Answers to True/False Questions, Part I

If you find an error, let me know immediately!

1. True or False. T = true, F = false, and O = open, meaning that the answer is not known science at this time. In the questions below, $\mathcal{P}$ and $\mathcal{NP}$ denote $\mathcal{P}$-TIME and $\mathcal{NP}$-TIME, respectively.

   (i) **F** Let L be the language over $\{a, b, c\}$ consisting of all strings which have more a's than b's and more b's than c's. There is some PDA that accepts L.

   (ii) **T** The language $\{a^n b^n \mid n \geq 0\}$ is context-free.

   (iii) **F** The language $\{a^n b^n c^n \mid n \geq 0\}$ is context-free.

   (iv) **T** The language $\left\{a^i b^j c^k \mid j = i + k\right\}$ is context-free.

   (v) **T** The intersection of any three regular languages is regular.

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

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

   (viii) **T** If L is a context-free language over an alphabet with just one symbol, then L is regular.

   (ix) **T** There is a deterministic parser for any context-free grammar. (But not necessarily an LALR parser.)

   (x) **T** The set of strings that your high school algebra teacher would accept as legitimate expressions is a context-free language.

   (xi) **T** Every language accepted by a non-deterministic machine is accepted by some deterministic machine.

   (xii) **T** The problem of whether a given string is generated by a given context-free grammar is decidable.

   (xiii) **T** If G is a context-free grammar, the question of whether $L(G) = \emptyset$ is decidable.

   (xiv) **F** Every language generated by an unambiguous context-free grammar is accepted by some DPDA.

   (xv) **T** The language $\{a^n b^n c^n d^n \mid n \geq 0\}$ is recursive.

   (xvi) **T** The language $\{a^n b^n c^n \mid n \geq 0\}$ is in the class $\mathcal{P}$-TIME.

   (xvii) **O** There exists a polynomial time algorithm which finds the factors of any positive integer, where the input is given as a binary numeral.

   (xviii) **F** Every undecidable problem is $\mathcal{NP}$-complete.

   (xix) **F** Every problem that can be mathematically defined has an algorithmic solution.

   (xx) **F** The intersection of two undecidable languages is always undecidable.

   (xxi) **T** Every $\mathcal{NP}$ language is decidable.

(xxii) **T** The intersection of two $\mathcal{NP}$ languages must be $\mathcal{NP}$.

(xxiii) **F** If L_1 and L_2 are $\mathcal{NP}$-complete languages and $L_1 \cap L_2$ is not empty, then $L_1 \cap L_2$ must be $\mathcal{NP}$-complete.

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

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

(xxvi) **O** $\mathcal{NP} = \mathcal{P}$-SPACE

(xxvii) **O** $\mathcal{P}$-SPACE = EXP-TIME

(xxviii) **O** EXP-TIME = EXP-SPACE

(xxix) **F** EXP-TIME = $\mathcal{P}$-TIME.

(xxx) **F** EXP-SPACE = $\mathcal{P}$-SPACE.

(xxxi) **T** The traveling salesman problem (TSP) is $\mathcal{NP}$-complete.

(xxxii) **T** The knapsack problem is $\mathcal{NP}$-complete.

(xxxiii) **T** The language consisting of all satisfiable Boolean expressions is $\mathcal{NP}$-complete.

(xxxiv) **T** The Boolean Circuit Problem is in $\mathcal{P}$.

(xxxv) **O** The Boolean Circuit Problem is in $\mathcal{NC}$.

(xxxvi) **F** If L_1 and L_2 are undecidable langugages, there must be a recursive reduction of L_1 to L_2.

(xxxvii) **T** The language consisting of all strings over $\{a, b\}$ which have more a's than b's is LR(1).

(xxxviii) **T** 2-SAT is $\mathcal{P}$-TIME.

(xxxix) **O** 3-SAT is $\mathcal{P}$-TIME.

(xl) **T** Primality is $\mathcal{P}$-TIME.

(xli) **T** There is a $\mathcal{P}$-TIME reduction of the halting problem to 3-SAT.

(xlii) **T** Every context-free language is in $\mathcal{P}$.

(xliii) **O** Every context-free language is in $\mathcal{NC}$.

(xliv) **T** Addition of binary numerals is in $\mathcal{NC}$.

(xlv) **O** Every context-sensitive language is in $\mathcal{P}$.

(xlvi) **F** Every language generated by a general grammar is recursive.

(xlvii) **F** The problem of whether two given context-free grammars generate the same language is decidable.

(xlviii) **T** The language of all fractions (using base 10 numeration) whose values are less than π is decidable. (A *fraction* is a string. "314/100" is in the language, but "22/7" is not.)

(xlix) **T** There exists a polynomial time algorithm which finds the factors of any positive integer, where the input is given as a unary ("caveman") numeral.

(l) **T** For any two languages L_1 and L_2, if L_1 is undecidable and there is a recursive reduction of L_1 to L_2, then L_2 must be undecidable.

(li) **F** For any two languages L_1 and L_2, if L_2 is undecidable and there is a recursive reduction of L_1 to L_2, then L_1 must be undecidable.

(lii) **F** If P is a mathematical proposition that can be written using a string of length n, and P has a proof, then P must have a proof whose length is $O(2^{2^n})$.

(liii) **T** If L is any $\mathcal{NP}$ language, there must be a $\mathcal{P}$–TIME reduction of L to the partition problem.

(liv) **F** Every bounded function is recursive.

(lv) **O** If L is $\mathcal{NP}$ and also co-$\mathcal{NP}$, then L must be $\mathcal{P}$.

(lvi) **T** If L is $\mathcal{RE}$ and also co-$\mathcal{RE}$, then L must be decidable.

(lvii) **T** Every language is enumerable.

(lviii) **F** If a language L is undecidable, then there can be no machine that enumerates L.

(lix) **T** There exists a mathematical proposition that can be neither proved nor disproved.

(lx) **T** There is a non-recursive function which grows faster than any recursive function.

(lxi) **T** There exists a machine that runs forever and outputs the string of decimal digits of π (the well-known ratio of the circumference of a circle to its diameter).

(lxii) **F** For every real number x, there exists a machine that runs forever and outputs the string of decimal digits of x.

(lxiii) **O Rush Hour**, the puzzle sold in game stores everywhere, generalized to a board of arbitrary size, is $\mathcal{NP}$–complete.

(lxiv) **O** There is a polynomial time algorithm which determines whether any two regular expressions are equivalent.

(lxv) **O** If two regular expressions are equivalent, there is a polynomial time proof that they are equivalent.