LAST NAME: ______________________________

FIRST NAME: ______________________________

THEORY OF COMPUTATION
CSCI 320, course # 66688
Test Solution # 2
May 11, 2016
instructor: Bojana Obrenić

<u>NOTE:</u> It is the policy of the Computer Science Department to issue a failing grade in the course to any student who either gives or receives help on any test.

Your ability and readiness to follow the test protocol described below is a component of the technical proficiency evaluated by this test. If you violate the test protocol you will thereby indicate that you are not qualified to pass the test.

this is a **closed-book** test, to which it is **forbidden** to bring anything that functions as: paper, calculator, hand-held organizer, computer, telephone, camera, voice or video transmitter, recorder or player, or any device other than pencils (pens), erasers and clocks;

answers should be written only in the space marked "**Answer:** " that follows the statement of the problem (unless stated otherwise);

scratch should never be written in the answer space, but may be written in the enclosed scratch pad, the content of which *will not be graded;*

any problem to which you give **two or more (different) answers** receives the **grade of zero** automatically;

student name has to be written **clearly** on **each page** of the problem set and on the first page of the **scratch pad** the during the **first five minutes of the test**—there is a penalty of **at least 1 point** for each missing name;

when requested, **hand in** the problem set together with the scratch pad;

once you leave the classroom, you cannot come back to the test;

your **handwriting** must be legible, so as to leave no ambiguity whatsoever as to what exactly you have written.

You may work on as many (or as few) problems as you wish.
time: 75 minutes.

each **fully** solved problem: 20 points.
full credit: 100 points.

Good luck.

problem:	01	02	03	04	05	06	07	total: [%]
grade:								

LAST NAME: ____________________

FIRST NAME: ____________________

Problem 1 Let:

$$L = \{a^i c^j a^k b^\ell g^m b^n a^x f^y\}$$

where $i, j, k, \ell, m, n, x, y \geq 0$
are natural numbers such that:

$$i = k,\ n = 3i + 1,\ j = 0,\ \ell = m + 1,\ x = y$$

(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 template for L is:

$$L = \{a^{2k} b^{m+1} g^m b^{3k+1} a^x f^x\}$$

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

$$\begin{aligned}
&S \to AB \\
&A \to aaAbbb \mid Db \\
&D \to bDg \mid b \\
&B \to aBf \mid \lambda
\end{aligned}$$

(b) Draw 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, we show that Pumping Lemma does not hold for L.

Observe that all words of L satisfy the following characteristic property: number of g's is by one less than the number of b's to the left of g's.

Assume the opposite, that L is regular. Let ξ be the constant as in the Pumping Lemma for L. Let $m > \xi$; then the word $w = b^{m+1} g^m b$ belongs to L, as it is obtained from the template by setting $k = x = 0$.

In any "pumping" decomposition such that
$b^{m+1} g^m b = uvx$, we have: $|uv| \leq \xi < m$. Hence, the "pumping" substring v consists entirely of b's, say $v = b^\ell$. Recall that $\ell > 0$, since the "pumping" substring cannot be empty. Pump up once, obtaining the word:

$$w_1 = b^{m+1+\ell} g^m b$$

Since $m + 1 + \ell > m + 1$, word w_1 violates the stated characteristic property and thus $w_1 \notin L$, in violation of the Pumping Lemma.

LAST NAME: ________________________

FIRST NAME: ________________________

Problem 2 Let:

$$L = \{a^i c^j a^k b^\ell g^m b^n a^x f^y\}$$

where $i, j, k, \ell, m, n, x, y \geq 0$
are natural numbers such that:

$$y = i + j, \ \ell = n + 1, \ m = 0, \ k = x$$

(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 template for L is:

$$L = \{a^i c^j a^k b^{2n+1} a^k f^j f^i\}$$

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

$$\begin{aligned} S &\to aSf \mid A \\ A &\to cAf \mid B \\ B &\to aBa \mid D \\ D &\to bbD \mid b \end{aligned}$$

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

Answer: This regular expression does not exist, since L is not regular. To prove this, we show that Pumping Lemma does not hold for L.

Observe that all words of L satisfy the following characteristic property: number of f's is equal to the number of c's plus the number of a's to the left of c's.

Assume the opposite, that L is regular. Let ξ be the constant as in the Pumping Lemma for L. Let $i > \xi$; then the word $w = c^j b f^j$ belongs to L, as it is obtained from the template by setting $i = k = n = 0$.

In any "pumping" decomposition such that
$c^j b f^j = uvx$, we have: $|uv| \leq \xi < i$. Hence, the "pumping" substring v consists entirely of c's, say $v = c^\ell$. Recall that $\ell > 0$, since the "pumping" substring cannot be empty. Pump up once, obtaining the word:

$$w_1 = c^{j+\ell} b f^j$$

Since $j + \ell > j$, word w_1 violates the stated characteristic property and thus $w_1 \notin L$, in violation of the Pumping Lemma.

LAST NAME: ____________________

FIRST NAME: ____________________

Problem 3 Let L be the language accepted by the pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q, F)$ where: $Q = \{q, p\},\ \Sigma = \{a, b, c, d, e, f\}$, $\Gamma = \{A, E, M, X\},\ F = \{p\}$ and the transition function δ is defined as follows:

$$[q, f, \lambda, p, EXAM]$$
$$[p, a, A, p, \lambda]$$
$$[p, b, E, p, \lambda]$$
$$[p, c, M, p, \lambda]$$
$$[p, d, X, p, \lambda]$$
$$[p, e, E, p, \lambda]$$
$$[p, f, \lambda, p, \lambda]$$

(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) List 6 distinct strings that belong to L. If this is impossible, state it and explain why.

Advice for Answer: Regular expression for L:

$$\boldsymbol{f\, f^* c\, f^* a\, f^* d\, f^* (b \cup e)\, f^*}$$

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

Answer: See Figure 1.

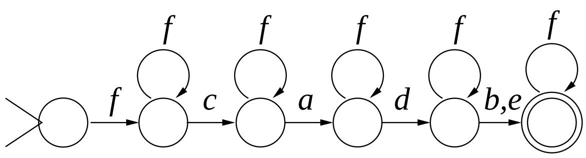

Figure 1:

(c) Is L decidable? Prove your answer.

Answer: Yes. By the answer to part (b), L is regular, and every regular language is decidable.

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

Answer: Yes. As the complement of a regular language, $\overline{L}$ is regular, and thus (decidable) and recursively enumerable.

(e) 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.

LAST NAME: ______________________________

FIRST NAME: ______________________________

Problem 4 Let L be the language accepted by the pushdown automaton $M = (Q, \Sigma, \Gamma, \delta, q, F)$ where: $Q = \{q, p\}$, $\Sigma = \{a, b, c, d, e, f\}$, $\Gamma = \{A, E, M, X\}$, $F = \{p\}$ and the transition function δ is defined as follows:

$$
\begin{array}{l}
[q, f, \lambda, q, EX] \\
[q, e, \lambda, q, AM] \\
[p, a, A, p, \lambda] \\
[p, b, E, p, \lambda] \\
[p, c, M, p, \lambda] \\
[p, d, X, p, \lambda] \\
[q, \lambda, \lambda, p, \lambda]
\end{array}
$$

(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) List 6 distinct strings that belong to L. If this is impossible, state it and explain why.

Answer:

$$\lambda, fdb, eca, fecadb, fefcadbca, fffdbdbdb$$

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

$$S \to fSdb \mid eSca \mid \lambda$$

(c) State one non-trivial property of recursively enumerable languages that is true for L and true for $\emptyset$. Explain carefully why this property is non-trivial, and prove that it is true for L and true for $\emptyset$. If such a property does not exist, state it, and explain why it is so.

Answer:

is context free

The property is non-trivial, since some recursively enumerable languages are context-free (such as $\emptyset$) and some are not context-free (such as the set of palindromes over Σ.) L is context-free, as is witnessed by the grammar constructed in the answer to part (b). ($\emptyset$ is context-free because it is finite.)

(d) State one non-trivial property of recursively enumerable languages that is true for L but false for $\emptyset$. Explain carefully why this property is non-trivial, and prove that it is true for L and false for $\emptyset$. If such a property does not exist, state it, and explain why it is so.

Answer:

is not empty

The property is non-trivial, since some recursively enumerable languages are not empty (such as L) and some are empty (such as $\emptyset$.) L is not empty, as is witnessed by the strings listed in the answer to part (a).

(e) State one trivial property of recursively enumerable languages that is false for L but true for $\emptyset$. Explain carefully why this property is trivial, and prove that it is false for L and true for $\emptyset$. If such a property does not exist, state it, and explain why it is so.

Answer: This property does not exist. By definition, a trivial property assumes the same value for every recursively enumerable language, and if it is true for L it cannot be false for $\emptyset$.

LAST NAME: ______________________________

FIRST NAME: ______________________________

Problem 5 Consider the Turing machine $M = (Q, \Sigma, \Gamma, \delta, q, F)$ such that: $Q = \{q, p, v, z, x\}$; $\Sigma = \{0, 1\}$; $\Gamma = \{B, 0, 1\}$; $F = \{x\}$; and δ is defined by the following transition set:

$$\begin{array}{lll} [q, 0, q, 0, R] & [p, 1, q, 1, R] & [v, 0, z, 0, R] \\ [q, 1, p, 1, R] & [p, 0, p, 0, R] & [v, 1, x, 1, R] \\ [q, B, q, B, R] & [p, B, v, B, L] & \end{array}$$

(M has an one-way infinite tape (infinite to the right only.) B is the designated blank symbol. M accepts by final state.)

Let L_A be the set of string which M *accepts.*
Let L_R be the set of string which M *rejects.*
Let L_D be the set of string on which M *diverges.*

(a) List 6 distinct strings that belong to L_A. If this is impossible, state it and explain why.

Advice for Answer: M skips over the input to the right, flipping states to remember the parity of the number of 1's seen. If this number is even, M diverges. If the number of 1's is odd, M accepts or rejects according to the rightmost input symbol.

See the answer to part (d).

(b) List 6 distinct strings that belong to L_R. If this is impossible, state it and explain why.

Advice for Answer: See the answer to part (e).

(c) List 6 distinct strings that belong to L_D. If this is impossible, state it and explain why.

Advice for Answer: See the answer to part (f).

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

Answer:

0* (10*1) 0*1

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

Answer:

0* (10*1) 0*10*0

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

Answer:

0* (10*1) 0*

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

INPUT: String w over $\{0, 1\}$.

OUTPUT: **yes** if w is a string such that the Turing Machine M (defined at the beginning of this problem) rejects w;
no otherwise.

If this algorithm does not exist, prove it.

Answer: This algorithm decides whether the given input string belongs to the language L_R, whose regular expression is given in the answer to part (e). Therefore, convert this regular expression into a finite automaton, convert this automaton into a deterministic finite automaton, simulate this deterministic automaton and decided as it decides.

LAST NAME: ______________________________

FIRST NAME: ______________________________

Problem 6 Consider the Turing machine $M = (Q, \Sigma, \Gamma, \delta, q, F)$ such that: $Q = \{q, p, v, z, x\}$; $\Sigma = \{0, 1\}$; $\Gamma = \{B, 0, 1, N\}$; $F = \{x\}$; and δ is defined by the following transition set:

$$\begin{array}{lll} [q,0,p,N,R] & [p,0,p,0,R] & [v,1,v,1,L] \\ [q,1,q,1,R] & [p,1,p,1,R] & [v,0,x,0,R] \\ [q,B,q,B,R] & [p,B,v,B,L] & [v,N,z,0,R] \end{array}$$

(M has an one-way infinite tape (infinite to the right only.) B is the designated blank symbol. M accepts by final state.)

Let L_A be the set of string which M *accepts.*
Let L_R be the set of string which M *rejects.*
Let L_D be the set of string on which M *diverges.*

(a) List 6 distinct strings that belong to L_A. If this is impossible, state it and explain why.

Advice for Answer: M moves to the right, looking for a 0. If M does not find a 0, it diverges. Otherwise, M rewrites the first 0 into N, skips over the input, and on the way back looks for yet another 0 (not rewritten.) M accepts if it finds this second 0 and rejects otherwise.

See the answer to part (d).

(b) List 6 distinct strings that belong to L_R. If this is impossible, state it and explain why.

Advice for Answer: See the answer to part (e).

(c) List 6 distinct strings that belong to L_D. If this is impossible, state it and explain why.

Advice for Answer: See the answer to part (f).

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

Answer:

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

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

Answer:

$$\mathbf{1^*01^*}$$

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

Answer:

$$\mathbf{1^*}$$

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

INPUT: String w over $\{0, 1\}$.

OUTPUT: **yes** if w represents a Turing Machine that accepts exactly those strings which the Turing Machine M (defined at the beginning of this problem) accepts;
no otherwise.

If this algorithm does not exist, prove it.

Answer: This algorithm does not exist. If it existed, it would decide the set of those Turing Machines whose languages have the non-trivial property: "is equal to L_A". By Rice's Theorem, this is impossible. This property is non-trivial because it is true for L_A but false for L_D.

LAST NAME: ______________________

FIRST NAME: ______________________

Problem 7 Consider the Turing machine $M = (Q, \Sigma, \Gamma, \delta, q, F)$ such that: $Q = \{q, r, s, p, v, t, z, x, y\}$; $\Sigma = \{0, 1\}$; $\Gamma = \{B, 0, 1\}$; $F = \{x\}$; and δ is defined by the following transition set:

$$\begin{array}{lll} [q,1,r,1,R] & [p,0,p,0,R] & [y,0,y,0,R] \\ [r,0,s,0,R] & [p,1,p,1,R] & [y,1,y,1,R] \\ [s,1,t,1,R] & [p,B,v,B,L] & [y,B,y,B,R] \\ \\ [t,0,p,0,R] & [v,0,z,0,L] & [z,0,x,0,L] \\ [t,1,p,1,R] & [v,1,x,1,L] & [z,1,y,1,L] \end{array}$$

(M has an one-way infinite tape (infinite to the right only.) B is the designated blank symbol. M accepts by final state.)

Let L_A be the set of string which M *accepts.*
Let L_R be the set of string which M *rejects.*
Let L_D be the set of string on which M *diverges.*

(a) List 6 distinct strings that belong to L_A. If this is impossible, state it and explain why.

Advice for Answer: M halts and rejects immediately unless the input has at least four symbols and begins with the substring 101. If M finds such four symbols, it skips over the input and inspects the rightmost symbols to decide whether to accept or diverge.

See the answer to part (d).

(b) List 6 distinct strings that belong to L_R. If this is impossible, state it and explain why.

Advice for Answer: See the answer to part (e).

(c) List 6 distinct strings that belong to L_D. If this is impossible, state it and explain why.

Advice for Answer: See the answer to part (f).

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

Answer:

$$\mathbf{101\,(0 \cup 1)^*\,(1 \cup 00)}$$

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

Answer:

$$\mathbf{(0 \cup 1)\,(0 \cup 1)\,(0 \cup 1)\ \cup\ (0 \cup 11 \cup 100)\,(0 \cup 1)^*}$$

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

Answer:

$$\mathbf{101\,(0 \cup 1)^*\,10\ \cup 1010}$$

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

INPUT: String w over $\{0, 1\}$.

OUTPUT: **yes** if w represents a Turing Machine that accepts exactly those strings on which the Turing Machine M (defined at the beginning of this problem) halts;
no otherwise.

If this algorithm does not exist, prove it.

Answer: This algorithm does not exist. If it existed, it would decide the set of those Turing Machines whose languages have the non-trivial property: "is equal to $L_A \cup L_R$". By Rice's Theorem, this is impossible. This property is non-trivial because it is true for $L_A \cup L_R$ but false for L_D.