CS103
Spring 2012

Handout 25
June 8, 2012

CS103 Final Exam

This final exam is open-book, open-note, open-computer, but closed-network. This means that if you want to have your laptop with you when you take the exam, that's perfectly fine, but you **must not** use a network connection. You should only use your computer to look at notes you've downloaded in advance. Although you may use laptops, you **must** hand-write all of your solutions on this physical copy of the exam. No electronic submissions will be considered without prior consent of the course staff.

SUNetID: _____

Last Name: _____

First Name: _____

I accept both the letter and the spirit of the honor code. I have not received any assistance on this test, nor will I give any.

(signed) _____

You have three hours to complete this exam. There are 180 total points, and this exam is worth 25% of your total grade in this course. You may find it useful to read through all the questions to get a sense of what the test contains. As a rough sense of the difficulty of each question, there is one point on this exam per minute of testing time.

(1) Relations Revisited

(2) Regular Languages

(3) Context-Free Languages

(4) **R** and **RE** Languages

(5) **P** and **NP** Languages

/30	
/30	
/25	
/60	
/35	
/180	

It has been a pleasure teaching CS103 this quarter. Good luck on the exam!

Problem One: Relations Revisited (30 points total)

Recall that a binary relation R over a set A is formally defined as a subset of $A \times A$; that is, R is a set of ordered pairs (x, y) where xRy iff $(x, y) \in R$. This means that in addition to our previous treatment of relations, we can consider relations from a set-theoretic perspective.

(i) Properties of Equivalence Relations (15 Points)

Prove or disprove: Every binary relation R over a set A is a subset of some equivalence relation over the set A.

(ii) Properties of Partial Orders **(15 Points)**

Prove or disprove: Every binary relation R over a set A is a subset of some partial order relation over the set A.

Problem Two: Regular Languages (30 points total)

Given two strings of **0**s and **1**s, we say that those strings have the same **1**-parity if both of the strings contain an odd number of **1**s or both of the strings strings contain an even number of **1**s.

Consider the following language over $\Sigma = \{ \mathtt{0}, \mathtt{1}, \mathtt{M} \}$:

$$\mathit{1PARITY} = \{ w\mathtt{M}x \mid w, x \in \{ \mathtt{0}, \mathtt{1} \}^* \text{ and } w \text{ and } x \text{ have the same } \mathtt{1}\text{-parity.} \}$$

For example, **01M111** $\in$ *1PARITY*, **0011M111111** $\in$ *1PARITY*, and **M** $\in$ *1PARITY*. However, **1M0** $\notin$ *1PARITY*, **MM** $\notin$ *1PARITY*, **00M01** $\notin$ *1PARITY*, and $\varepsilon \notin$ *1PARITY*.

(i) Finite Automata (10 Points)

Design a DFA that accepts *1PARITY*.

(ii) Regular Expressions (10 Points)

Write a regular expression for *1PARITY*.

Consider the following language over the alphabet $\Sigma = \{\mathtt{1}, \mathtt{\geq}\}$:

$$GE = \{\ \mathtt{1}^m\mathtt{\geq}\mathtt{1}^n \mid m, n \in \mathbb{N} \text{ and } m \geq n\ \}$$

For example, $\mathtt{111\geq1} \in GE$, $\mathtt{1111\geq111} \in GE$, and $\mathtt{1\geq} \in GE$. However, $\mathtt{111\geq1111} \notin GE$, $\mathtt{\geq\geq1} \notin GE$, and $\varepsilon \notin GE$.

(iii) The Pumping Lemma **(10 Points)**

Using the pumping lemma for regular languages, prove that GE is not regular. To help out, we have sketched out a part of the proof; you should fill in the appropriate blanks.

Proof: By contradiction; assume that GE is regular. Let n be the length guaranteed by the pumping lemma. Then consider the string $w =$ [ ]
We then have that $w \in GE$ and $|w| \geq n$, so by the pumping lemma we can write $w = xyz$ such that $|xy| \leq n$, $y \neq \varepsilon$, and for any $i \in \mathbb{N}$, $xy^iz \in GE$.

(finish the proof in the box below)

We have reached a contradiction, so our assumption was wrong and GE is not regular. ■

Problem Three: Context-Free Languages (25 Points)

(i) Writing CFGs (10 Points)

Let $\Sigma = \{$ `0`, `1` $\}$. Consider the language *NEP* defined as follows.

$$NEP = \{ w \mid w \text{ is not an even-length palindrome} \}$$

For example, `0111` $\in$ *NEP*, `101` $\in$ *NEP*, and `101010` $\in$ *NEP*. However, `10100101` $\notin$ *NEP*, $\varepsilon \notin$ *NEP*, and `0000` $\notin$ *NEP*.

Write a CFG for *NEP*.

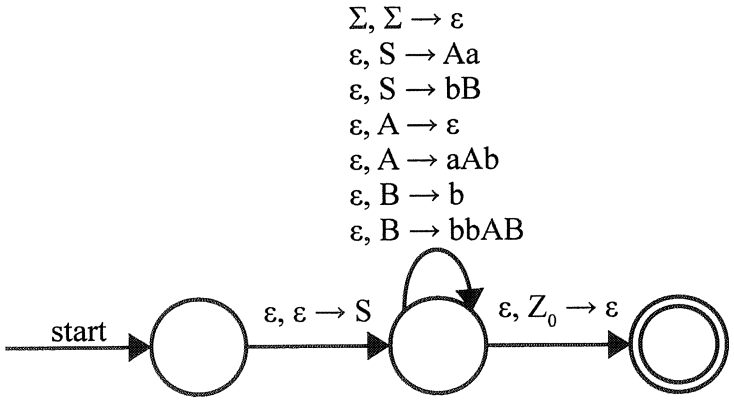
In lecture, we sketched a proof that if a language is context-free, there is a PDA for that language. Our proof constructed a PDA from an arbitrary CFG that tried to simulate a derivation of the input string from the start symbol. The construction built a PDA with three states:

- A **start** state that sets up the PDA's stack with the start symbol of the grammar,
- A **parsing** state where each transition simulates either *predicting* which production to use or *matching* a predicted symbol with the next character of the input.
- An **accepting** state entered when we find that the input string could be parsed.

The only part of the PDA that depends on the grammar is the set of transitions from the parsing state to itself. This state always has the transition $\Sigma, \Sigma \rightarrow \varepsilon$ to itself. Additionally, it has one transition for every production in the grammar. For example, given the following grammar:

$$S \rightarrow Aa \mid bB$$
$$A \rightarrow \varepsilon \mid aAb$$
$$B \rightarrow b \mid bbAB$$

The resulting PDA is as follows:



Given a grammar G, let's denote by $P(G)$ the automaton constructed this way.

For most grammars G, $P(G)$ is a nondeterministic PDA. However, there are many grammars G for which $P(G)$ is a deterministic PDA.

(The question is on the next page. Feel free to tear this page out as a reference.)

(ii) Deterministic Parsing Automata **(15 Points)**

What property or properties must a context-free grammar G have for P(G) to be a DPDA? Explain why P(G) is a DPDA iff G has the property or properties that you describe, though you don't need to formally prove it. Make sure to address both directions of implication.

Problem Four: R and RE Languages (60 Points)

(i) RE and Verifiers (20 Points)

Recall that a verifier for a language L is a Turing machine V such that

$$w \in L \quad \text{iff} \quad \exists x \in \Sigma^*.\ V \text{ accepts } \langle w, x \rangle$$

In the context of **NP** we considered polynomial-time verifiers, verifiers that run in time polynomial in the size of w. Let's relax this description and define a *deciding verifier* to be a verifier V for a language L such that V is a decider (that is, V halts on all inputs).

Prove that if there is a deciding verifier for a language L, then $L \in$ **RE**.

(ii) Unsolvable Problems **(25 Points)**

Consider the following language L_{some}:

$$L_{some} = \{ \langle M_1 \rangle \mid \mathscr{L}(M_1) \neq \emptyset \text{ and } \mathscr{L}(M_1) \neq \Sigma^* \}$$

Prove that L_{some} is not **RE**.

(iii) Properties of Reductions **(15 Points)**

Prove or disprove: If $L_1 \leq_M L_2$ and $L_1 \leq_M L_3$, then $L_1 \leq_M L_2 \cap L_3$.

Problem Five: P and NP Languages (35 points total)

Just how hard are the **NP**-complete problems? In a sense, they are the "hardest" problems in **NP**, because a solution to any one of them can be used to solve all other **NP** problems. How accurate is that intuition?

It turns out that is possible to construct languages that are **NP**-complete but which can be decided efficiently in many cases. One way to do this uses the disjoint union operation that you saw in Problem Set 8. Recall that given languages L_1 and L_2 over $\{\mathtt{0}, \mathtt{1}\}^*$, the disjoint union $L_1 \uplus L_2$ is the language

$$L_1 \uplus L_2 = \{ \mathtt{0}w \mid w \in L_1 \} \cup \{ \mathtt{1}w \mid w \in L_2 \}$$

(i) Relatively Easy NP-Complete Languages (20 Points)

Let L_1 be an **NP**-complete language and L_2 be any language in **P**. Prove that $L_1 \uplus L_2$ is **NP**-complete. This new language, while **NP**-complete, is easy for many inputs; we can decide in polynomial-time whether any string starting with a $\mathtt{1}$ is contained within $L_1 \uplus L_2$.

(More space for your answer to Problem 5.i, if you need it.)

(ii) $P \stackrel{?}{=} NP$ **(15 Points)**

This problem explores the question

What would it take to prove whether P = NP?

Below are ten statements. For each statement, if the statement would definitely prove that **P = NP**, write "**P = NP**" on the appropriate line. If the statement would definitely prove that **P ≠ NP**, write "**P ≠ NP**" on the appropriate line. If the statement would not prove either result, write "neither" on the appropriate line. No explanation is necessary.

There is a regular expression for SAT. _____

There is ***no*** regular expression for SAT. _____

There is a ***deterministic*** polynomial-time algorithm for SAT. _____

There is a ***nondeterministic*** polynomial-time algorithm for SAT. _____

Every $f(n)$-time ***single-tape*** NTM can be converted into an $f(n)^8$-time ***single-tape*** TM. _____

Every $f(n)$-time ***multitape*** NTM can be converted into an $f(n)^8$-time ***multitape*** TM. _____

For any k, there is a language in **NP** that cannot be decided in time $O(n^k)$. _____

There is a language in **NP** that, for any k, cannot be decided in time $O(n^k)$. _____

There is a polynomial-time TM that correctly decides SAT for all strings of length ***at most*** 10^{100}, but that might give incorrect answers for longer strings. _____

There is a polynomial-time TM that correctly decides SAT for all strings of length ***at least*** 10^{100}, but that might give incorrect answers for shorter strings. _____