University of Nevada, Las Vegas Computer Science 456/656 Fall 2023

CSC 456/656 Fall 2023 Answers to Second Examination October 25, 2023

A binary function is defined to be a function F on binary strings such that, for each binary string w, $F(w)$ is a binary string. (Of course, the strings could be numerals.)

A real number x is defined to be recursive if there is a machine which runs forever writing the decimal expansion of x.

1. True or False. If the question is currently open, write "O" or "Open."

   (i) **F** Every subset of a regular language is regular.

   Every language is a subset of some regular language.

   (ii) **O** $\mathcal{P} = \mathcal{NP}$.

   (iii) **O** $\mathcal{P} = \mathcal{NC}$.

   (iv) **T** Every regular language is $\mathcal{NC}$.

   (v) **T** Every context-free language is $\mathcal{NC}$.

   (vi) **O** The Boolean circuit problem is $\mathcal{NC}$

   This problem is $\mathcal{P}$–complete, meaning that if it is $\mathcal{NC}$, then $\mathcal{P} = \mathcal{NC}$.

   (vii) **T** The complement of any $\mathcal{P}$–TIME language is $\mathcal{P}$-TIME.

   A machine that decides a $\mathcal{P}$–TIME language also decides its complement.

   (viii) **O** The complement of any $\mathcal{NP}$ language is $\mathcal{NP}$.

   If $\mathcal{P} = \mathcal{NP}$ it is trivially true since the complement of any $\mathcal{P}$ problem is $\mathcal{P}$.

   (ix) **T** The complement of any $\mathcal{P}$–SPACE language is $\mathcal{P}$–SPACE.

   A machine that decides a $\mathcal{P}$–SPACE language also decides its complement.

   (x) **T** The complement of every recursive language is recursive.

   A machine that decides a recursive language also decides its complement.

   (xi) **F** The complement of every recursively enumerable language is recursively enumerable.

   (xii) **T** If a language L is accepted by an NFA M with p states, then p is the pumping length of L.

   If $w \in L$, of length at least p, a computation of M with input w must visit some state twice. A substring of w starting and ending with visits to the same state is a pumpable substring.

(xiii) **T** Given any unambiguous context-free grammar G and any string $w \in L(G)$, there is always a unique leftmost derivation of w using G.

One of the definitions of ambiguity is that some string has more than one leftmost derivations.

(xiv) **F** For any deterministic finite automaton, there is always a unique minimal non-deterministic finite automaton equivalent to it.

Trick question! It's the converse which is true.

(xv) **T** The union of any two context-free languages is context-free.

(xvi) **F** The class of languages accepted by non-deterministic push-down automata is the same as the class of languages accepted by deterministic push-down automata.

A language of palindromes is accepted by a PDA but not by any DPDA.

(xvii) **T** Let π be the ratio of the circumference of a circle to its diameter. Then π is recursive.

There is a program that, if run forever, would output the decimal expansion of π.

(xviii) **T** The Kleene closure of any recursive language is recursive.

Let $L \subseteq \Sigma^*$ be recursive, and let $w \in \Sigma^*$ of length n. Then W has $O(n^2)$ substrings. Let $\mathcal{L}$ be the set of all substrings of w which are in L, which we can campute since L is decidable. Then, in polynomial time, determine whether w is the concatenation of members of $\mathcal{L}$.

(xix) **T** If $\mathcal{P} = \mathcal{NP}$, then all one-way encoding systems are breakable in polynomial time.

If $\mathcal{P} = \mathcal{NP}$, there is no one-way function.

(xx) **T** A language L is in $\mathcal{NP}$ if and only if there is a polynomial time reduction of L to SAT.

SAT is $\mathcal{NP}$–complete.

(xxi) **T** The intersection of any context-free language with any regular language is context-free.

(xxii) **F** Let L be the set of all strings of the form $\langle G_1\rangle\langle G_2\rangle$ where G_1 and G_2 are equivalent context-free grammars. Then L is recursively enumerable.

But it is co-$\mathcal{RE}$.

(xxiii) **T** If L_1 reduces to L_2 in polynomial time, and if L_2 is $\mathcal{NP}$, and if L_1 is $\mathcal{NP}$-complete, then L_2 must be $\mathcal{NP}$-complete.

This is the most common way that new $\mathcal{NP}$–complete languages are found.

(xxiv) **O** The question of whether two regular expressions are equivalent is $\mathcal{NP}$-complete. (Do not guess. Look it up.)

The problem is –SPACE complete. But it is unknown whether $\mathcal{P}$–SPACE $= \mathcal{NP}$.

(xxv) **F** The intersection of any two context-free languages is context-free.

(xxvi) **T** If L_1 reduces to L_2 in polynomial time, and if L_2 is $\mathcal{NP}$, then L_1 must be $\mathcal{NP}$.

(xxvii) **T** Every language which is accepted by some non-deterministic machine is accepted by some deterministic machine.

But it might take exponentially longer to accept a string.

(xxviii) **T** The language of all regular expressions over the binary alphabet is a regular language.

Trick question. Impossible since the language of regular expressions has matched parentheses, and no such language can be regular. In fact, the language of all regular expressions over the binary alphabet is context-free.

(xxix) **F** The equivalence problem for C++ programs is recursive.

Machine equivalence is undecidable.

(xxx) **F** Every function that can be mathematically defined is recursive.

This is somewhat hard to explain. An example is the busy beaver function.

(xxxi) **F** There is some recursive function F such that $\beta(n) = O(F(n))$, where β be the busy beaver function.

The busy beaver function grows asymptotically faster than any recursive function, as expained on the Wikipedia page.

(xxxii) **T** A language is L is $\mathcal{NP}$ if and only if there is a polynomial time reduction of L to Boolean satisfiability.

A basic property of $\mathcal{NP}$–complete languages, in fact, one of the definitions.

(xxxiii) **T** If there is a recursive reduction of the halting problem to a language L, then L is undecidable.

If L_1 can be reduced to L_2 by an "easy" function then L_2 cannot be "harder" than L_1

(xxxiv) **F** If there is a recursive reduction of a language L to the halting problem, then L is undecidable.

The converse of the previous problem, but an easy language can be easily reduced to a hard language.

(xxxv) **T** The set of rational numbers is countable.

Every fraction can be thought of as the ordered pair (p,q) where p and q are integers. Since there are countably many integers, there are countably many such pairs.

(xxxvi) **F** The set of real numbers is countable.

Cantor's diagonalization proof shows that there are uncountably many real numbers.

(xxxvii) **T** The set of recursive real numbers is countable.

The decimal expansion of every recursive real number is written by some machine, and there are only countably many machines with a given alphabet.

(xxxviii) **F** There are countably many binary functions.

Let Σ^* be the set of all binary strings, which is countably infinite. The set of binary functions is then $(\Sigma*)^{\Sigma^*}$, which is uncountable.

(xxxix) **T** There are countably many recursive binary functions.

Each has to be computed by a machine, and there are countably many such machines.

2. Give a problem which is in both $\mathcal{NP}$ and co-$\mathcal{NP}$ but is not known to be $\mathcal{P}$.

   The 0/1 factoring problem over binary numerals. It is not known to be $\mathcal{P}$-time, but is known to be both $\mathcal{NP}$ and co-$\mathcal{NP}$.

3. Give a context-free language whose complement is not context-free.

   The complement of the language $\{a^n b^n c^n \, : \, n \geq 0\}$.

4. State the pumping lemma for regular languages.

   For any regular langage L
   There exists an integer p
   Such that for any string $w \in L$ of length at least p
   There exist string x, y, and z
   Such that the following statements hold
   1. $w = xyz$
   2. $|xy| \leq p$ 3. $y \neq \lambda$ (or $|y| \geq 1$)
   4. For any integer $i \geq 0$, $xy^iz \in L$.

5. State the Church-Turing thesis. Why is it important?

   Every computation by any machine can be emulated by some Turing machine. This is important because Turing machines are simple, making it easier to prove that a given computation cannot be done by a Turing machine, hence not by any machine.

6. For a given instance of an $\mathcal{NP}$ problem, a witness, a certificate, and a guide string all have the same purpose.

   Let L be any $\mathcal{NP}$ language. There exists a machine V (the verifier) and there exists an integer k such that

   (a) For any $w \in L$ there is a string c of length at most n^k, where $n = |w|$, such that V accepts the string w, c

   (b) If $w \notin L$ and c is any string, V does not accept w, c.

   The string c could be called either a "certificate" (certifying that $w \in L$) or a "witness."

   Alternatively, if L is accepted by a non-deterministic machine M is polynomial time, then a guide string for $w \in L$ is a string of instructions which tells M, with input w, what to do at each step where there is a choice, in order to reach a final state.

In all three cases, we mean a string of polynomial length which allows a machine to verify that $w \in L$. For example, for some Boolean expression, a witness would be an assignment of the variables which satisfies the expression.

7. Give a polynomial time reduction of the subset sum problem to the partition problem.

   If $X = (x_1, x_2, \dots x_n, K)$ is an instance of the subset sum problem, let $S = \sum_{i=1}^{n}$. Then $Y = (x_1, x_2, \dots x_n, S + 1, = (x_1, x_2, \dots x_n, S + 1, K - S + 1)$ is an instance of the partition problem which has a solution if and only if X has a solution.

8. On the internet you can find many instances of RUSH HOUR together with solutions. All the solutions given are short. Is it true that every solvable instance of RUSH HOUR has a solution of polynomial length? Explain your answer.

   I did not grade this problem, since I had not gone over $\mathcal{P}$–SPACE enough. A similar problem will be on the next examination, though.

9. Prove that every decidable language is can be enumerated by some machine. **In canonical order.**

   Let $w_1, w_2, \dots$ be the canonical enumeration of Σ^*. The following program enumerates a deciable language $L \subseteq \Sigma^*$ in canonical order:

```
For each $i$ from 1 to $\infty$
    If($w_i \in L$)
write $w_i$
```

   It is important that L be decidable, else the program could hang at the if statement, being unable to decide the condition.

10. Prove that every language which can be enumerated in canonical order by some machine is recursive.

    If L is finite, we are done, since a finite language is recursive.

    Otherwise, suppose some machine enumerates L in canonical order: $w_1, w_2, \dots$. The following program decides L.

```
Read $w$.
For i = 1 to $\infty$
  If($w = w_i$)
    Write "Yes" and halt
  else if ($w < w_i$) (in canonical order)
    Write "No" and halt
```

11. Prove that every language which can be enumerated by any machine is accepted by some machine.

    Suppose some machine enumerates L as $w_1, w_2, \ldots$, not necessarily in canonical order. The following program accepts L.

```
Read $w$
For $i = 1$ to $\infty$
  If($w = w_i$)
    Write "Yes" and halt.
```

12. Prove that every language which is accepted by any machine can be enumerated by some machine.

    Suppose $L \subseteq \Sigma^*$ is accepted by a deterministic machine M. The following program enumerates L, but in any order.

```
For $t = 1 to \infty$
  For i = 1 to t
    If $M$ accepts $w_i$ within its first $t$ steps
      Write $w_i$
```