique. This means that the machines M and M' contain exactly the same quintuples, i.e., they are the same machine.

(c) By Chapters 6, 7 and 8 of Boolos and Jeffrey, the class of partial recursive function and that of Turing computable functions coincide. By part (b) there is an injection G from the set of Turing Machines to the natural numbers $\mathbf{N}$, while by part (a) there is no bijection between the reals in $(0, 1)$ and $\mathbf{N}$. Therefore there must be a real r which is not given by any recursive function.

Question 5. (*25 points*) Consider the Ackermann function

$$\begin{array}{rcll} \alpha(m, 0) & = & m + 1 & \text{(i)} \\ \alpha(0, n+1) & = & \alpha(1, n) & \text{(ii)} \\ \alpha(m+1, n+1) & = & \alpha(\alpha(m, n+1), n) & \text{(iii)} \end{array}$$

(a) Prove by induction on n that

$$n + n < \alpha(n, 2).$$

You may use the *Fact*, given in class, that $\alpha(m, n+1) \geq \alpha(m+1, n)$, for all m, n, and also the fact that the Ackermann function is monotone in both arguments (for $i < j$, $m < n$, $\alpha(i, m) < \alpha(j, m)$ and $\alpha(i, m) < \alpha(i, n)$).

(15 points)

(b) Outline the proof given in class that the function $\beta(n) = \alpha(n, n) + 1$ is not primitive recursive.

(10 points)

(*Hint:* Use the *Lemma* that for every primitive recursive function $f(x_1, \ldots, x_k)$ there exists an $n \in \mathbf{N}$ such that for all $x_1, \ldots, x_k$, $f(x_1, \ldots, x_k) < \alpha(max(x_1, \ldots, x_k), n)$.)

Answer. (a) We have the *base case*

$$\begin{array}{rcll} \alpha(0,2) & = & \alpha(1,1) & \text{by (ii)} \\ & = & \alpha(\alpha(0,1),0) & \text{by (iii)} \\ & = & \alpha(0,1)+1 & \text{by (i)} \\ & = & \alpha(1,0)+1 & \text{by (ii)} \\ & = & 3 & \text{by (i)} \\ & > & 0 = 0+0 & \end{array}$$

The inductive step is left as an exercise to you.

(b) Suppose $\beta(n)$ was primitive recursive. Then by the Lemma, there is a k such that $\beta(m) < \alpha(m, k)$ for all m. Therefore

$$\begin{array}{rcll} \alpha(k,k)+1 & = & \beta(k) & \text{definition of } \beta \\ & < & \alpha(k,k) & \text{by the Lemma} \end{array}$$

a contradiction. Therefore $\beta(n)$ is not primitive recursive.

Question 6. (*25 points*) Let $\mathcal{L} = (M_1, M_2, \ldots)$ be an enumeration (without gaps and repetitions) of the set of all Turing Machines that compute partial functions from $\mathbf{N}$ to $\mathbf{N}$ and let f_i be the partial function computed by M_i.

(a) What does it mean to say that a Turing Machine M^U is a universal machine for the list $\mathcal{L}$? You may assume that such machines exist, but you do not need to define one in detail.

(5 points)

(b) Using (a) show that there is a Turing Machine M' which computes the following partial function g:

$$\begin{array}{lcl} g(n) & = 0 & \text{if } M_n \text{ with input } n \text{ terminates in accepting state;} \\ g(n) & \text{is undefined} & \text{otherwise.} \end{array}$$

(5 points)

(c) Show that there is no Turing Machine M^H which computes the following partial function h:

$$\begin{array}{lll} h(n) & = 0 & \text{if } M_n \text{ with input } n \text{ does not terminate in accepting state;} \\ h(n) & \text{is undefined} & \text{otherwise.} \end{array}$$

(5 points)

(d) Consider the set

$$K = \{M_n | M_n \text{ with input } n \text{ terminates in accepting state}\}$$

and its complement $\overline{K}$. Explain what it means to say that a set A is *recursively enumerable*. Show that the set K is recursively enumerable, but $\overline{K}$ is not.

(5 points)

(e) Explain what it means to say that a set A is *semidecidable*. Briefly recall Church's Thesis, and explain why, assuming Church's Thesis, we can say that K is semidecidable and $\overline{K}$ is not.

(5 points)

Answer. (a) A Turing Machine M^U is universal for the list $\mathcal{L}$ if M^U behaves as an *interpreter* for the Turing machines in $\mathcal{L}$. Namely, M^U takes as input a coding $\underline{M_n}$ of Turing machine M_n (i.e., the program as a datum, representable by the Gödel number of M_n) and an input k and behaves as follows:

$$\begin{array}{lll} M^U(\underline{M_n}, k) & \text{terminates with output } f_n(k) & \text{if } M_n(k) \text{ terminates with output } f_n(k) \\ M^U(\underline{M_n}, k) & \text{undefined} & \text{if } M_n(k) \text{ undefined.} \end{array}$$

(b) Define M' as follows: $M'(\underline{M_n}, n) = 0$ if $M^U(\underline{M_n}, n)$ terminates; $M'(\underline{M_n}, n)$ undefined otherwise.

(c) Suppose M^H computed the partial function h. Then M^H occurs at some point in the list $\mathcal{L}$, say $M^H = M_n$. Now

$$\begin{array}{lrl} h(n) & = \quad 0 & \text{if } M_n \text{ with input } n \text{ does not terminate in accepting state;} \\ h(n) & \text{is undefined} & \text{otherwise.} \end{array}$$

• Suppose $h(n)$ is undefined: by definition of h, M_n with input n terminates in an accepting state. But $M_n = M^H$ computes h, therefore $h(n)$ is defined, a contradiction;
• Suppose $h(n)$ is defined: since M^H computes h, M^H with input n terminates in accepting state; but $M^H = M_n$, so by definition of h, we have $h(n)$ undefined, a contradiction.
In either cases, we have a contradiction; the only remaining assumption, that h is computable by a Turing Machine, is therefore false.

(d) A set is A *recursively enumerable* if either A is empty or there is a total recursive function $f : \mathbf{N} \to A$ which is surjective. By Chapters 6, 7 and 8 of Boolos and Jeffrey, a partial function is recursive if and only if it is Turing computable. Therefore we can say that a set A is recursively enumerable if and only if there is a total Turing computable function $f : \mathbf{N} \to A$ which is surjective. It has been shown in class (Proposition 4, page 3.16 in the lecture notes) that a set A is recursively enumerable if and only if there is a Turing machine M which accepts A. By part (b) the set K is recursively enumerable and by part (c) its complement $\overline{K}$ is not.

(e) Church's Thesis claims that every function which is effectively computable by some mechanical procedure (abstraction being made on the resources available for computation) is also computable by a Turing Machine. (This claim is supported by evidence that the various formal definitions so far proposed for effectively computable classes of functions have been proved equivalent to the definition of Turing computability.)
By Church's Thesis, *effectively computable* can be identified with *Turing computable*. Therefore (b) shows that the set K is semidecidable and (c) shows that its complement $\overline{K}$ is not semidecidable.