Assignment 2

CS60005: Foundations of Computing Science — Autumn, 2021
Deadline: 8th November, 23:59 — Total Marks: 20

Solve all problems. Stick to notation used in the classes.
Write solutions on white paper, scan and then upload a single pdf file. Make sure that the file size does not exceed 20 MB. Any format other than pdf is not acceptable.
Upload in CSE-Moodle course page (suitable entry is already created)

---

1. A *one-counter automaton* is an automaton with a finite set of states Q, a two-way read-only input head and a separate counter that can hold any non-negative integer. The input x is enclosed in endmarkers $\vdash, \dashv \notin \Sigma$ and the input head may not go outside the endmarkers. The machine starts in its start state s with its counter set to 0 and with its input head pointing to $\vdash$. In each step, it can test its counter for 0. Based on this information, its current state and the symbol its input head is currently reading, it can either add $1, -1$ to its counter and move its input head either left or right and enter a new state. It accepts by entering a distinguished final state t.

   (a) Give a rigorous formal definition of these machines, including a definition of acceptance. Your definition should begin as follows: "A one-counter automaton is a 7-tuple $\mathcal{M} = (Q, \Sigma, \vdash, \dashv, s, t, \delta)$, where …". [2]

   Solution: A one-counter automaton is a 7-tuple $\mathcal{M} = (Q, \Sigma, \vdash, \dashv, s, t, \delta)$, where

   - Q is a finite set of states
   - Σ is the input alphabet
   - $\vdash$, $\dashv$ are left and right endmarkers
   - s is the start state
   - t is the accept state
   - $\delta : Q \times \Gamma \times \{Z, NZ\} \to Q \times \{L, R\} \times \{1, -1\}$ is the transition function with Z, NZ denoting the counter being zero, non-zero respecively. Here $\Gamma = \Sigma \cup \{\vdash, \dashv\}$.

   The following restrictions apply:

   - $\forall p \in Q\, \forall A \in \{Z, NZ\}, \exists q \in Q\; c \in \{1, -1\}$ such that $\delta(p, \vdash, A) = (q, R, c)$
   - $\forall p \in Q\; A \in \{Z, NZ\}, \exists q \in Q\; c \in \{1, -1\}$ such that $\delta(p, \dashv, A) = (q, L, c)$
   - $\forall p \in Q \setminus t\; a \in \Gamma, \exists q \in Q\; Y \in \{L, R\}$ such that $\delta(p, a, Z) = (q, Y, 1)$
   - $\forall a \in \Gamma\; A \in \{Z, NZ\}, \exists Y \in \{L, R\}\; c \in \{1, -1\}$ such that $\delta(t, a, A) = (t, Y, c)$

   Define a configuration of $\mathcal{M}$ to be a string in $Q \times \Sigma^* \times \mathbb{N} \times \mathbb{N}$ containing the current state, contents of the tape, position of the tape head and value of the counter. The start configuration on input x is $(s, \vdash x \dashv, 0, 0)$. Further, define for $m \neq 0$

$$(p, z, n, m) \xrightarrow[\mathcal{M}]{1} \begin{cases} (q, z, n+1, m-1) \text{ if } \delta(p, z_n, NZ) = (q, R, -1) \\ (q, z, n+1, m+1) \text{ if } \delta(p, z_n, NZ) = (q, R, 1) \\ (q, z, n-1, m-1) \text{ if } \delta(p, z_n, NZ) = (q, L, -1) \\ (q, z, n-1, m+1) \text{ if } \delta(p, z_n, NZ) = (q, L, 1) \end{cases}$$

   and

$$(p, z, n, 0) \xrightarrow[\mathcal{M}]{1} \begin{cases} (q, z, n+1, 1) \text{ if } \delta(p, z_n, Z) = (q, R, 1) \\ (q, z, n-1, 1) \text{ if } \delta(p, z_n, Z) = (q, L, 1) \end{cases}$$

The machine $\mathcal{M}$ is said to accept a string x if $(s, \vdash x \dashv, 0, 0) \xrightarrow[\mathcal{M}]{*} (t, \vdash x \dashv, m, n)$ for some $m, n \in \mathbb{N}$.

Note that the input tape is read-only. A configuration for a fixed input may be defined without the second component as well.

(b) Prove that the membership problem (given $\mathcal{M}, x$, does $\mathcal{M}$ accept x?) for deterministic one-counter automata is decidable. 2

Solution: At any point of time, $\mathcal{M}$ can be in one of $|Q|$ possible states and its tape head can be in one of $|x| + 2$ possible positions. The machine loops on a given input whenever the first three components of a configuration (p, z, n, m) repeats with a counter value $\geq m$. So, for a halting sequence of configurations, the counter value can never exceed $t = |Q| \times (|x|+2)$. For otherwise, atleast one combination of state and tape-head position must repeat as the counter is incremented each step of the computation thus making the machine loop. This immediately gives us a decision procedure for testing if $\mathcal{M}$ accepts x – simulate $\mathcal{M}$ on x for t^2 steps; if the first three components of any configuration repeats with an increased value of counter, then the machine loops; otherwise, if $\mathcal{M}$ halts, check whether it enters the accept or reject state and accordingly determine if $\mathcal{M}$ accepts x.

2. Describe a language over alphabet $\{0\}$ for each of the following classes and justify.

(a) Regular

Solution:

$$\{0^n \mid n \bmod 2 \equiv 0\}$$

Regular expression for the language: $(00)^*$.

(b) Recursive but not context-free

Solution:

$$\{0^{n^2} \mid n \in \mathbb{N}\}$$

Possible to design a total TM recognising the language. Hence recursive.

Suppose that L is context free. Then by pumping lemma, there exists a constant k such that for every string $s \in L$ of length $\geq k$, we can write $s = uvwxy$ s.t., $|vx| \geq 1$, $|vwx| \leq k$ and $uv^iwx^iy \in L$ for all $n \geq 0$. Consider the language $\mathsf{SQ} = \{0^{n^2} \mid n \in \mathbb{N}\}$ and suppose that it is context-free. Pumping lemma guarantees existence of a constant k s.t., $s = 0^{k^2}$ can be written as $s = uvwxy$ with $|vx| = \ell$ and $0 < \ell \leq k$. The string uv^2wx^2y must be in SQ. But $|uv^2wx^2y| = k^2 + \ell \leq k^2 + k$. But SQ does not contain any string of length between k^2 and $(k+1)^2$ whereas $k^2 < k^2 + k < (k+1)^2$. This contradicts out assumption that SQ is context-free.

(c) Recursively enumerable but not recursive

Solution: For an alphabet Σ, Σ^* is countable and hence there exists a 1-1 map $\tau : \Sigma^* \to \mathbb{N}$. Consider an enumeration of all Turing machines encoded as strings over Σ. Every TM $\mathcal{M}_x$ in the list (where $x \in \Sigma^*$) can be encoded by an integer given by $\tau(x)$. Note that $\tau(x)$ also encodes the string x. Now, consider the halting problem. Every instance $(\mathcal{N}, x)$ of HP can be encoded using two integers. We can encode the every $a \in \mathbb{N}$ using the unary alphabet $\{0\}$ as 0^a. Putting it all together, we have $\mathsf{HP} = \{(0^{\tau(y)}, 0^{\tau(x)}) \mid \mathcal{M}_y \text{ halts on input } x\}$, which is *r.e.* but not recursive.

5 = (1+2+2)

3. Let L be the set of Turing machines $\mathcal{M}$ with input alphabet Σ such that $\mathcal{M}$ writes the symbol $a \in \Sigma$ at some point on its tape. Show that L is undecidable. 3

Solution: We show that L is undecidable via a reduction from HP. Let $(\mathcal{N}, x)$ be an instance of HP with Γ being the tape alphabet of $\mathcal{N}$. Let $\mathcal{M}$ be Turing machine defined as follows: tape alphabet consists of encodings of Γ with $\Sigma \setminus \{a\}$. (Additional delimiter symbols may also be included.) $\mathcal{M}$ erases its own input and simulates $\mathcal{N}$ on input x (here, x is also encoded with symbols from $\Sigma \setminus \{a\}$). If $\mathcal{N}$ halts on x, $\mathcal{M}$ enters a new state q; while in q, $\mathcal{M}$ writes a on the tape and then halts. If $\mathcal{N}$ halts on x, then $\mathcal{M}$ writes a on its tape. Otherwise $\mathcal{M}$ never writes a on its tape.

4. Suppose that $\mathbf{P} \neq \mathbf{NP}$. Prove that it is undecidable, given $L \in \mathbf{NP}$, whether or not $L \in \mathbf{P}$. [3]

Solution: Let Q be a property on *r.e.* sets defined as follows: conditioned on $L \in \mathbf{NP}$, $Q(L) = \top$ if $L \in \mathbf{P}$ and and $Q(L) = \bot$ otherwise. Assuming that $\mathbf{P} \neq \mathbf{NP}$, no $\mathbf{NP}$-complete language can be in $\mathbf{P}$. So $Q(L) = \top$ for any $L \in \mathbf{P}$ and $Q(L) = \bot$ for any language $L \in \mathbf{NP}$-Complete. Hence Q is a non-trivial property and by Rice's theorem Q is undecidable.

5. A language L is in class $\mathbf{DP}$ (where $\mathbf{D}$ stands for difference) iff there are languages $L_1 \in \mathbf{NP}$ and $L_2 \in \mathbf{coNP}$ so that $L = L_1 \cap L_2$.

   (a) Define completeness for the class $\mathbf{DP}$ under polynomial time reductions. [0.5]

   Solution: A language L is $\mathbf{DP}$-Complete if $L \in \mathbf{DP}$ and for all $A \in \mathbf{DP}$, $A \leq_p L$.

   (b) The problem $\mathsf{SAT\text{-}UNSAT}$ is defined as the set of all pairs of Boolean formulae $\langle \phi, \psi \rangle$ such that ϕ is satisfiable and ψ is unsatisfiable. Show that $\mathsf{SAT\text{-}UNSAT}$ is $\mathbf{DP}$-complete. [2.5]

   Solution: Define $L_1 \in \mathbf{NP}, L_2 \in \mathbf{coNP}$ as follows.

$$L_1 = \{\langle \phi, \psi \rangle \mid \phi \in \mathsf{SAT} \text{ and } \psi \text{ is any Boolean formula}\}$$

$$L_2 = \{\langle \phi, \psi \rangle \mid \phi \text{ is any Boolean formula and } \psi \in \mathsf{UNSAT}\}$$

   Clearly $\mathsf{SAT\text{-}UNSAT} = L_1 \cap L_2$ and hence $\mathsf{SAT\text{-}UNSAT} \in \mathbf{DP}$.

   We know that L_1 is $\mathbf{NP}$-Complete and L_2 is $\mathbf{coNP}$-complete. Let $A \in \mathbf{DP}$. Then $A = A_1 \cap A_2$ where $A_1 \in \mathbf{NP}$ and $A_2 \in \mathbf{coNP}$. We have $A_1 \leq_p L_1$ and $A_2 \leq_p L_2$. Let f_1, f_2 be the corresponding reduction functions. For $x \in A_1 \cap A_2$, let $f_1(x) = \langle \phi_1, \psi_1 \rangle$ and $f_2(x) = \langle \phi_2, \psi_2 \rangle$. Then $\langle \phi_1(x), \psi_2(x) \rangle$ is in $\mathsf{SAT\text{-}UNSAT}$. Furthermore,

$$\begin{aligned} \langle \phi_1(x), \psi_2(x) \rangle \in \mathsf{SAT\text{-}UNSAT} &\iff \phi_1 \in \mathsf{SAT} \text{ and } \psi_2 \in \mathsf{UNSAT} \\ &\iff f_1(x) \in L_1 \text{ and } f_2(x) \in L_2 \\ &\iff x \in A_1 \text{ and } x \in A_2 \\ &\iff x \in A_1 \cap A_2 = A \end{aligned}$$

   Hence $A \leq_p \mathsf{SAT\text{-}UNSAT}$ for any $A \in \mathbf{DP}$ and $\mathsf{SAT\text{-}UNSAT}$ is $\mathbf{DP}$-Complete.

   (c) A (undirected) graph G is *Hamiltonian* if it contains a Hamiltonian cycle (a cycle visiting every vertex exactly once). The language $\mathsf{HC\text{-}CRITICAL}$ consists of all graphs G such that G is not Hamiltonian but adding any edge to G will make it Hamiltonian. Show that $\mathsf{HC\text{-}CRITICAL}$ is in $\mathbf{DP}$. [2]

   Solution: Let HC be the set of all Hamiltonian graphs. A certificate for a graph to be Hamiltonian would consist of the sequence of vertices in the Hamiltonian cycle. Clearly the certificate is verifiable in deterministic polynomial time and hence $\mathsf{HC} \in \mathbf{NP}$.

   Let

$$L_1 = \{G \mid \text{ adding any edge to } G \text{ makes it Hamiltonian}\}$$

and

$$L_2 = \{G \mid G \text{ is not Hamiltonian}\}.$$

We have $L_2 = \neg\mathsf{HC}$ and hence $L_2 \in \mathbf{coNP}$. Now, consider L_1 and a "yes" instance $G = (V, E)$ of L_1. A certificate for G would contain for each edge $e \in E$, a sequence of vertices of length $|V|$ forming a Hamiltonian cycle in $G_e = (V, E \cup \{e\})$. Since there are at most $|V|^2$ possible edges, the length of the certificate is polynomial in the size of the instance. Also, the certificate can be verified deterministically in polynomial time. Hence $L_1 \in \mathbf{NP}$.

We have $\mathsf{HC\text{-}CRITICAL} = L_1 \cap L_2$ where $L_1 \in \mathbf{NP}$ and $L_2 \in \mathbf{coNP}$; therefore $\mathsf{HC\text{-}CRITICAL} \in \mathbf{DP}$.