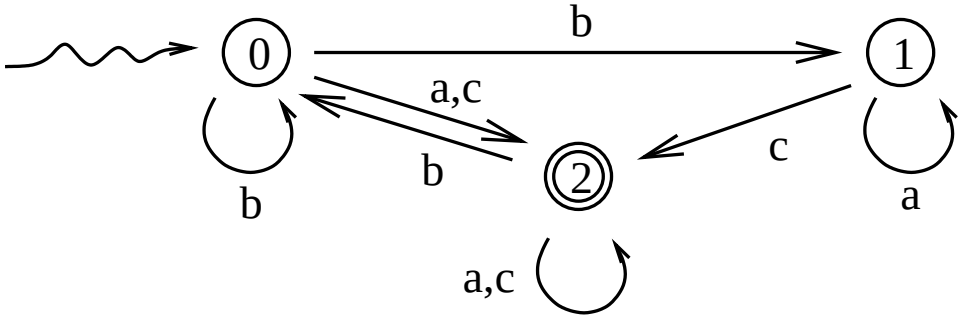
University of Nevada, Las Vegas Computer Science 456/656 Spring 2021

Practice Problems for the Examination on February 8, 2023

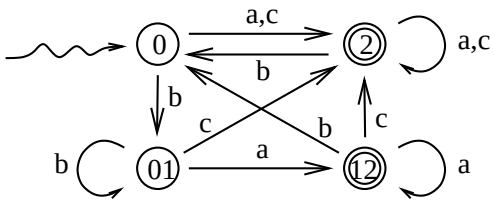
1. Review answers to homework1:
   `http://web.cs.unlv.edu/larmore/Courses/CSC456/S23/Assignments/hw1ans.pdf`

2. Review answers to homework2:
   `http://web.cs.unlv.edu/larmore/Courses/CSC456/S23/Assignments/hw2ans.pdf`

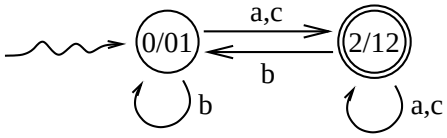
3. Find a minimal DFA equivalent to the NFA shown below.



	a	b	c
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
0	2	01	2
1	1	$\emptyset$	2
$*2$	2	0	2
01	12	01	2
$*02$	2	01	2
$*12$	1	0	2
$*012$	12	01	2



(a)



(b)

The transtion table for the initial DFA is to the left. Figure (a) shows that DFA when useless states are removed. Figure (b) shows the minimal DFA.

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

   I have struck out more questions that I definitely will not ask on the examination on February 8. I have also removed more duplicate question.

   (i) **T** The complement of every regular language is regular.

   (ii) **F** The complement of every context-free language is context-free.

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

   (iv) **T** ~~The complement of any $\mathcal{NP}$ language is $\mathcal{NP}$.~~

   (v) **T** ~~The complement of any $\mathcal{P}$-SPACE language is $\mathcal{P}$-SPACE.~~

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

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

(viii) **T** ~~Every language which is generated by a general grammar is recursively enumerable.~~

(ix) **T** ~~The context-free membership problem is undecidable.~~

(x) **T** ~~The factoring problem, where inputs are written in binary notation, is co-$\mathcal{NP}$.~~

(xi) **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.~~

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

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

F ~~The question of whether two regular expressions are equivalent is known to be $\mathcal{NP}$-complete.~~

(xiv) **T** ~~The halting problem is recursively enumerable.~~

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

(xvi) **F** ~~The question of whether a given Turing Machine halts with empty input is decidable.~~

(xvii) **T** The class of languages accepted by non-deterministic finite automata is the same as the class of languages accepted by deterministic finite automata.

(xviii) **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.~~

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

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

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

(xxii) **T** ~~Let π be the ratio of the circumference of a circle to its diameter. The problem of whether the n^{th} digit of the decimal expansion of π for a given n is equal to a given digit is decidable.~~

(xxiii) **T** ~~There cannot exist any computer program that can decide whether any two C++ programs are equivalent.~~

(xxiv) **T** ~~Every context-free language is in the class $\mathcal{P}$-TIME.~~

(xxv) **T** ~~Every regular language is in the class $\mathcal{NC}$~~

(xxvi) **F** ~~Every Function that can be mathematically defined is recursive.~~

(xxvii) **F** ~~The language of all binary strings which are the binary numerals for prime numbers is context-free.~~

(xxviii) **F** The language of all binary strings which are the binary numerals for prime numbers is regular.

(xxix) **F** ~~Every bounded function from integers to integers is Turing-computable. (We say that f is bounded if there is some B such that $|f(n)| \leq B$ for all n.)~~

(xxx) **F** ~~The language of all palindromes over $\{0,1\}$ is inherently ambiguous.~~

(xxxi) **F** ~~Every context-free grammar can be parsed by some deterministic top-down parser.~~

(xxxii) **T** ~~Every context-free grammar can be parsed by some non-deterministic top-down parser.~~

(xxxiii) **F** ~~Commercially available parsers cannot use the LALR technique, since most modern programming languages are not context-free.~~

(xxxiv) **T** ~~If anyone ever proves that $\mathcal{P} = \mathcal{NP}$, then all one-way encoding systems will be insecure.~~

(xxxv) **T** If a string w is generated by a context-free grammer G, then w has a unique leftmost derivation if and only if it has a unique rightmost derivation.

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

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

(xxxviii) **T** The intersection of any context-free language with any regular language is context-free. (It's in the chart.)

(xxxix) **T** ~~Every language which is generated by a general grammar is recursively enumerable.~~

(xl) **T** ~~The question of whether two context-free grammars generate the same language is undecidable.~~

(xli) **T** ~~There exists some proposition which is true but which has no proof.~~

(xlii) **T** ~~T~~he set of all binary numerals for prime numbers is in the class $\mathcal{P}$.

(xliii) **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.~~

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

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

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

(xlvii) **F** ~~No language which has an ambiguous context-free grammar can be accepted by a DPDA.~~

(xlviii) **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.~~

(xlix) **T** The intersection of any two regular languages is regular.

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

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

(lii) **T** ~~Let $F(0) = 1$, and let $F(n) = 2^{F(n-1)}$ for $n > 0$. Then F is recursive. (That means that some machine computes F.)~~

(liii) **T** Every language which is accepted by some non-deterministic machine is accepted by some deterministic machine. (Use brute force: try everything.)

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

(lv) **F** ~~For any real number x, The problem of whether the n^{th} digit of x, for a given n, is equal to a given digit is decidable.~~

(lvi) **T** ~~There cannot exist any computer program that decides whether any two given C++ programs are equivalent.~~

(lvii) **F** ~~An undecidable language is necessarily $\mathcal{NP}$-complete.~~

(lviii) **T** ~~Every context-free language is in the class $\mathcal{P}$-TIME.~~

(lix) **T** ~~Every context-free language is in the class $\mathcal{P}$-TIME.~~

(lx) **T** The language of all binary strings which are the binary numerals for multiples of 57 is regular.

(lxi) **F** ~~Commercially available parsers cannot use the LALR technique, since most modern programming languages are not context-free.~~

(lxii) **T** ~~If anyone ever proves that $\mathcal{P} = \mathcal{NP}$, then all public key/private key encryption systems will be known to be insecure.~~

(lxiii) **F** ~~If a sequence of fractions converges to a real number x, then x must be a recursive real number.~~

T ~~If a machine outputs a sequence of fractions which converges to a real number x, then x must be a recursive real number.~~