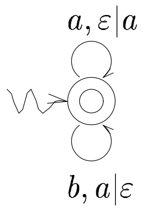
Review Questions

Mark all correct answers for each of the following questions.

Σ denotes an arbitrary alphabet and L an arbitrary language over Σ, unless otherwise specified.

1. Let $M = (\{s\}, \{a, b\}, \{a\}, \Delta, s, \{s\})$ be the following automaton:



Let $G = (N, \{a, b\}, R, S)$ be the context-free grammar, constructed according to the algorithm presented in class, satisfying $L(G) = L(M)$. Let M_1 be the automaton obtained from M upon adjoining the transition $((s, c, a), (s, a))$ to Δ, and M_2 the automaton obtained from M upon adjoining the transition $((s, c, a^2), (s, a^2))$ to Δ. (Each time, the alphabet Σ is augmented from $\{a, b\}$ to $\{a, b, c\}$.) Let

$$G_1 = (N_1, \{a, b, c\}, R_1, S), \qquad G_2 = (N_2, \{a, b, c\}, R_2, S)$$

be the corresponding context-free grammars.

Let $L \subseteq \{a, b, c\}^*$ be the language consisting of all words w over $\{a, b, c\}$ satisfying $|w|_a = |w|_b$ and $|u|_a \geq |u|_b$ for every prefix u of w.

(a) $|R_1| = |R| + 1$.

(b) $L(M_1) = L$.

(c) The language $L - L(M_1)$ is infinite.

(d) $|R_2| = |R| + 10$.

(e) N_2 consists of 9 letters, but only for 7 of them there exists a rule in R_2 allowing us to substitute them by some word.

(f) $L(M_1) - L(M_2)$ is infinite.

(g) None of the above.

2. Let M_1 and M_2 be Turing machines with the same alphabets Σ and Γ and let $w \in \Sigma^*$.

(a) If M_1 halts for the input w, then so does M_1M_2.

(b) If M_1 does not halt for the input w, then so does M_1M_2.

(c) If M_2 does not halt for the input w, then so does M_1M_2.

(d) If M_1M_2 does not halt for the input w, then either there exists a prefix u of w such that M_1 does not halt for u or there exists a suffix v of w such that M_2 does not halt for v.

(e) If $L(M_1M_2) \neq \emptyset$, then $L(M_2) \neq \emptyset$.

(f) If $L(M_1) = L(M_2) = \Sigma^*$, then $L(M_1M_2) = \Sigma^*$.

(g) None of the above.

3. (a) Let $M = (Q, \Sigma, \Gamma, \delta, s, h)$ be a Turing machine. Let $w \in \Sigma^*$ be an input for which M neither halts nor hangs. Define the configuration $u_nq_nv_n$ for $n = 0, 1, \ldots,$ as the configuration at which M arrives after n steps. (That is, inductively, $u_0q_0v_0 = sw$ and $u_nq_nv_n \vdash_M u_{n+1}q_{n+1}v_{n+1}$.) Then the set $\{u_n : n \geq 0\}$ may be infinite, but the set $\{v_n : n \geq 0\}$ is necessarily finite.

(b) For every Turing-decidable language L there exists a constant C (depending on L) and a Turing machine M deciding L, such that for every input $w \in \Sigma^*$ the computation takes at most C steps. However, there exists no constant C having this property for all Turing-decidable languages.

(c) Given an arbitrary language L, there exists a Turing machine $M = (Q, \Sigma, \Gamma, \delta, s, h)$ and $q \in Q$ such that for every $w \in L$ we have $sw \vdash_M^* q\textcircled{\scriptsize Y}$ and for every $w \notin L$ we have $sw \vdash_M^* q\textcircled{\scriptsize N}$.

(d) If L^* is Turing-decidable, then L is Turing-decidable as well.

(e) Let $L \subseteq \Sigma^*$ be a language for which there exists a Turing-decidable language $L_1 \subseteq \Sigma^*$ and a Turing machine M computing a function $f : \Sigma^* \longrightarrow \Sigma^*$ possessing the properties $f(L) \subseteq L_1$ and $f(\Sigma^* - L) \subseteq \Sigma^* - L_1$. Then L is Turing-decidable.

(f) Let $L \subseteq \Sigma^*$ be a language having the property that for each $w \in L$ there exists a Turing machine M with input alphabet Σ such that $sw \vdash_M^* h\textcircled{\scriptsize Y}$ and for each $w \in \Sigma^* - L$ there exists a Turing machine M with input alphabet Σ such that $sw \vdash_M^* h\textcircled{\scriptsize N}$. Then L is Turing-decidable.

(g) None of the above.

4. Let M_1, M_2, M_3 be Turing machines computing the functions f_1, f_2, f_3 : $\{0,1\}^* \longrightarrow \{0,1\}^*$, respectively, where:
$f_1(w) = w^2, \qquad w \in \{0,1\}^*.$
$f_2(w) = \sigma_1^n \sigma_2^{n-1} \ldots \sigma_n, \qquad w = \sigma_1\sigma_2 \ldots \sigma_n \in \{0,1\}^*.$
$f_3(w) = 1^{|w|}0, \qquad w \in \{0,1\}^*.$
All three machines have the head pointing at the leftmost square on the tape when they halt.

(a) All three machines compute functions from $\mathbf{Z}_+$ to $\mathbf{Z}_+$. However, there exist machines computing functions from $\{0,1\}^*$ but not from $\mathbf{Z}_+$ to $\mathbf{Z}_+$.

(b) The machine M_1M_3 computes a function $g_{13} : \mathbf{Z}_+ \longrightarrow \mathbf{Z}_+$, and $g_{13}(200) = 401$.

(c) The machine M_2 computes a function $g_2 : \mathbf{Z}_+ \longrightarrow \mathbf{Z}_+$, and $g_2(999) = 499499$.

(d) The machine M_3^{38} computes a function $g_{3,38} : \mathbf{Z}_+ \longrightarrow \mathbf{Z}_+$, and $g_{3,38}(53) = 92$.

(e) If M is a Turing machine computing both a function f from $\{0,1\}^*$ to $\{0,1\}^*$ and a function g from $\mathbf{Z}_+$ to $\mathbf{Z}_+$, then f is injective if and only if g is.

(f) If M is a Turing machine computing both a function f from $\{0,1\}^*$ to $\{0,1\}^*$ and a function g from $\mathbf{Z}_+$ to $\mathbf{Z}_+$, and f is surjective, then so is g.

(g) None of the above.

5. (a) If M_1 is a Turing machine computing the function $g_1 : \mathbf{Z}_+ \longrightarrow \mathbf{Z}_+$ given by

$$g_1(n) = n^2, \qquad n \in \mathbf{Z}_+\,,$$

then the machine M_1^2 computes the function $g_{12} : \mathbf{Z}_+ \longrightarrow \mathbf{Z}_+$ given by

$$g_{12}(n) = n^4, \qquad n \in \mathbf{Z}_+\,.$$

(b) The Turing machine M_2 represented by the diagram

$$\begin{tikzcd}
{} \arrow[r, squiggly] & W_1 & T_R \arrow[l, bend right, "0"'] \arrow[loop right, "1"] \arrow[d, "B"'] \\
& & W_0
\end{tikzcd}$$

computes a function $g_2 : \mathbf{Z}_+^3 \longrightarrow \mathbf{Z}_+$, and $g_2(33, 42, 35) = 113$.

(c) The Turing machine M_3 represented by the diagram

$$\begin{tikzcd}
{} \arrow[r, squiggly] & W_B\,T_R \arrow[r, "{0,1}"] \arrow[d, "B"'] & T_R \arrow[r, "{0,1}"] \arrow[d, "B"'] & T_R \arrow[ll, bend right, "{0,1}"'] \arrow[d, "B"'] & & & \\
& T_L \arrow[d, "{0,1}"] \arrow[drrrrr, bend right, "B"'] & T_L \arrow[d, "{0,1}"] \arrow[drrrr, bend right, "B"'] & T_L \arrow[d, "{0,1}"] \arrow[dr, "B"] & & & \\
& W_B \arrow[u, bend left] & W_B \arrow[u, bend left] & W_B \arrow[u, bend left] & W_1\,T_R \arrow[r] & W_1\,T_R \arrow[r] & W_0
\end{tikzcd}$$

computes a function $g_3 : \mathbf{Z}_+ \longrightarrow \mathbf{Z}_+$, and $g_3(37) = 1$.

(d) If M_4 is a Turing machine computing a function $g_4 : \mathbf{Z}_+ \longrightarrow \mathbf{Z}_+$ and $g_4(n) = n$ for all $n \geq 1$, then $g_4(0) = 0$.

(e) If M_5 is a Turing machine computing a function from $\mathbf{Z}_+$ to $\mathbf{Z}_+$, then it computes a function from $\mathbf{Z}_+^2$ to $\mathbf{Z}_+^2$, but the converse is not necessarily true.

(f) If M_6 is a Turing machine computing a function from $\{0,1\}^*$ to $\{0,1\}^*$, then for every positive integer k it computes a function from $\mathbf{Z}_+^k$ to $\mathbf{Z}_+^k$, and conversely.

(g) None of the above.

Solutions

1. Recall that the automaton M is not simple. To make it simple we added to Δ the transition $((s,a,a),(s,a^2))$ to obtain a new set of transitions Δ'. Upon adding the transition $((s,c,a),(s,a))$, the automaton stays simple. Hence the only rule R_1 contains on top of the rules of R is:
$\langle s,a,s\rangle \rightarrow c\langle s,a,s\rangle$.

   All words of the form c^n belong to L, but not to $L(M_1)$. Hence $L - L(M_1)$ is infinite.

   Upon adding the transition $((s,c,a^2),(s,a^2))$ to Δ', the automaton becomes non-simple. According to our algorithm, we need to add to Q a new state q and replace the additional transition by the two transitions $((s,\varepsilon,a),(q,\varepsilon))$ and $((q,c,a),(s,a^2))$. The new state q is not accepting, so there is still a single type-1 rule:
$S \rightarrow \langle s,\varepsilon,s\rangle$.
The transition $((s,a,\varepsilon),(s,a)$ contributes now two type-2 grammatical rules instead of one:

   - $\langle s,\varepsilon,s\rangle \rightarrow a\langle s,a,s\rangle$,
   - $\langle s,\varepsilon,q\rangle \rightarrow a\langle s,a,q\rangle$.

   The transition $((s,a,a),(s,a^2))$ contributes four type-2 grammatical rules instead of one:

- $\langle s, a, s\rangle \to a\langle s, a, s\rangle\langle s, a, s\rangle$,
- $\langle s, a, s\rangle \to a\langle s, a, q\rangle\langle q, a, s\rangle$,
- $\langle s, a, q\rangle \to a\langle s, a, s\rangle\langle s, a, q\rangle$,
- $\langle s, a, q\rangle \to a\langle s, a, q\rangle\langle q, a, q\rangle$.

The transition $((s, b, a), (s, \varepsilon))$ contributes two type-3 rules instead of one:

- $\langle s, a, s\rangle \to b\langle s, \varepsilon, s\rangle$,
- $\langle s, a, q\rangle \to b\langle s, \varepsilon, q\rangle$.

The transition $((s, \varepsilon, a), (q, \varepsilon))$ contributes two type-3 rules:

- $\langle s, a, s\rangle \to \langle q, \varepsilon, s\rangle$,
- $\langle s, a, q\rangle \to \langle q, \varepsilon, q\rangle$.

The transition $((q, c, a), (s, a^2))$ contributes two type-2 rules:

- $\langle q, a, s\rangle \to c\langle s, a, s\rangle\langle s, a, s\rangle$,
- $\langle q, a, s\rangle \to c\langle s, a, q\rangle\langle q, a, s\rangle$,
- $\langle q, a, q\rangle \to c\langle s, a, s\rangle\langle s, a, q\rangle$,
- $\langle q, a, q\rangle \to c\langle s, a, q\rangle\langle q, a, q\rangle$.

Finally, there are now two type-4 rules instead of one:

- $\langle s, \varepsilon, s\rangle \to \varepsilon$,
- $\langle q, \varepsilon, q\rangle \to \varepsilon$.

Altogether, $|N_2| = 9$ and $|R_2| = 17$. Only the letter $\langle q, \varepsilon, s\rangle$ has no rule which allows converting it to some other word. (The reason is that one cannot proceed in the automaton from the state q if the stack is empty.)

All words of the forms ac^nb belong to $L(M_1)$, but not to $L(M_2)$, so that $L(M_1) - L(M_2)$ is infinite.

Thus, only (a), (c) and (f) are true.

2. If M_1 halts for the input w, then the machine M_1M_2 will get to the state h_1 and continue to the initial state of M_2. However, at that point the word on the tape is in general not the original input word w (and even if it is – there is no reason to expect the head to point at the leftmost square of the tape). Hence there is no reason to expect M_1M_2 to behave one way or another. For the same reason, if it is given that M_2 does not halt for w, there is no reason to expect M_1M_2 not to halt; if and when M_1 halts, the word on the tape and the head's position may well be such that M_2 will arrive at the state h_2. On the other hand, if M_1 does not halt for w, then M_1M_2 will hang or enter an infinite loop when given the input w already at the first stage, while M_1 is active.

   It may be the case that M_1 and M_2 halt for every input while M_1M_2 halts for no input. For example, suppose $M_1 = W_{\gamma_0}$ for some $\gamma_0 \in \Gamma - (\Sigma \cup \{B\})$, while M_2 halts right away if it reads a letter in $\Sigma \cup \{B\}$ and hangs for any other letter. Then after the first stage of action of M_1M_2 the tape has a γ_0 at its leftmost square, so the machine will hang at the second stage.

   If $M_1 = T_R$ and $M_2 = T_L$, then $L(M_1M_2) = \Sigma^*$ while $L(M_2) = \emptyset$.

   Thus, only (b) is true.

3. Let M be a machine which, given any input, acts as follows. First it moves the input one square to the right. Then, at each stage, it first moves the head all the way to the left, then moves it back to the right and copies the letter at the rightmost written square to the square on its right. Clearly, if the input is, say, σ, then the v_n's will assume in the process any value in $\{\varepsilon, B\sigma\}\{\sigma\}^*$.

   A machine deciding a language erases any input and writes in its stead either $\textcircled{\scriptsize Y}$ or $\textcircled{\scriptsize N}$. Hence the computation must consume at least as many steps as the length of the input. (Actually, it must be at least twice, as the head needs to be at the beginning of the tape by the end of the computation.)

   Let M_1 be a machine deciding the language Σ^*, having the property that the head points at the leftmost square when the computation ends. Let M_2 be a machine which repetitively changes the letter at the current square from $\textcircled{Y}$ to $\textcircled{N}$ and from $\textcircled{N}$ to $\textcircled{Y}$. The machine $M = M_1M_2$ has the property that both $sw \vdash^*_M s_2\textcircled{Y}$ and $sw \vdash^*_M s_2\textcircled{N}$ for every $w \in \Sigma^*$.

Let L be any undecidable language. Adding or removing finitely many words from an undecidable language clearly leaves it undecidable, so that we may assume $L \supseteq \Sigma$. But then $L^* = \Sigma^*$ is decidable, while L is not.

Let L be a language with the properties in (e). By first moving the input one square to the right, then moving the head back to the leftmost letter of the input, letting M operate, and then moving the output one square to the left, we may assume M always leaves the head at the leftmost square when it halts. Now let M_1 be a machine deciding L_1. Then MM_1 decides L.

For every word $w \in \Sigma^*$ there exist both a machine M such that $sw \vdash^*_M h\textcircled{Y}$ and a machine M such that $sw \vdash^*_M h\textcircled{N}$. In fact, just take for the first a machine deciding Σ^* and for the second a machine deciding $\emptyset$. (In both cases we assume the head points at the leftmost square when the machine halts.)

Thus, only (c) and (e) are true.

4. A Machine computing a function from $\{0,1\}^*$ to $\{0,1\}^*$ computes also a function from $\mathbf{Z}_+$ to $\mathbf{Z}_+$ if and only if, whenever the input is of the form 1^n0 for some non-negative integer n, the output is of this form as well. In our case the machine M_1 does not compute a function from $\mathbf{Z}_+$ to $\mathbf{Z}_+$ since the output corresponding to any input contains twice as many zeros as the input. The machines M_2 and M_3 satisfy the condition, and are readily seen to compute the functions
$g_2(n) = (n+1) + n + \ldots + 2 = \frac{(n+1)(n+2)}{2} - 1, \qquad n \in \mathbf{Z}_+\,,$
$g_3(n) = n + 1, \qquad n \in \mathbf{Z}_+\,,$
respectively. In particular, $g_2(999) = \frac{1000 \cdot 1001}{2} - 1 = 500499$.

Since the head is returned to the beginning of the tape at the end of the operation of each machine, any concatenation of these machines computes the corresponding composition of the functions computed by the machines. Hence M_1M_3 computes the function f_{13} from $\{0,1\}^*$ to $\{0,1\}^*$ given by

$$f_{13}(w) = f_3(w^2) = 1^{2|w|}0, \qquad w \in \{0,1\}^*\,,$$

and M_3^{38} computes the function f_3^{38}:

$$f_3^{38}(w) = 1^{|w|+37}0, \qquad w \in \{0,1\}^* .$$

Therefore

$$g_{13}(n) = 2n + 2, \qquad n \in \mathbf{Z}_+ ,$$

and

$$g_{3,38}(n) = n + 38, \qquad n \in \mathbf{Z}_+ .$$

In particular, $g_{13}(200) = 402$ and $g_{3,38}(53) = 91$.

If M computes both a function f from $\{0,1\}^*$ to $\{0,1\}^*$ and a function g from $\mathbf{Z}_+$ to $\mathbf{Z}_+$, then g is obtained from the restriction of f to the set of inputs of the form 1^n0. Hence, if f is injective, then g is also injective. However, the converse is false; the machine M_3 provides a counter-example. If f is surjective, then g is not necessarily surjective, as it is possible that some words of the form 1^n0 are obtained as values of f only at points which are not of this form. An explicit example to this effect is provided by a machine computing the function f given by:

$$f(w) = \begin{cases} u^2 0, & w = u0,\ u \in \{0,1\}^*, \\ \sigma_1\sigma_2 \ldots \sigma_{\lfloor n/2 \rfloor}, & w = u1,\ u = \sigma_1\sigma_2 \ldots \sigma_n \in \{0,1\}^*, \\ \varepsilon, & w = \varepsilon. \end{cases}$$

Thus, only (g) is true.

5. The information regarding M_1 is not enough to know what M_1^2 will compute (if at all). If we knew that when M_1 halts the head is pointed to the leftmost square on the tape, it would follow immediately that M_1^2 indeed computes g_{12}. However, if (for example) M_1 starts with checking whether the letter on the tape is B and hangs in that case, and in all other cases halts with the head at the first blank square, then M_1^2 does not halt at all.

   The machine M_2 changes all input letters to 1, changes the first blank square to 0 and then halts. Hence for each k it computes the function from $\mathbf{Z}_+^k$ into $\mathbf{Z}_+$ taking $(n_1, n_2, \ldots, n_k)$ to $\sum_{i=1}^k (n_i + 1) = \sum_{i=1}^k n_i + k$. In particular, $g_2(33, 42, 35) = 33 + 42 + 35 + 3 = 113$.

The machine M_3 erases the first letter and then moves the head right until it gets to the first blank square, looping periodically in the meanwhile between the three T_R machines at the top part of the diagram. Then it erases the whole input. However, if it leaves the top row through one of the first two T_R machines it just writes 0 at the first square of the tape, whereas in the remaining cases it writes 110 on the tape. Now the first row of the diagram is exited at the third T_R machine if and only if the length of the input is divisible by 3, that is the input itself is 2 modulo 3. Hence

$$g_3(n) = \begin{cases} 0, & n \equiv 0 \pmod 3 \ \vee n \equiv 1 \pmod 3, \\ 2, & n \equiv 2 \pmod 3). \end{cases}$$

In particular, $g_3(37) = 0$.

Suppose M_4 halts if the first square is marked with 1, but writes 10 on the tape otherwise. Then it computes the function $g_4 : \mathbf{Z}_+ \longrightarrow \mathbf{Z}_+$ given by

$$g_4(n) = \begin{cases} n, & n \geq 1, \\ 1, & n = 0. \end{cases}$$

The machine M_2 computes both a function from $\{0,1\}^*$ to $\{0,1\}^*$ and a function from $\mathbf{Z}_+$ to $\mathbf{Z}_+$, yet it does not compute a function from $\mathbf{Z}_+^2$ to $\mathbf{Z}_+^2$.

Thus, only (b) is true.