Exercises
CSC 341, “Automata, Formal Languages, and Computational Complexity”
Department of Computer Science - Grinnell College
March 17, 2021

1 (20 points; due February 3). Let \( A \) be any alphabet containing two or more symbols, and let \( a \) be the number of symbols in \( A \). Using mathematical induction, prove that, for every natural number \( n \), the number of strings over the alphabet \( A \) of length \( n \) or less is \( \frac{a^{n+1} - 1}{a - 1} \).

2 (25 points; due February 4). Design and formally specify a finite automaton that recognizes the set of binary numerals (non-empty strings on the alphabet \{0, 1\}, with leading zeroes allowed) that denote multiples of five: \{0, 00, 000, 0000, 00000, 000000, ...\}.
(Hint: What needs to be stored in the automaton’s memory in this case?)

3 (25 points; due February 5). Prove that, for any regular language \( A \), the language \( \{w^R \mid w \in A\} \) is also regular. (Recall that \( w^R \) is the reversal of the string \( w \).)

4 (20 points; due February 8). For each natural number \( k \), find a (different) regular expression describing the set of binary numerals (possibly with leading zeroes) that denote powers of \( 2^k \). (For instance, in the case where \( k = 3 \), the regular expression should describe the set \{1, 01, 001, 100, 0001, 0100, 00001, 00100, 000001, 000100, 1000000, ...\} of binary numerals for powers of eight (\( 2^3 \)).)

5 (20 points; due February 9). Let \( K \) be the language \{\( \varepsilon, a, aab, aababb, aababbabb, aababbabbabbb, \ldots \} \) of strings in which the \( i \)th occurrence of \( a \) is followed by exactly \( i - 1 \) occurrences of \( b \) (for every positive integer \( i \) less than or equal to some natural number \( n \)). Using the Pumping Lemma, prove that \( K \) is not regular.

6 (20 points; due February 10). (a) Find the largest possible set of strings on the alphabet \{c, d\} that is pairwise distinguishable by the language \( L \) of strings not containing the substring \( cdd \). Determine the index of \( L \).
(b) Formally define a deterministic finite automaton that recognizes \( L \) and has a number of states equal to the index of \( L \).

7 (20 points; due February 11). Devise a context-free grammar that generates the language \( \{a^ib^jc^k \mid i > j \text{ or } k > j\} \) and show that your grammar is ambiguous by providing two different leftmost derivations for the same string.

8 (20 points; due February 12). Problem 1.54 in the textbook (page 91) shows that the language \( \{a^ib^j c^k \mid i, j, k \geq 0 \text{ and if } i = 1 \text{ then } j = k\} \) is not regular (and so there is no finite automaton that recognizes it). Formally specify a pushdown automaton that recognizes this language.
9 (20 points; due February 15). The context-free grammar presented in Exercise 2.1 of our textbook (page 154) illustrates the construction of arithmetic expressions in many programming languages:

\[
\begin{align*}
E &\rightarrow E + T \\
E &\rightarrow T \\
T &\rightarrow T \times F \\
T &\rightarrow F \\
F &\rightarrow (E) \\
F &\rightarrow a
\end{align*}
\]

Here \( E, T, \) and \( F \) are variables in the grammar and \( a, +, \times, (, \) and \( ) \) are terminals.

Define a pushdown automaton that recognizes the language that this context-free grammar specifies. (You may use the construction method described in the proof of Lemma 2.21 if you like, but you may be able to create a design that is equally good, or better, on your own.)

10 (20 points; due February 16). A unary numeral is a string of 1s that is understood as denoting its length. (So, for example, the unary numeral 11111 denotes the natural number five.) An addition fact for unary numerals is a string on the alphabet \( \{1, +, =\} \) of the form \( 1^i + 1^j = 1^k \) such that \( i + j = k \).

(a) Prove that the language of addition facts for unary numerals is context-free.

(b) Define an analogous language of multiplication facts for unary numerals and prove that it is not context-free.

11 (25 points; due February 17). A regular grammar is a context-free grammar in which the right-hand side of every rule is either (a) exactly one terminal followed by exactly one variable or (b) the empty string. Prove that a language is regular if, and only if, it is specified by some regular grammar. You may use any of the theorems, lemmas, and corollaries from Chapters 1 and 2 of the textbook.

12 (20 points; due February 18). The bitwise complement \( \sim w \) of a string \( w \) on the alphabet \( \{0, 1\} \) is the result of changing every 0 in \( w \) to a 1 and every 1 to a 0. Formally specify a single-tape, deterministic Turing machine that decides the language \( \{w \sim w \mid w \in \{0, 1\}^*\} \).

13 (25 points; due February 19). A cell splitter is similar to an ordinary Turing machine, but its transition function has the form

\[ \delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{R, L, S\}. \]

On each transition \( \delta(q, a) = (r, b, S) \), the machine in effect divides the tape cell that is under the read-write head into two adjacent tape cells and writes the symbol \( b \) into both cells. It then positions the read/write head on the second cell (the one on the right) and enters state \( r \).

Prove, using simulation, that every language that is recognized by a cell splitter is Turing-recognizable.

14 (20 points; due February 22). Sipser uses the term “standard string order” to mean the arrangement of strings in order of increasing length, and then in alphabetical
order among strings of the same length. For example, the standard string order of all strings on the alphabet \{a, b, c\} begins:

\[\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, aab, \ldots\]

Give a proof analogous to Sipser’s proof of Theorem 3.21 (page 181) to show that a language \(L\) is decidable if, and only if, there is an enumerator \(E\) that prints the members of \(L\) in standard string order. (\(E\) should print each member of \(L\) only once and should not print non-members of \(L\) at all.)

15 (20 points; due February 23). (a) If the Church-Turing thesis is correct, could we use a high-level programming language (such as Scheme, C, or Java) to write what Sipser calls “high-level descriptions” of Turing machines? For example, could we write

\[
\text{(lambda} (w)
  \text{(let} ((len (string-length w)))
    \text{(and} (positive? len)
      \text{(let loop} ((rest len))
        \text{(or} (= rest 1)
          \text{(and} (even? rest)
            \text{(loop} (quotient rest 2))))
        \text{(let loop} ((position 0))
          \text{(or} (= position len)
            \text{(and} (char=? (string-ref w position) #\0)
              \text{(loop} (+ position 1)))))))
\]

as part of a proof by construction that language \(A\) from Example 3.7 (page 171) is decidable?

(b) Suggest a way to encode signed integers for processing by Turing machines.

(c) Suggest a way to encode strings of Unicode characters for processing by Turing machines.

16 (20 points; due February 24). Prove that the problem of whether a given deterministic finite automaton \(M\) rejects at least one string of even length is decidable.

17 (20 points; due February 25). A sequence of positive integers is fast-rising if, and only if, each element after the first is strictly greater than the square of the element that precedes it. Fast-rising sequences can be either finite (such as \((3, 10, 101, 10202)\)) or infinite (such as \((1, 2, 8, 128, 32768, \ldots)\)).

Prove by diagonalization that the set of all fast-rising sequences of positive integers is uncountable.

18 (20 points; due February 26). Prove, using a diagonalization argument, that the language \(R_{\text{TM}} = \{\langle M, w \rangle \mid M\text{ is a Turing machine and rejects } w\}\) is undecidable.

19 (20 points; due March 1). A Turing machine is a universal acceptor if it accepts every string on its input alphabet. Prove that the set of all encodings of universal acceptor Turing machines is undecidable.

20 (25 points; due March 2). The Bounded-Prefix variant of the Post Correspondence Problem (BPPCP) is a domino-matching problem similar to PCP, except that a sequence of dominoes counts as a “near-match” to the Bounded-Prefix variant if there is a non-empty string \(w\) and strings \(u\) and \(v\) of length 3 or less such that the string of
symbols across the top of the domino sequence is $uw$ and the string of symbols across the bottom is $vw$.

In other words, to qualify as a near-match, either the top and bottom strings must match exactly ($u = v = \varepsilon$) or they must match if one ignores the first symbol or the first two or three symbols from either or both of the strings ($u$ and $v$ contain the ignored symbols). So some domino collections that have no matches under the rules of the PCP will have near-matches under the Bounded-Prefix rule and so will be elements of $BPPCP$.

Prove that the set of (encodings for) instances of the Bounded-Prefix variant of the Post Correspondence Problem that have near-matches is undecidable.

21 (20 points; due March 3). Design a semi-Thue process in which, for any string $w$ on the alphabet $\{a, b\}$, $w = \ast w \ast$ generates the one-symbol string $+$ if, and only if, $w$ is a palindrome. (The process alphabet will of course include $a$, $b$, $\ast$, and $\ast$. You may add additional symbols as appropriate.)

22 (25 points; due March 4). Let $T$ be the set of strings on the alphabet $\{a, b\}$ that contain an odd number of $a$s. Prove that a language $L$ is decidable if, and only if, $L \leq_m T$.

23 (20 points; due March 5). Using Rice’s Theorem, prove that the language $$\{ \langle M \rangle \mid M \text{ is a Turing machine and every string that } M \text{ accepts is a palindrome} \}$$ is undecidable.

24 (25 points; due March 8). (a) Using the Recursion Theorem, prove that the language $NH_{\text{TM}}$, defined as $$\{ \langle M \rangle \mid M \text{ is a Turing machine that does not halt on any input string } w \}$$ is undecidable.

(b) Explain why this result is not a special case of Rice’s Theorem.

25 (25 points; due March 9). Prove that, for any computable function $t : \Sigma^* \rightarrow \Sigma^*$, there is a natural number $c_t$ such that, for any string $x \in \Sigma^*$, $K(t(x)) \leq K(x) + c_t$. (In other words, applying any computable function to a string cannot increase the descriptive complexity by more than a fixed amount that depends on the function but not on the string to which it is applied.)

26 (25 points; due March 10). Let’s say that a natural number $n$ is arbitrary if (and only if) the binary numeral for $n$ is incompressible.

For this problem, we stipulate that the unique binary numeral for zero is 0 and that the unique binary numeral for any positive integer begins with 1 (no leading zero bits). For example, the one and only binary numeral for 17 is 10001, and 0010111, 00, and $\varepsilon$ are not binary numerals.

(a) Prove that the set $\{ n \mid n \text{ is arbitrary and } n \text{ is a power of } 7 \}$ is finite.

(b) For any binary string $w$, let $^\wedge w$ be the bitwise complement of $w$, that is, the result of changing every 0 in $w$ to 1 and every 1 to 0. Prove that, if $w$ is incompressible, $^\wedge w$ is also incompressible.

(c) Prove that, for every natural number $n$, there is at least one arbitrary number $m$ such that $2^n \leq m < 2^{n+1}$. 
27 (25 points; due March 11). Suppose that language $L$ is decided by some non-deterministic Turing machine with a running-time function $g$ (as specified in Definition 7.9, page 283) such that $g(n) = \log_2 n$. Prove that there is a deterministic Turing machine with a polynomial running-time function that decides $L$.

28 (25 points; due March 12). Prove that, if $L \in \text{P}$, then $L^* \in \text{P}$.

29 (25 points; due March 15). An instance of the PARTITION problem asks whether or not a collection $C$ of natural numbers (a multiset — repetitions allowed — as in the SUBSET-SUM problem) can be partitioned into two subcollections that have the same sum.

For example, if the members of $C$ are 4, 11, 16, 21, 23, and 33, then the answer is yes, because we can put 4, 11, 16, and 23 in one subcollection and 21 and 33 in the other. The sum of the elements in each subcollection is 54. On the other hand, if the members of $C$ are 1, 2, 4, 8, and 31, then the answer is obviously no — whichever subcollection contains 31 will necessarily have a larger sum than the other.

Formally, the PARTITION problem is

$$\{\langle C \rangle \mid C \text{ can be partitioned into two subcollections with equal sums}\}.$$

Prove that SUBSET-SUM $\leq_p$ PARTITION. You may assume that the elements of any candidate for membership in SUBSET-SUM are natural numbers.

30 (25 points; due March 16). Let 3SAT-WITHOUT-NEGATION be the language

$$\{\langle \phi, k \rangle \mid \text{there is an assignment that satisfies } \phi \text{ in which exactly } k \text{ variables receive the value } 1\}.$$

where $\phi$ is a 3cnf formula in which no variable is negated and $k$ is a natural number.

So, for example, if $\phi$ is $(u \lor v \lor w) \land (w \lor x \lor y) \land (v \lor y \lor z)$, then $\langle \phi, 2 \rangle \in 3SAT-WITHOUT-NEGATION$, since $\phi$ is satisfied by the assignment $u = 0, v = 0, w = 1, x = 0, y = 1, z = 0$. But $\langle \phi, 1 \rangle \notin 3SAT-WITHOUT-NEGATION$, since no assignment that assigns 1 to exactly one variable satisfies $\phi$.

Prove that 3SAT $\leq_p$ 3SAT-WITHOUT-NEGATION.

31 (25 points; due March 17). In an undirected graph $G$ with vertices $V$ and edges $E$, let’s say that a subset $C$ of $V$ is a “core” of $G$ if, and only if, every element of $V$ is either an element of $C$ or connected by an edge to an element of $C$. Let CORE be the language

$$\{\langle G, k \rangle \mid G \text{ has a core with exactly } k \text{ elements}\}.$$

Prove that CORE is NP-complete.

32 (20 points; due March 18). The subgraph isomorphism problem asks, for two given graphs $G$ and $H$, whether $G$ contains a subgraph that is isomorphic to $H$. In other words: Is there a way of mapping every vertex of $H$ to a different vertex of $G$ so that there is an edge between two vertices in $H$ if, and only if, there is an edge in $G$ between the vertices to which they are mapped?

Let SIP be the language

$$\{\langle G, H \rangle \mid G \text{ and } H \text{ are graphs and } H \text{ is isomorphic to a subgraph of } G\}.$$

Prove that SIP is NP-complete.
33 (20 points; due March 19). Let $L_1, L_2, \ldots, L_m$ be any languages that are all elements of $\text{SPACE}(n^k)$ for some positive integer $k$. Prove that $L_1 \cup L_2 \cup \ldots \cup L_m$ is also an element of $\text{SPACE}(n^k)$. 