CSC 456/656 Fall 2026 Answers to Second Examination October 29, 2026

In the questions of this test, $\mathcal{P}$ and $\mathcal{NP}$ denote $\mathcal{P}$-TIME and $\mathcal{NP}$-TIME, respectively.
If L is a language over an alphabet Σ, we define the *complement* of L to be the set of all strings over Σ which are not in L. If $\mathcal{C}$ is a class of languages, we define co-$\mathcal{C}$ to be the class of all complements of members of $\mathcal{C}$.

1. True or False. 5 points each. T = true, F = false, and O = open, meaning that the answer is not known science at this time.

   (i) [5 points] **F** The set of languages over the binary alphabet is countable.

   (ii) [5 points] **O** $\mathcal{P} = \mathcal{NC}$.

   (iii) [5 points]**O** The complement of any $\mathcal{NC}$ language is $\mathcal{NC}$.

   (iv) [5 points]**T** Every context-free language is $\mathcal{NC}$.

   (v) [5 points]**O** CVP, the circuit value probelem, is $\mathcal{NC}$.

   (vi) [5 points]**F** LALR parsing can be used for any context-free grammar.

   (vii) [5 points]**F** LALR parsing cannot be used for any ambiguous grammar.

   (viii) [5 points]**F** The context-free grammar equivalence problem is recursively enumerable.

   (ix) [5 points]**T** If L is any recursively enumerable language and $w \in L$, then there is a proof that $w \in L$.

2. [20 points] Give a definition of the class $\mathcal{P}$-complete.

   L is $\mathcal{P}$-complete if
   1. L is $\mathcal{P}$
   2. Every P language can be reduced to L using a reduction which can be computed in polylogarithmic time using polynomially many processors. Equivalently, using a reduction which can be computed using polylogarithmic space.

   Name a language which is $\mathcal{P}$-complete. There are many correct answers, but only one such language has been mentioned in class, and that is the circuit value problem.

3. If the set is countable, enter **C**, if uncountable, enter **U**.

   (i) [5 points] **C** the set of all rational numbers.

   (ii) [5 points] **U** the set of all irrational numbers.

   (iii) [5 points] **C** The set of all finite sets of integers.

   (iv) [5 points] **U** The set of all sets of integers.

   (v) [5 points] **U** The set of all finite sets of real numbers.

   (vi) [5 points] **U** The set of all sets of real numbers.

   (vii) [5 points] **U** The set of all languages over the binary alphabet.

(viii) [5 points] **U** The set of all undecidable languages over the binary alphabet.

(ix) [5 points] **C** the set of all recursive real numbers.

(x) [5 points] **C** The set of all recursively enumerable languages over the binary alphabet.

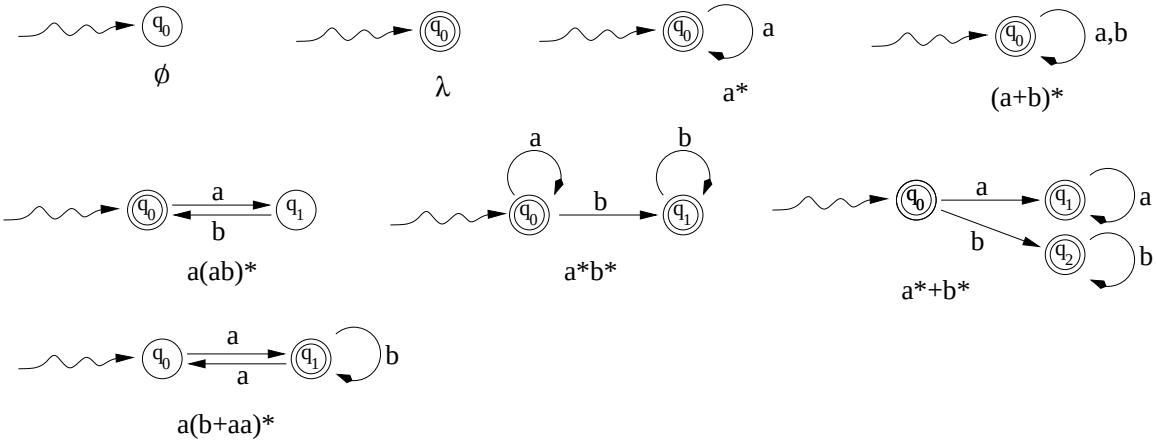
(xi) [5 points] **C** The set of all regular expressions over the binary alphabet.

4. [10 points] Let L be the language of all strings over the alphabet $\{a, b\}$ which cootain the substring aba. Write a regular expression for L.

   There are infinitely many correct answers, but deciding whether an answer is correct could take more than polynomial time. So my rule is, if I can't see that it's correct within the first minute, I'll mark it wrong.

   My answer is $(a+b)^*aba(a+b)^*$. There are other not-so-complex correct answers, but I believe mine is the simplest.

5. [40 points] For each of these regular expressions: $a(b+aa)^*$, $(ab)^*$, a^*b^*, $\emptyset$, λ, $(a+b)^*$, a^*, a^*+b^*, find a corresponding DFA in the figures below.



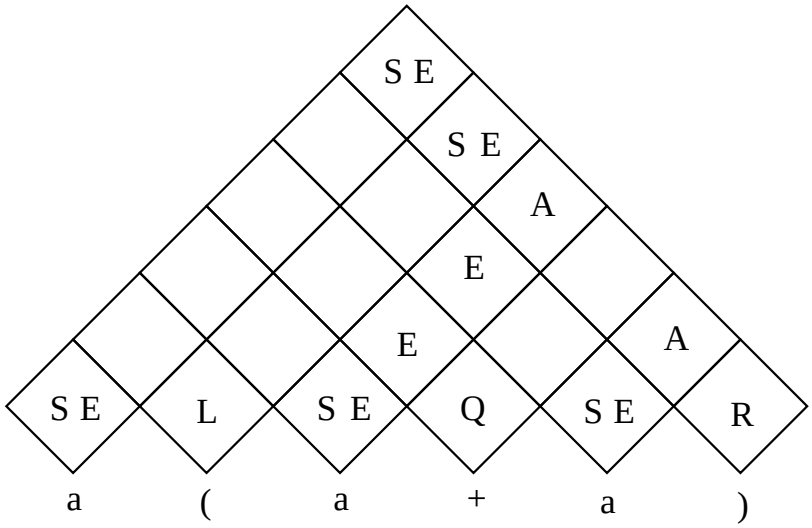
6. [20 points] Give a CNF grammar for the language $L = \{a^n b^n \ : \ n \geq 0\}$.

The usual grammar is: $S \to aSb$
$S \to \lambda$.
and $L(S) = L$.

There are many correct answers to this problem. We introduce a new variable T which generates all of L except the empty string. That is, $L(T) = \{a^n b^n \ : \ n \geq 1\}$. We introduce new variables A and B, such that $L(A) = \{a\}$ and $L(B) = \left\{a^n b^{n+1} \ : \ n \geq 0\right\}$. Our grammar is now $S \to AB$
$T \to AB$
$A \to a$
$B \to TB$
$B \to b$
$S \to \lambda$
Here is a rightmost derivation of $aabb$.

7. [30 points] Let L be the language generated by the Chomsky Normal Form (CNF) grammar given below. Then use the CYK algorithm to verify that $a(a + a) \in L$.

(i) $S \to a$
(ii) $E \to a$
(iii) $S \to LA$
(iv) $E \to LA$
(v) $L \to ($
(vi) $A \to ER$
(vii) $R \to)$
(viii) $S \to PE$
(ix) $E \to PE$
(x) $S \to EE$
(xi) $E \to EE$
(xii) $P \to EQ$
(xiii) $Q \to +$



8. [20 points] Give a polynomial time reduction of 3SAT to the independent set problem.

Let $E = C_1 * C_2 * \cdots * C_k$ be a Boolean expression in 3-CNF form. Each C_i is a clause which is the disjunction of three terms, each of which is either a variable or the negation of a variable. The reduction maps E onto a graph G, which $3k$ vertices, each labeled by one of the $3k$ terms of E. Each clause C_i corresponds to the 3-clique Q_i whose vertices are labeled by the three terms of C_i. The remaining edges of G are between cliques. These edges connect any two vertices of G which have contradictory labels. We show that E is satisfiable if and only if G has an independent set of size k.

Suppose E has a satisfying assignment. Pick one term of each clause which is true under the assignment, we call that the *satisfying* term of that clause. Let I be the set of vertices labeled by the certifying terms, k vertices altogether. Any two members of I cannot be connected by a clique edge, since they lie

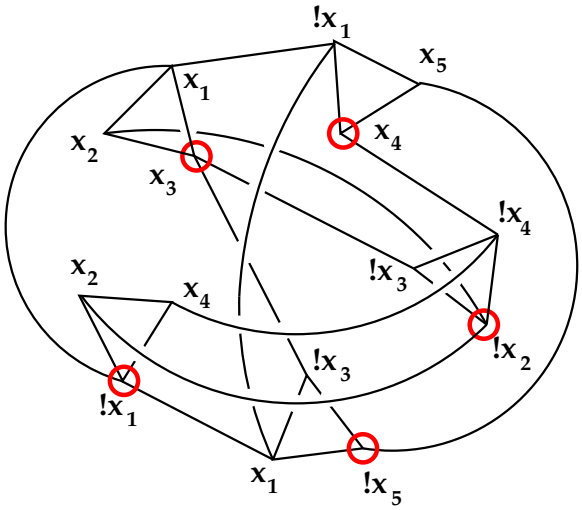
in different cliques. They also cannot be connected by an edge between cliques, since the labels of the ends of that edge cannot both be true under the assignment. Thus G has an independent set of size k.

Conversely, suppose G has an independent set I of size k. There must No two of these vertices can lie in the same clique, hence there must be one in each clique. Let v_0 be the sole member of $I \cap Q_i$. Now pick an assignment such that the label of each member of I is assigned true. This is possible because no two labels of members of I can contradict. If there is a variable whose value has not been assigned, assign that variable the value true. (Actually, that assignment is arbitrary.) This assignment satisfies E, since each clause has a certifying term.

Example. Let $E = (x_1 + x_2 + x_3) * (!x_1 + x_4 + x_5) * (!x_2 + !x_3 + !x_4) * (x_1 + !x_3 + !x_5) * (!x_1 + x_2 + x_4)$

The following assignment satisfies E.
$x_1 = 0$
$x_2 = 0$
$x_3 = 1$
$x_4 = 1$
$x_5 = 0$

In the figure below, the vertices of the corresponding independent set of G are circled in red.



9. [20 points] Prove that every decidable language can be enumerated in canonical order by some machine.

10. [20 points] Prove that every recursively enumerable language is accepted by some machine.

11. [20 points] The halting problem is defined as follows. Given a machine M and a string w, does M halt if its input is w? We will, later in this course, prove that the halting problem is undecidable. **However:**

Here is a "proof" that the halting problem is decidable! Explain why this "proof" is invalid.

Every machine is equivalent to some C++ program, and each such program has finite length. Let $\mathcal{M}$ be the set of all ordered pairs (M, w) such that M is a machine and w is a string. We define the length of such a pair (M, w) to be the length of the C++ program equivalent to M, plus the length of w. There can be only finitely many members of $\mathcal{H}$ of length n. Let $\mathcal{H}$ be the set of pairs (M, w) such that M halts with input w, a subset of $\mathcal{M}$, hence finite.

For any n, let $T(n)$ be the maximum number of steps it takes M to halt with input w for any $(M, w) \in \mathcal{H}$.

Given $(M, w) \in \mathcal{M}$, we can determine whether M halts with input w, the following algorithm gives us the answer.

Let n be the length of (M, w). Run M with input w for $T(n) + 1$ steps or until it halts, whichever comes first. If it halts, the answer is yes, but if it runs $T(n) + 1$ steps, the answer is no. Thus, the halting problem is decidable.

Answer: The error is that the function T, although well-defined, is not computable. Thus, the algorithm given cannot be implemented by any program.