University of Nevada, Las Vegas Computer Science 456/656 Spring 2024

CSC 456/656 Fall 2024 Answers to Second Examination March 6, 2024

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

   (i) **F** Every subset of a decidable language is decidable.
   Every language over an alphabet Σ is a subset of Σ^*, which is trivially decidable.

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

   (iii) **T** Every context-free language is $\mathcal{P}$.
   The CYK algorithm is $\mathcal{P}$–TIME.

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

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

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

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

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

   (ix) **T** The complement of every undecidable language is undecidable.

   (x) **T** There is a program that runs forever, writing a sequence of fractions which converges to π.

   (xi) **T** The square root of any positive recursive real number is recursive.

   (xii) **T** Given any unambiguous context-free grammar G and any string $w \in L(G)$, there is always a unique rightmost derivation of w using G.
   One of the definitions of an unambiguous grammar.

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

   (xiv) **F** The union of any two undecidable languages is undecidable.
   IF $L \subseteq \Sigma^*$ is undecidable, its complement L' is also undecidable, and $L + L' = \Sigma^*$, which is trivially decidable.

   (xv) **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.
   The languages of palindromes over $\{0.1\}$ is CF but not accepted by any DPDA.

   (xvi) **T** Let π be the ratio of the circumference of a circle to its diameter. Then π is recursive.
   There is a program that runs forever and prints the decimal expansion of π. I once wrote one that halts after a thousand digits.

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

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

That would be a social disaster!

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

Because SAT is $\mathcal{NP}$–complete.

(xx) **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.

The theorem that allows us to "bootstrap" out list of $\mathcal{NP}$–complete languages.

(xxi) **O** The question of whether two regular expressions are equivalent is $\mathcal{NP}$-complete.

But, it's known to be $\mathcal{P}$–SPACE.

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

$\{a^ib^ic^j\} \cap \{a^ib^jc^j\}$ is not context-free.

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

L_2 cannot be easier than L_1.

(xxiv) **F** Every language which is accepted by some non-deterministic machine is accepted by some deterministic machine.

But it might run exponentially longer.

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

No language which has matching delimeters, such as parentheses, can be regular.

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

It's not $\mathcal{RE}$, either.

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

There are lots of counter-examples, such as the busy beaver function.

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

Because Boolean $\mathcal{NP}$–$\underline{\text{complete}}$.

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

L cannot be easier that HALT.

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

The decimal (or binary, or whatever) expansion of a recursive real number is the output of some Pascal program, and there are only countably many Pascal programs.

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

Same reason as the answer to (xxx).

(xxxii) **T** Every context-sensitive language is recursive.

Any context-sensitive language is decidable by a Pascal program.

(xxxiii) **F** Every co-$\mathcal{RE}$ language is generated by some unrestricted grammar.

The class of unrestricted grammar generates the class of $\mathcal{RE}$ languages. None of the Chomsky classes generate co-$\mathcal{RE}$ languagres.

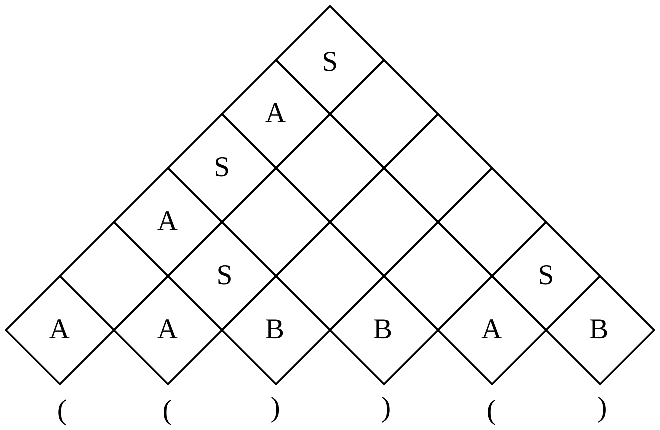
(xxxiv) **T** The tile placement problem is $\mathcal{NP}$–complete.

The <u>tile placement problem</u> is, given a finite set of tiles of various shapes and sizes, can the tiles be arranged inside a rectangle of a given size, such that no tiles overlap?

The tile placement problem is trivially in the class $\mathcal{NP}$, using the verification definition of that class, and there is a $\mathcal{P}$–TIME reduction of the partition problem to the tile placement problem. Do you see it?

2. The following CNF grammar generates all non-empty members of the Dyck language. Use the CYK agorithm to prove that this grammar generates (())().

$$\begin{aligned} &S \to AB \\ &A \to (\\ &B \to) \\ &A \to AS \\ &A \to SA \end{aligned}$$



3. Some time ago, one of my students submitted a proof that every language is decidable. Here is that proof.

Let $L \subseteq \Sigma^*$ be a language, and, for every n, let L_n be the language of all strings of length n in L. Since L_n is finite, it is decidable. Let M_n be a machine that decides L_n.

```
Read $w \in \Sigma^*$
$n = |w|$
If $M_n$ decides $w \in L_n$
    Accept $w$.
Else
    Reject $w$.
Halt.
```

Therefore, every language is decidable.

What is wrong with this proof?

I won't publish the answer now. No student in this class answered this correctly, but one of my students from an earlier class figured it out.

4. State the pumping lemma for context-free languages.

   For any regular language L
   There exists a number p such that
   For any $w \in L$ of length at least p
   There exist strings u, v, x, y, z such that
   1. $w = uvxyz$
   2. $|vxy| \leq p$
   3. $|v| + |y| > 0$
   4. For any $i \geq 0$ $uv^ixy^iz \in L$.

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

   The Church-Turing thesis states that any computation that can be done by any machine can be done by some Turing machine.

   This is important, since Turing machines are very simple, making it easier to prove that a computation cannot be done by proving it cannot be done by any Turing machine.

6. Explain the verifier definition of the class $\mathcal{NP}$. A language L is in the class $\mathcal{NP}$ if and only if, for any $w \in L$, there is a polynomial time proof that $w \in L$.

   That means there is a machine V such that, for any $w \in L$ there exists a string c such that V accepts the ordered pair (w, c) in time which is a polynomial function of $|w|$, and that, for any $w \notin L$, V does not accept (w, c) for any string c.

7. Give a polynomial time reduction of 3-SAT to the independent set problem.

   If $e = C_1 * C_2 * \cdots * C_k$ is any Boolean expression such that each C_i is the disjunction of three terms, each of which is either a variable or the negation of a variable, there is a graph G such that G has an independent set of order k if and only if e has a satisfying assignment. Each $C_i = t_{i,1} + t_{i,2} + {}_{i,3}$ Let the vertices of G be the set $V_{i,j}$ for all $1 \leq i \leq k$ and $j = 1, 2, 3$. There is an edge from $V_{i,j}$ to $V_{i',j'}$ if either $i = i'$ or $t_{i,j} * t_{i',j'}$ is a contradiction.

8. Prove that every decidable language can be enumerated in canonical order by some machine.

   Let L be a decidable language over an alphabet Σ^*, and let $w_1, w_2, \ldots$ be the canonical enumeration of Σ^*. The following program writes a canonical order enumeration of L.

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

9. Prove that the halting problem is undecidable.

   See handout.