CS 3350 Automata, Computability, and Formal Languages
Spring 2019, Test 3

Last 4 digits of UTEP ID: __________

General comments:

• you are allowed up to 5 pages of handwritten notes;
• if you need extra pages, place your name on each extra page;
• the main goal of most questions is to show that you know the corresponding algorithms; so, if you are running out of time, just follow the few first steps of the corresponding algorithm.

Good luck!
1. Illustrate the pumping lemma for context-free grammars by showing how it will represent the word \( w = -10.01 \) which is generated by the CFG with starting variable \( V \) and rules \( V \rightarrow SN.N, V \rightarrow N.N, S \rightarrow +, S \rightarrow -, N \rightarrow DN, N \rightarrow D, D \rightarrow 0, \) and \( D \rightarrow 1 \) as \( uvxyz \). Show, step-by-step, how the corresponding word \( uvvxyyz \) can be derived from this CFG.

\[ u = - \]
\[ v = 1 \]
\[ x = 0 \]
\[ y = \varepsilon \]
\[ z = .01 \]

\[ uvxyz = -10.01 \]
\[ uvvxxyz = -110.01 \]
2. Prove that the language of all the words that contain equal number of 0's, 1's, and 2's is not context-free. **Comment:** this language contains words 012, 102, 021, 012012, etc.

**Thm.** \( L = \{0^n1^m2^n : n = 0, 1, 2, \ldots \} \) is not context-free.

**Proof (by contradiction):** Assume \( L \) is CFG. Then by pumping lemma \( \exists p \in \mathbb{N} : \forall n \geq p \) if \( w \in L \) then \( \exists u, v, x, y, z \) s.t. \( w = uvxyz \) and \( |vxy| \leq p \) and \( |vy| > 0 \) and \( uv^i(vxy)^jz \in L \) for all \( i, j \geq 0 \). Let's take \( w = 0^p1^p2^p \). So \( |w| = 3p \geq p \), so \( w = uvxyz \) by pumping lemma.

Let's consider the possible locations of \( vxy \).

1) \( vxy \) is in 0's. When we go from \( uvxy^2z \) to \( uvvxyyz^2 \), we add 0's but not 1's or 2's, so balance is disrupted \( \not\in L \).
2) \( vxy \) is in between 0's & 1's. When we go from \( uvxy \) to \( uvvxyyz^2 \), we add 0's & 1's but not 2's, so balance is disrupted \( \not\in L \).
3) \( vxy \) is in 1's. We add 1's but not 0's or 2's, so balance is disrupted \( \not\in L \).
4) \( vxy \) is in between 1's & 2's. We add 1's & 2's but not 0's, so balance is disrupted \( \not\in L \).
5) \( vxy \) is in 2's. We add 2's but not 0's or 1's, so balance is disrupted \( \not\in L \).

In all cases, \( uvvxyyz^2 \not\in L \), but by pumping lemma \( uvvxyyz^2 \in L \). This contradiction shows our assumption was wrong, so \( L \) is not CFG.
3. Trace the following Turing machine on the example of the word 01: start, _ → work, R (here, _ means blank); work, 0 → R; work, 1 → 0, R; work, _ → back, L; back, 0 → L; back, _ → halt.

Explain how each step will be represented if we interpret the Turing machine as a finite automaton with two stacks.
4. Design a Turing machine that, given a sequence of 0s and 1s, replaces all 0s with 1s and all 1s with 0s. For example, the string 101 should be transformed into 010. Trace your Turing machine, step-by-step, on the example of the string 101. Why in Turing machines (and in most actual computers) the representation of a binary number starts with the least significant digit?

\[
\begin{array}{c}
0 \quad 1 \quad 0 \\
\text{start} \\
1 \quad 0 \quad 1 \\
\text{move} \\
0 \quad 0 \quad 1 \\
\text{move} \\
1 \\
\text{move} \\
0 \quad 1 \quad 0 \\
\text{back} \\
0 \quad 1 \quad 0 \\
\text{back} \\
0 \quad 1 \quad 0 \\
\text{back} \\
0 \quad 1 \quad 0 \\
\text{back, halt}
\end{array}
\]

Because every operation is started like that, starting with the least significant digit. So we save time.

\[
\begin{align*}
\text{start, } & \rightarrow \text{move, } R \\
\text{move, } 0 & \rightarrow 1, R \\
\text{move, } 1 & \rightarrow 0, R \\
\text{move, } & \rightarrow \text{back, } L \\
\text{back, } 0 & \rightarrow L \\
\text{back, } 1 & \rightarrow L \\
\text{back, } & \rightarrow \text{halt}
\end{align*}
\]
5. The following finite automaton describes even binary numbers. This automaton has the starting state $s$, the final state $e$, an additional state $d$, and the following transitions: from each of the three states, 0 leads to $e$, while 1 leads to $d$. Use the general algorithm to transform this finite automaton into a Turing machine. Show, step-by-step, how your Turing machine will accept the string 10.

\[
\begin{align*}
\text{start,} & \rightarrow s, R \\
\text{s, 0} & \rightarrow e, R \\
\text{s, 1} & \rightarrow d, R \\
\text{e, 0} & \rightarrow R \\
\text{e, 1} & \rightarrow d, R \\
\text{d, 0} & \rightarrow e, R \\
\text{d, 1} & \rightarrow R \\
\end{align*}
\]

\[
\text{start,} \rightarrow \text{reject} \\
\text{d,} \rightarrow \text{reject} \\
\text{e,} \rightarrow \text{accept}
\]
6. Give the formal definition of a feasible algorithm. Give two examples:

1. of a computation time which is formally feasible, but not practically feasible, and
2. a computation time which is practically feasible but not formally feasible.

Feasible algorithm = an algorithm is feasible if there exists a polynomial \( P(n) \) such that for every input \( x \), the algorithm finishes on or before time \( P(\text{len}(x)) \)

\[
1. \quad n^{10^{12}} \quad \text{and} \quad t = 10^{140} \\
2. \quad 2^{10^10}n \quad \text{and} \quad n = 10^8
\]
7. What is P? What is NP? What does it mean for a problem to be NP-hard? NP-complete? Give brief definitions. Give an example of an NP-complete problem. Is P equal to NP?

\[ P = \text{class of all problems that can be solved in polynomial time.} \]
\[ NP = \text{class of all problems for which once we have a candidate solution, we can check in polynomial time whether it is indeed a solution.} \]
\[ NP\text{-hard} = \text{every problem from the class NP can be reduced to it.} \]
\[ NP\text{-complete} = \text{NP-hard and NP}\]

\[ \text{ex. traveling salesman problem: given a set of cities and all distances, what is the shortest round trip (in distance) to take to visit all given cities?} \]

\[ P = NP? \text{ We don't know. It is an open problem.} \]
8. Prove that the square root of 5 is not a rational number.

**Thm.** \( \sqrt{5} \) is irrational.

**Proof (by contradiction):** Let's assume \( \sqrt{5} \) is rational. Then \( \sqrt{5} = \frac{a}{b} \), where \( a, b \in \mathbb{Z} \), \( b \neq 0 \), and \( a \) and \( b \) have no common factors. Then \( 5 = \frac{a^2}{b^2} \iff 5b^2 = a^2 \). So \( a^2 \) is divisible by 5, so \( a \) is divisible by 5. So \( a = 5p \) for some \( p \in \mathbb{Z} \). Now \( 5b^2 = (5p)^2 \iff 5b^2 = 25p \iff b^2 = 5p \). So \( b^2 \) is divisible by 5, so \( b \) is divisible by 5. Now \( a \) and \( b \) have a common factor of 5. But we had stated \( a \) and \( b \) have no common factors. We get a contradiction, so our assumption is wrong. Thus, \( \sqrt{5} \) is irrational.
9. Formulate the halting problem. For extra credit: prove that it is not possible to check whether a given program halts on given data.

**Thm.** No algorithm is possible such that given a pair \((p, d)\) checks whether program \(p\) halts on data \(d\).

**Proof (by contradiction).** Let's assume such an algorithm exists:

\[
\begin{align*}
h(p, d) &= \begin{cases} 
\text{true} & \text{if } p \text{ halts on } d \\
\text{false} & \text{otherwise}
\end{cases}
\end{align*}
\]

Design an auxiliary program.

```java
public static boolean pr(String x) {
    if (h(x, x)) {
        while (true) {
            x = x;
        }
    } else return true;
}
```

Consider \(pr(pr)\). Will it halt? If it halts, then \(h(pr, pr) = \text{true} \), \(pr\) will go into an infinite loop, so it will not halt. If it does not halt, \(h(pr, pr) = \text{false} \), \(pr\) will return \(true\), so it will halt. So if we assume that \(h(p, d)\) exists, we get a contradiction. So our assumption is wrong: such an algorithm does not exist.
10. Formulate Church-Turing thesis. Is it a mathematical theorem? Is it a statement about the physical world?

CT Thesis: Anything that can be computed on any physical device can also be computed by a Java program on a Turing machine.

CT Thesis depends on physics, it is not a mathematical statement (can’t be proved).