THEORY OF COMPUTATION
CSCI 320, course # 64371
CSCI 320, course # 64372
Test Solution # 2
December 14, 2015
instructor: Bojana Obrenić

Problem 1 Let:

$$L = \{c^{n+2}a^m b^j a^p d^{\ell+1} \mid n = m + j \wedge p = \ell\}$$

where $m, n, j, p, \ell \in N$.

(a) Write a complete formal definition of a context-free grammar that generates L. If such a grammar does not exist, prove it.

Answer: The general template for the strings of L is:

$$cc\, c^j\, c^m\, a^m\, b^j\, a^p\, d^p\, d$$

whence the grammar: $G = (V, \Sigma, P, S)$, where $\Sigma = \{a, b, c, d\}$, $V = \{S, A, B, D\}$, and the production set P is:

$$\begin{aligned} &S \to ccABd \\ &A \to cAb \mid D \\ &D \to cDa \mid \lambda \\ &B \to aBd \mid \lambda \end{aligned}$$

(b) Construct a state-transition graph of a finite automaton that accepts L. If such an automaton does not exist, prove it.

Answer: This automaton does not exist, since L is not regular.

To prove this, assume the opposite—that L is regular. Observe that every word of L satisfies the following property:

> the number of c's is greater by 2 than the number of b's plus the number of a's between c's and b's.

Let k be the constant as in the Pumping Lemma for L. Consider a word $w = c^{2+j+m}\, a^m\, b^j\, a^p\, d^{p+1}$, where $j > k$.

Let $w = uvx$ be a decomposition of w, where v is the pumping part. By the Lemma, it must be that $|uv| \le k < j$. Hence, the pumping part v is entirely within the c-segment, meaning that $v = c^\ell$ for some ℓ such that $0 < \ell \le k$.

After pumping once (up) we obtain a word: $w_1 = c^{2+j+m+\ell}\, a^m\, b^j\, a^p\, d^{p+1}$, where there are $2 + j + m + \ell$ c's, but still only $j + m$ b's and relevant a's. This means that w_1 violates the stated property of L and $w_1 \notin L$, whence the contradiction.

(c) Is L decidable? Prove your answer.

Answer: Yes. L is context free (as is demonstrated in the answer to part (a)), and every context language is decidable.

(d) Is L recursively enumerable? Prove your answer.

Answer: Yes. L is context free, and every context language is recursively enumerable.

Problem 2 Let:

$$L = \{c^{n+2}a^m b^j a^p d^{\ell+1} \mid n = j + p \wedge m = 0\}$$

where $m, n, j, p, \ell \in N$.

(a) Write a complete formal definition of a context-free grammar that generates L. If such a grammar does not exist, prove it.

Answer: The general template for the strings of L is:

$$c^p\, c^j\, cc\, b^j\, a^p\, d^{\ell+1}$$

whence the grammar: $G = (V, \Sigma, P, S)$, where $\Sigma = \{a, b, c, d\}$, $V = \{S, A, B, D\}$, and the production set P is:

$$\begin{aligned} &S \to AD \\ &A \to cAa \mid B \\ &B \to cBb \mid cc \\ &D \to dD \mid d \end{aligned}$$

(b) Construct a state-transition graph of a finite automaton that accepts L. If such an automaton does not exist, prove it.

Answer: This automaton does not exist, since L is not regular.

To prove this, assume the opposite—that L is regular. Observe that every word of L satisfies the following property:

> the number of c's is greater by 2 than the number of b's plus the number of a's.

Let k be the constant as in the Pumping Lemma for L. Consider a word $w = c^{2+p+j}\, b^j\, a^p\, d^{\ell+1}$, where $j > k$.

Let $w = uvx$ be a decomposition of w, where v is the pumping part. By the Lemma, it must be that $|uv| \le k < j$. Hence, the pumping part v is entirely within the c-segment, meaning that $v = c^m$ for some m such that $0 < m \le k$.

After pumping once (up) we obtain a word: $w_1 = c^{2+p+j+m}\, b^j\, a^p\, d^{\ell+1}$, where there are $2 + p + j + m$ c's, but still only $p + j$ b's and a's. This means that w_1 violates the stated property of L and $w_1 \notin L$, whence the contradiction.

(c) Is L recursively enumerable? Prove your answer.

Answer: Yes. L is context free (as is demonstrated in the answer to part (a)), and every context language is recursively enumerable.

(d) Is L decidable? Prove your answer.

Answer: Yes. L is context free, and every context language is decidable.

Problem 3 Let:

$$L = \{c^{n+2}a^m b^j a^p d^{\ell+1} \mid j = n \land m = p\}$$

where $m, n, j, p, \ell \in N$.

(a) Write a complete formal definition of a context-free grammar that generates L. If such a grammar does not exist, prove it.

Answer: Such a grammar does not exist, since L is not context-free. To prove that L is not context-free, recall that

$$L = \{c^{n+2}a^m b^n a^m d^{\ell+1} \mid m, n, \ell \geq 0\}$$

and consider the following two languages:

$$L_1 = \{c^{n+2}a^m b^n a^m d \mid m, n \geq 0\}$$

$$R = \boldsymbol{c^* \, a^* \, b^* \, a^* \, d}$$

Observe that R is regular, but also:

$$L_1 = L \cap R$$

If L was context free then its intersection with any regular language would be context-free. L_1 would be context-free, since it is the intersection of L with the regular language R. To prove that L is not context-free, it is sufficient to prove that L_1 is not context-free. Observe that every string in the language L_1 satisfies all the following properties:

1. number of c's is by two greater than the number of b's;
2. number of a's in the first a-segment is equal to the number a's in the second a-segment;
3. number of d's is equal to one.

Assume that L_1 is context-free, and let η be the constant as in the Pumping Lemma for L. Select a word:

$$w = c^{n+2}a^m b^n a^m d \text{ where } n, m > \eta$$

The word w is long enough and must pump. In any pumping decomposition: $w = xy_1ty_2z$, the length of the pumping window y_1ty_2 is not greater than η. Additionally, the substring d must remain outside any pumping window (or else pumping would produce more than one d's, violating the third property.) Hence, the pumping window is located either entirely within one of the four substrings c^{n+2}, a^m, b^n, a^m, or it is shared by two adjacent ones, since it is too short to span more than two of these substrings. If the pumping window is within any one of the four named substrings, the pumping produces an excess of the letter pumped; if the window is shared by two substrings, the pumping produces a lack of a third letter, thereby in all cases leading to a word that violates at least one of the first two properties. Hence, L_1 violates the Pumping Lemma and thereby cannot be context-free, meaning that L cannot be context-free either.

(b) Write a complete formal definition of a context-free grammar that generates $L \cap \boldsymbol{c^* \, a^* \, d^*}$. If such a grammar does not exist, prove it.

Answer: The template for elements of $L \cap \boldsymbol{c^* \, a^* \, d^*}$ is:

$$L = \{c^2a^{2m}d^{\ell+1} \mid m, \ell \geq 0\}$$

whence the grammar: $G = (V, \Sigma, P, S)$, where $\Sigma = \{a, c, d\}$, $V = \{S, A, D\}$, and the production set P is:

$$\begin{aligned} S &\to ccAD \\ A &\to aaA \mid \lambda \\ D &\to dD \mid d \end{aligned}$$

Problem 4 Let L be the set of those strings over the alphabet $\{a, b, c\}$ such that all c's come before any b's, all b's come before any a's, the number of a's is equal to $n+1$, the number of b's is equal to $n+2$, and the number of c's is equal to $n+3$ for some natural number n.

(a) Write a complete formal definition of a context-free grammar that generates L. If such a grammar does not exist, prove it.

Answer: Such a grammar does not exist, since L is not context-free. To prove this, we show that Pumping Lemma does not hold for L.

Observe that the general template for strings of L is:

$$c^{n+3} \, b^{n+2} \, a^{n+1}$$

Assume the opposite, that L is context-free. Let k be the constant as in the Pumping Lemma for L. Select $n > k$ and consider a word $w = c^{n+3} \, b^{n+2} \, a^{n+1} \in L$, which must pump. In any pumping decomposition, the pumping window is entirely within one of the following three segments: $c^{n+3}, b^{n+2}, a^{n+1}$ or it spans two adjacent segments, since the pumping window is too short to extend into more than two of them, because its length is less than k, and thereby less than n. Thus, if we pump up once, there will be at least one of the segments $c^{n+3}, b^{n+2}, a^{n+1}$ where pumping occurs and at least one where it does not occur. The number of occurrences of at least one of the letters c, b, a will have increased, while the number of occurrences of at least one other of these letters will not have changed. The new word will violate the template and thus will not belong to L. Since L violates the Pumping Lemma, L is not context free.

(b) Construct a state-transition graph of a finite automaton that accepts L. If such an automaton does not exist, prove it.

Answer: This automaton does not exist, since L is not regular. If L was regular, then L would be content free, because every regular language is content free. However, we have proved that L is not context free, hence L cannot be regular.

(c) State the cardinality of L. If L is finite, state the exact number; if L is infinite, specify whether it is countable or not countable.

Answer: L is infinite and countable.

(d) State the cardinality of L^*. If L^* is finite, state the exact number; if L^* is infinite, specify whether it is countable or not countable.

Answer: L^* is infinite and countable.

Problem 5 Let L be the set of those strings over the alphabet $\{a, b, c\}$ such that all c's come before any b's, all b's come before any a's, the number of a's is at least 1, the number of b's is at least 2, and the number of c's is at least 3.

(a) Write a complete formal definition of a context-free grammar that generates L. If such a grammar does not exist, prove it.

Answer: $G = (V, \Sigma, P, S)$, where $\Sigma = \{a, b, c\}$, $V = \{S, A, B, K\}$, and the production set P is:

$$\begin{aligned} &S \rightarrow KBA \\ &A \rightarrow aA \mid a \\ &B \rightarrow bB \mid bb \\ &K \rightarrow cK \mid ccc \end{aligned}$$

(b) Construct a state-transition graph of a finite automaton that accepts L. If such an automaton does not exist, prove it.

Answer: See Figure 1.

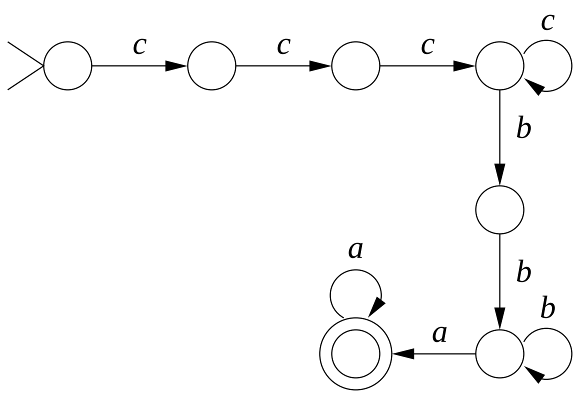

Figure 1:

(c) State the cardinality of L. If L is finite, state the exact number; if L is infinite, specify whether it is countable or not countable.

Answer: L is infinite and countable.

(d) State the cardinality of $\mathcal{P}(L)$ (the set of subsets of L.) If $\mathcal{P}(L)$ is finite, state the exact number; if $\mathcal{P}(L)$ is infinite, specify whether it is countable or not countable.

Answer: $\mathcal{P}(L)$ is infinite and uncountable.

Problem 6 Let:

$$L = \{c^{n+2}a^m c^j b^p d^{\ell+1} \mid j = n \wedge m = 0\}$$

where $m, n, j, p, \ell \in N$.

(a) Write a complete formal definition of a context-free grammar that generates L. If such a grammar does not exist, prove it.

Answer: The general template for strings of L is:

$$c^{2n+2}b^p d^{\ell+1} \mid n, p, \ell \geq 0\}$$

whence the grammar: $G = (V, \Sigma, P, S)$, where $\Sigma = \{b, c, d\}$, $V = \{S, B, K, D\}$, and the production set P is:

$$\begin{aligned} &S \rightarrow KBD \\ &K \rightarrow ccK \mid cc \\ &B \rightarrow bB \mid \lambda \\ &D \rightarrow dD \mid d \end{aligned}$$

(b) Construct a state-transition graph of a finite automaton that accepts L. If such an automaton does not exist, prove it.

Answer: See Figure 2.

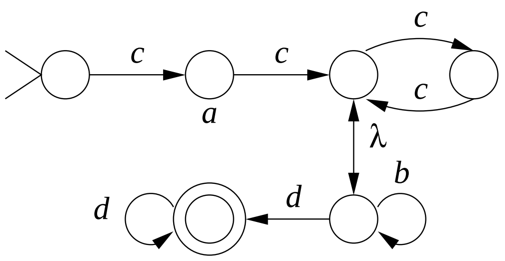

Figure 2:

(c) Is $\overline{L}$ (the complement of L) decidable? Prove your answer.

Answer: Yes. L is context free (as is demonstrated in the answer to part (a)) and every context free language is decidable.

(d) Is $\overline{L}$ (the complement of L) context free? Prove your answer.

Answer: Yes. L is regular (as is demonstrated in the answer to part (b)) and every regular language has a regular complement, which in turn is context free because every regular language is context free.

Problem 7 Let L be the set of those strings over the alphabet $\{a, b, c\}$ that contain exactly two b's.

Let L_1 be the set of those strings over the alphabet $\{a, b, c\}$ that have an odd length.

(a) Construct a state-transition graph of a finite automaton that accepts L. If such an automaton does not exist, prove it.

Answer: See Figure 3.

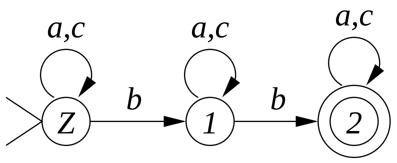

Figure 3:

(b) Construct a state-transition graph of a finite automaton that accepts $L \cap L_1$. If such an automaton does not exist, prove it.

Answer: See Figure 4.

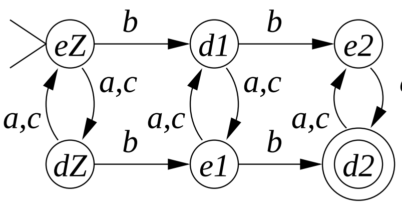

Figure 4:

(c) Explain how to construct an algorithm that solves the following problem:

INPUT:
two finite automata: M and M_1, and a string x.

OUTPUT:
yes if x is a Turing Machine that accepts the intersection of languages defined by $L(M)$ and $L(M_1)$;
no otherwise.

If this algorithm does not exist, prove it.

Answer: This algorithm does not exist. If it existed, it would decide whether the argument Turing Machine x accepts a language with the non-trivial property "is equal to $L(M) \cap L(M_1)$". This property is non-trivial because the language $L(M) \cap L(M_1)$ has this property (as it indeed is equal to itself) while its complement $\overline{L(M) \cap L(M_1)}$ does not have this property (because it is not equal to $L(M) \cap L(M_1)$.) By Rice's Theorem, such a decision is impossible.

Problem 8 Let L be the set of those strings over the alphabet $\{a, b, c\}$ that contain exactly one b.

Let L_1 be the set of those strings over the alphabet $\{a, b, c\}$ whose length is divisible by 3.

(a) Construct a state-transition graph of a finite automaton that accepts L. If such an automaton does not exist, prove it.

Answer: See Figure 5.

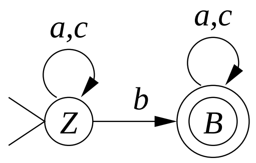

Figure 5:

(b) Construct a state-transition graph of a finite automaton that accepts $L \cap L_1$. If such an automaton does not exist, prove it.

Answer: See Figure 6.

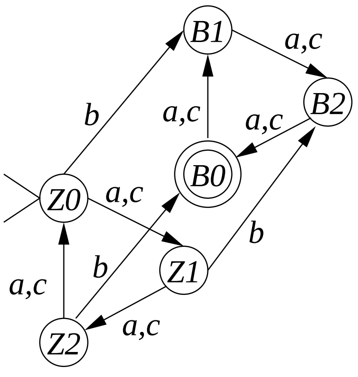

Figure 6:

(c) Explain how to construct an algorithm that solves the following problem:

INPUT:
two finite automata: M and M_1, and a string x.

OUTPUT:
yes if x is a string that belongs to the intersection of languages $L(M)$ and $L(M_1)$;
no otherwise.

If this algorithm does not exist, prove it.

Answer: The algorithm simulates the two argument automata M and M_1 on the argument string x, and returns **yes** if and only if both of these two simulations return **yes**.

Problem 9 Let L be the language accepted by the pushdown automaton: $M = (Q, \Sigma, \Gamma, \delta, q, F)$ where: $Q = \{q, t\}, \Sigma = \{a, b, c\}, \Gamma = \{B, Z\}, F = \{t\}$ and the transition function δ is defined as follows:

$$\begin{array}{l}
[q, a, \lambda, q, ZBB] \\
[q, \lambda, \lambda, t, \lambda] \\
[t, b, B, t, \lambda] \\
[t, c, Z, t, \lambda]
\end{array}$$

(Recall that M is defined so as to accept by final state and empty stack. Furthermore, if an arbitrary stack string, say $X_1 \ldots X_n \in \Gamma^*$ where $n \geq 2$, is pushed on the stack by an individual transition, then the leftmost symbol X_1 is pushed first, while the rightmost symbol X_n is pushed last.)

(a) Write a complete formal definition of a context-free grammar that generates L. If such a grammar does not exist, prove it.

Answer: $G = (V, \Sigma, P, S)$, where $\Sigma = \{a, b, c\}$, $V = \{S\}$, and the production set P is:

$$S \to aSbbc \mid \lambda$$

(b) Explain how to construct an algorithm that solves the following problem:

Input: A Turing Machine T.

Output: A push-down automaton P, such that

$$L(P) = L(T)$$

If this algorithm does not exist, prove it.

Answer: This algorithm does not exist. A language accepted by a Turing Machine need not be context free (and thereby accepted by some pushdown automaton), and it is undecidable whether it is context free.

(c) Explain how to construct an algorithm that solves the following problem:

Input: A non-deterministic push-down automaton P.

Output: A Turing Machine T such that

$$L(T) = L(P)$$

If this algorithm does not exist, prove it.

Answer: The output Turing Machine T simulates the argument pushdown automaton P and decides exactly as P does.

Problem 10 Let L be the language accepted by the pushdown automaton: $M = (Q, \Sigma, \Gamma, \delta, q, F)$ where: $Q = \{q, s, t\}, \Sigma = \{a, b, c, d\}, \Gamma = \{B, K, Z\}, F = \{q\}$ and the transition function δ is defined as follows:

$$\begin{array}{l}
[q, a, \lambda, s, ZKK] \\
[q, a, \lambda, t, ZBB] \\
[t, b, B, t, \lambda] \\
[t, d, Z, q, \lambda] \\
[s, c, K, s, \lambda] \\
[s, d, Z, q, \lambda]
\end{array}$$

(Recall that M is defined so as to accept by final state and empty stack. Furthermore, if an arbitrary stack string, say $X_1 \ldots X_n \in \Gamma^*$ where $n \geq 2$, is pushed on the stack by an individual transition, then the leftmost symbol X_1 is pushed first, while the rightmost symbol X_n is pushed last.)

(a) Write a complete formal definition of a context-free grammar that generates L. If such a grammar does not exist, prove it.

Answer: $G = (V, \Sigma, P, S)$, where $\Sigma = \{a, b, c, d\}$, $V = \{S\}$, and the production set P is:

$$S \to \lambda \mid SS \mid accd \mid abbd$$

(c) Is the automaton M deterministic? Explain your answer.

Answer: No. In state q, while looking at input symbol a, regardless of the stack content, the automaton is able to perform any one of the first two transitions.

(d) Is $L(M)$ decidable? Explain your answer.

Answer: Yes. L is accepted by the pushdown automaton M, and thereby decidable.

(e) Is $L(M)$ accepted by any deterministic finite automaton? Explain your answer.

Answer: Yes. By inspection of the grammar constructed in the answer to part (a), we conclude that L is regular, hence accepted by some deterministic finite automaton.

(f) Is $\overline{L(M)}$ (the complement of $L(M)$) context free? Explain your answer.

Answer: Yes. As is explained in the answer to part (e), L is regular, and thus has a regular complement, which in turn is context free, because every regular language is context free.

Problem 11 Let L be the language accepted by the pushdown automaton: $M = (Q, \Sigma, \Gamma, \delta, q, F)$ where: $Q = \{q, t\}, \Sigma = \{a, b, c\}, \Gamma = \{B, K, Z\}, F = \{q\}$ and the transition function δ is defined as follows:

$$\begin{array}{l}[q, a, \lambda, t, ZKK]\\ [q, a, \lambda, t, ZBB]\\ [t, b, B, t, \lambda]\\ [t, c, K, t, \lambda]\\ [t, \lambda, Z, q, \lambda]\end{array}$$

(Recall that M is defined so as to accept by final state and empty stack. Furthermore, if an arbitrary stack string, say $X_1 \dots X_n \in \Gamma^*$ where $n \geq 2$, is pushed on the stack by an individual transition, then the leftmost symbol X_1 is pushed first, while the rightmost symbol X_n is pushed last.)

(a) Write a complete formal definition of a context-free grammar that generates L. If such a grammar does not exist, prove it.

Answer: $G = (V, \Sigma, P, S)$, where $\Sigma = \{a, b, c\}$, $V = \{S\}$, and the production set P is:

$$S \to \lambda \mid SS \mid acc \mid abb$$

(b) Explain how to construct an algorithm that solves the following problem:

INPUT: A non-deterministic push-down automaton P and a string x.

OUTPUT: **yes** if $x \in L(P)$;
no otherwise.

If this algorithm does not exist, prove it.

Answer: This algorithm simulates P on input x and decides exactly as P decides.

(c) Explain how to construct an algorithm that solves the following problem:

INPUT: A non-deterministic push-down automaton P and a string x.

OUTPUT: **yes** if x is a Turing Machine such that

$$L(x) = L(P)$$

no otherwise.

If this algorithm does not exist, prove it.

Answer: This algorithm does not exist. If it existed, it would decide whether the argument Turing Machine x accepts a language with the non-trivial property "is equal to $L(P)$". This property is non-trivial because the language $L(P)$ has this property (as it indeed is equal to itself) while its complement $\overline{L(P)}$ does not have this property (because it is not equal to $L(P)$.) By Rice's Theorem, such a decision is impossible.

Problem 12 Let L be the language accepted by the pushdown automaton: $M = (Q, \Sigma, \Gamma, \delta, q, F)$ where: $Q = \{q, t\}, \Sigma = \{a, b, c, d\}, \Gamma = \{B, K, Z\}, F = \{t\}$ and the transition function δ is defined as follows:

$$\begin{array}{l}[q, a, \lambda, q, ZK]\\ [q, a, \lambda, q, ZB]\\ [q, \lambda, \lambda, t, \lambda]\\ [t, b, B, t, \lambda]\\ [t, c, K, t, \lambda]\\ [t, d, Z, t, \lambda]\end{array}$$

(Recall that M is defined so as to accept by final state and empty stack. Furthermore, if an arbitrary stack string, say $X_1 \dots X_n \in \Gamma^*$ where $n \geq 2$, is pushed on the stack by an individual transition, then the leftmost symbol X_1 is pushed first, while the rightmost symbol X_n is pushed last.)

(a) Write a complete formal definition of a context-free grammar that generates L. If such a grammar does not exist, prove it.

Answer: $G = (V, \Sigma, P, S)$, where $\Sigma = \{a, b, c, d\}$, $V = \{S\}$, and the production set P is:

$$S \to aScd \mid aSbd \mid \lambda$$

(b) Explain how to construct an algorithm that solves the following problem:

INPUT: A context-free grammar G.

OUTPUT: A context-free grammar G_1 such that

$$L(G_1) = \overline{L(G)}$$

If this algorithm does not exist, prove it.

Answer: Impossible. The complement of the language given by the argument context free grammar need not be context free and it is undecidable whether it is context free.

(c) Explain how to construct an algorithm that solves the following problem:

INPUT: A context-free grammar G.

OUTPUT: A Turing Machine T such that

$$L(T) = \overline{L(G)}$$

If this algorithm does not exist, prove it.

Answer: The output Turing machine converts the argument grammar G into an equivalent pushdown automaton P, simulates P, and decides exactly as P does.

Problem 13 Let L be the language accepted (by final state) by the Turing Machine: $M = (Q, \Sigma, \Gamma, \delta, q, F)$ such that: $Q = \{q, r, s, t, v, x, y\}$; $\Sigma = \{0, 1\}$;
$\Gamma = \{B, 0, 1, Z\}$; $F = \{t\}$; and δ is defined by the following transition set:

$$\begin{array}{l} [q, 0, r, Z, R] \\ [q, 1, r, Z, R] \\ [r, 0, s, Z, R] \\ [r, 1, s, Z, R] \\ [s, 0, v, Z, R] \\ [s, 1, v, Z, R] \\ [v, 0, v, 0, R] \\ [v, 1, v, 1, R] \\ [v, B, x, B, L] \\ [x, 1, y, B, L] \\ [y, 0, t, 0, R] \end{array}$$

(where B is the designated blank symbol.)

(a) Write a regular expression that defines L. If such regular expression does not exist, prove it.

Answer:

$$\mathbf{(0 \cup 1)\,(0 \cup 1)\,(0 \cup 1)\,(0 \cup 1)^*\,01}$$

(b) Let L' be the set of strings which the Turing Machine M rejects. Is L' decidable? Prove your answer.

Answer: L' is decidable. To see this, observe (by inspection of the code) that the Turing Machine M never diverges. Hence, the language L' rejected by M is equal to the complement $\overline{L}$ of the language L accepted by M. As is demonstrated in the answer to part (a), L is regular. Hence, the complement $\overline{L} = L'$ is also regular, and thereby decidable, since every regular language is decidable.

(c) Explain how to construct an algorithm that solves the following problem:

INPUT: A string x over $\{0, 1\}$.

OUTPUT: **yes** if x is a Turing Machine such that $L(x)$ contains at least three strings accepted by M;
no otherwise.

If this algorithm does not exist, prove it.

Answer: This algorithm does not exist. If it existed, it would decide whether the argument Turing Machine x accepts a language with the non-trivial property "contains three strings that belong to L". This property is non-trivial because the language L has this property (as it indeed contains all of its infinitely many strings) while the empty set does not have this property (because it does not contain any strings and in particular does not contain three strings from L.) By Rice's Theorem, such a decision is impossible.

(d) Explain how to construct an algorithm that solves the following problem:

INPUT: A string x over $\{0, 1\}$.

OUTPUT: **yes** if x belongs to L', the language defined in part (b);
no otherwise.

If this algorithm does not exist, prove it.

Answer: The algorithm converts the regular expression proposed in the answer to part (a) into an equivalent finite automaton, simulates that automaton on input x, and decides exactly the opposite.

Problem 14 Let L be the language accepted (by final state) by the Turing Machine: $M = (Q, \Sigma, \Gamma, \delta, q, F)$ such that: $Q = \{q, r, s, t, v, x, y\}$; $\Sigma = \{0, 1\}$;
$\Gamma = \{B, 0, 1, Z\}$; $F = \{t\}$; and δ is defined by the following transition set:

$$\begin{array}{l} [q, 1, r, Z, R] \\ [r, 1, s, Z, R] \\ [s, 0, v, Z, R] \\ [v, 0, v, 0, R] \\ [v, 1, v, 1, R] \\ [v, B, x, B, L] \\ [x, 0, y, B, L] \\ [y, Z, t, B, R] \end{array}$$

(where B is the designated blank symbol.)

(a) Write a regular expression that defines L. If such regular expression does not exist, prove it.

Answer:

1100

(b) Is $\overline{L}$ (the complement of the language L) decidable? Prove your answer.

Answer: Yes. As is demonstrated in the answer to part (a), L is regular, and as such must have a regular complement, which in turn is decidable, because every regular language is decidable.

(c) Explain how to construct an algorithm that solves the following problem:

INPUT: A string x over $\{0, 1\}$.

OUTPUT: **yes** if x belongs to $\overline{L}$ (the complement of the language L);
no otherwise.

If this algorithm does not exist, prove it.

Answer: The algorithm converts the regular expression proposed in the answer to part (a) into an equivalent finite automaton, simulates that automaton on input x, and decides exactly the opposite.

(d) Explain how to construct an algorithm that solves the following problem:

INPUT: A string x over $\{0, 1\}$.

OUTPUT: **yes** if x is a Turing Machine that accepts $\overline{L}$ (the complement of the language L);
no otherwise.

If this algorithm does not exist, prove it.

Answer: This algorithm does not exist. If it existed, it would decide whether the argument Turing Machine x accepts a language with the non-trivial property "is equal to $\overline{L}$". This property is non-trivial because the language $\overline{L}$ has this property (as it indeed is equal to itself) while its complement L does not have this property (because it is not equal to $\overline{L}$.) By Rice's Theorem, such a decision is impossible.