University of Virginia - cs3102: Theory of Computation Spring 2010

Exam 1 - Preliminary Comments **4 March 2010**

Average: 71.6
Distribution: 90-100: 8; 80-89: 12; 70-79: 14; 60-69: 10; below 60: 11

Revision Opportunity - Read Carefully

These comments are not complete. In particular, only partial solutions are provided for questions 2c, 2d, 3, 5, and 6b. There is also an opportunity to present a revised answer to question 4, describe in the answer for question 4b. If you are not satisfied with your exam score, you may increase your score by turning in correct, well-written, and clear answers to as many of those questions as you wish. Your scores on the revised answers will replace your original scores for those questions. The grading expectations for the revised answers, though, will be much higher than for the original exam since you have more time to prepare a well written, clear, and convincing answer, and should benefit from the hints provided here.

Revised answers must be turned in on paper at the beginning of class of **Tuesday, March 16**. No answer revisions will be accepted after 2:01pm on March 16.

Honor Policy (please read and follow carefully!). If you have already discussed problems on the exam with others, kudos to you for being a curious, active student! You are free to use anything you learned from these conversations in your answers, but should include a credit saying with whom you discussed the problem. From now on, though, you should not discuss any of the problems on this exam until class on March 16. Except for other people, you may also use any other resources you want with one exception (described in the next sentence), but should cite any resources you use other than the textbook or course materials. The exception is that you may not submit any queries to search engines that contain the word "squarefree". Otherwise, you may use any printed or web resources you want to help in preparing your revised answers.

Problem 1: Definitions. For each question, provide a correct, clear, precise, and concise answer from the perspective of a theoretical computer scientist.

a. (Average: 4.9 / 5) What is a *language*?

Answer: A language is a *set of strings*.

b. (4.7 / 5) How do we measure the *power* of a type of machine such as a DFA or NPDA?

Answer: We measure the power of a machine type by the *set of languages it can recognize*.

Note that it is not correct to say the "number" of languages. All of the machine types we have defined can recognize infinitely many languages. Later in the class (and possibly in Prof. Robins' lecture today!) we will consider the question of whether two infinite sets can be of different sizes.

c. (4.3 / 5) What does it mean for a machine model to be *nondeterministic*?

Answer: A nondeterministic machine can have more than one possible action for a given configuration; it always chooses an action that leads to an accepting state if there is such an action. (Alternately, it tries all possible actions, and if any sequence of actions leads to an accept state it accepts.) Just saying that there can be more than one transition at each step was worth 4 points, but is not a satisfactory definition since the always guesses correctly property is crucial to understanding nondeterministic machine models.

Problem 2: Language Classification.

For each of these questions, provide a convincing argument supporting the proposition. You may use any technique you wish to make your argument, and may assume any of the properties we have proved in class or in the book.

a. (2.9 / 5, semi-trick question) Show that the language produced by the context-free grammar below is a regular language:

$$S \quad \rightarrow \quad 0S0 \mid 1S1$$

Answer: Since there is no rule that replaces S with only terminals, the grammar cannot produce *any* strings. This means the language it describes is the empty language. This is a regular language since it is recognized by a DFA with no accepting states ($F = \emptyset$).

b. (4.7 / 5) Show that the language produced by the context-free grammar below is a regular language:

$$S \quad \rightarrow \quad 0S \mid S1 \mid \epsilon$$

Answer: The grammar describes the language 0^*1^*. This is a regular language since we can describe it with a regular expression.

c. **Revision Opportunity!** (4.0 / 5) Show that the language $NOTREPEATING$ defined below is **not** a regular language:

$$NOTREPEATING = \{w | w \in \{0,1\}^* \wedge \text{there is no } x \text{ such that } w = xx\}$$

Answer: The easiest way to prove $NOTREPEATING$ is not a regular language is to use the property that the regular languages are closed under complement. This means that if we can prove that the complement of $NOTREPEATING$ is not a regular language, that proves that $NOTREPEATING$ itself is not a regular language. The complement of $NOTREPEATING$ is the language,

$$REPEATING = \{w | w \in \{0,1\}^* \wedge \text{there is an } x \text{ such that } w = xx\}$$

which is the same as the language, $\{ww | w \in \{0,1\}^*\}$. We have proven that this language is not context-free, so it must be non-regular.

It was also possible to get full credit for this question with an informal argument that explains that recognizing the language requires an infinite amount of state since we need to keep track of the entires first half of the input to know if the second half matches.

Another way to prove $NOTREPEATING$ is non-regular is to use the pumping lemma for regular languages. The easiest way to do this is to use the reverse version of the pumping lemma that starts with a string not in the language and pumps to produce a string in the language (which is just like using the complement property). A much harder way to prove this is to use the standard pumping lemma. A correct proof that uses the standard pumping lemma to prove this is worth a revised score on this question.

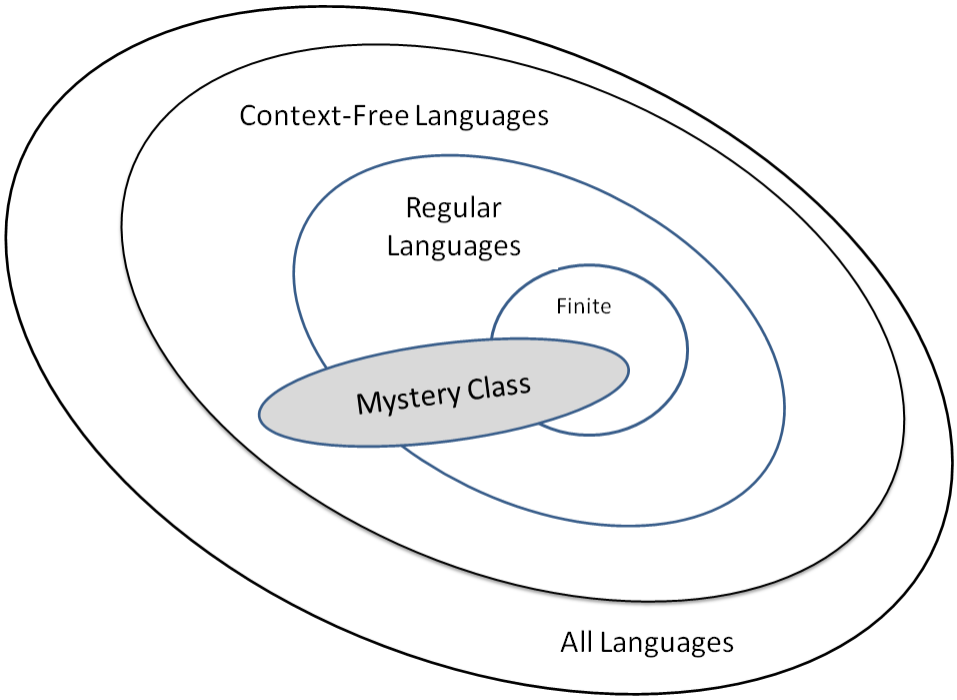
d. (3.7 / 5 + 5) **Revision Opportunity!** A *squarefree* sequence is a sequence that contains no adjacent repeating subsequences of any non-zero length. For example, ab and $abcba$ are squarefree, but aa is not since it contains a^2 and $abcbacbab$ is not since it contains $(cba)^2$. Show the language consisting of all squarefree strings in $\{a,b,c\}^*$ is not context-free.

Answer: The reason this is worth "5 + 5" points is because you receive full credit for a standard pumping lemma proof of this property, even though there is a flaw in such a proof. The problem is choosing a string s that is in the language and satisfies $|s| \geq p$. It is not necessary to choose a specific string to make the proof work, but it is necessary to argue that such a string must exist, otherwise the proof is broken. For this language, it is true, but not at all obvious, that a squarefree sequence longer than p exists for any p. There are at least three ways to deal with this problem. The most obvious (but hardest) way is to prove that the exist squarefree sequences longer than any length.

A revised answer consisting of well-written, correct, and convincing proof is worth up to 10 points.

Problem 3: Language Classes.

(7.4 / 15) **Revision Opportunity!** Define a type of machine that recognizes the set of languages in the *Mystery Class* ellipse depicted below:



Your class should include infinitely many non-regular languages, but there should be infinitely many finite languages that are not included in your class. For full credit, your answer should include a convincing argument why your class includes infinitely many non-regular languages and excludes infinitely many finite languages.

Answer: Many people had a tough time with this one. You answer should be a description of a machine model.

The most common not-quite-correct answer (worth 13 points) is a PDA with a limited number of states, for example PDAs where $|Q| \leq 8$. The problem with this answer is such a machine *can* recognize *all finite languages*. (Hint: the stack alphabet for a PDA is finite, but not bounded by any particular value.)

A more correct (worth full credit), but not completely satisfying, answer is to limit the input alphabet. This is more of a cosmetic change to the languages, since there is no important difference between the languages $a^n b^n$ and $0^n 1^n$. If the input string includes symbols not in the input alphabet, it is an invalid input string (which is different from saying the machine would reject the string, since the transition function is undefined for input symbols that are not in Σ).

A full-credit revised answer should include a convincing argument what the limited-state PDA can recognize all finite languages, define a machine that does recognize the Mystery Class, and a convincing argument why your machine recognizes exactly the described class of languages.

Problem 4: Broken Proofs.

Each of these "proofs" claims to prove a conjecture that is false. For each proof, identify the *first step* that is wrong, and briefly and clearly explain why. The explanation is more important than the step you identify.

a. (4.1 / 5) **False Conjecture:** The intersection of two context free languages is context free.

Claimed Proof.

1. We define the intersection of two languages
$$A \cap B = \{w | w \in A \wedge w \in B\}.$$
2. The language $X = A \cap B$ is a subset of the language A: $X \subseteq A$. This is true since every string in X must also be in A.
3. Since X is a subset of A, and A is context free, X is context free.

Answer: Step 3: knowing X is a subset of a context-free language does not tell us anything about X. For example, $\{0,1\}^*$ is a context-free language (it is also a regular language), but the non-context-free language $\{ww | w \in \{0,1\}^*\}$ is a subset of $\{0,1\}^*$.

b. (4.0 / 7) **Revision Opportunity! False Conjecture:** All whole numbers are even.

Claimed Proof. We use induction on the numbers to prove the conjecture.

1. A number n is *even* if $2m = n$ for some integer m.
2. *Basis:* 0 is even. Select $m = 0$, then $2m = 0$.
3. *Induction:* Assume the conjecture holds for all $i < n$. We show that it holds for n.
4. $n = k + 2$ for some integer $k < n$.
5. By the induction hypothesis, $k = 2m$ for some integer m.
6. $n = k + 2 = (2m) + 2 = 2(m + 1)$. Thus, n is even.

Answer: The problem is with step 5 (as it connects with step 4).

The induction hypothesis is about *whole numbers*, but this step uses integers. It is true that $n = k + 2$ for some integer $k < n$, but the induction hypothesis only

applies to whole numbers. It is not true that $n = k + 2$ for some whole number, since if $n = 1$ there is no non-negative number that satisfies this property.

Many people identified issues that are not problems, but reflected a very rigid notion of induction proofs. For example, you may expect the induction hypothesis to be written like, *Assume the conjecture holds for all* $i \leq n$. *Show it holds for* $n + 1$. If n is an integer, this means the exact same thing as showing it holds for n, assuming it holds for all $i < n$. It is also not the case that all induction proofs have to increment by one. The requirement is that if we want to prove some property holds for all members of a given set, the generation method used in the induction has to produce all members of the set.

The revision opportunity for this question is to demonstrate that you understand this by providing a *correct* induction proof of the following proposition:

> All even numbers greater than zero can be expressed as the sum of two odd numbers.

c.(6.0 / 8, *tricky*) **False Conjecture:** The language $TRIPLES$ is not regular.

$$TRIPLES = \{1^{3n} | n \geq 0\}$$

Claimed Proof. We use the pumping lemma for regular languages to obtain a contradiction.

1. Assume $TRIPLES$ is regular. Then, there exists some DFA M with pumping length p that recognizes $TRIPLES$.
2. Choose $s = 1^p$.
3. The pumping lemma requires that there is a way to divide s into $s = xyz$ where $|y| \geq 1$ and $|xy| \leq p$ and $xy^iz \in TRIPLES$ for all $i \geq 0$.
4. Since s is all 1s, we know y can contain only 1s.
5. Choose $y = 1$.
6. Choose $i = 2$.
7. Since xy^2z now has $p + 1$ ones, it is not in the language $TRIPLES$.
8. Thus, we have a contradiction of the pumping lemma and $TRIPLES$ must not be regular.

Answer: The first problem is with step 2. To use the pumping lemma, we need to choose a string that is in the language. We don't know if 1^p is or is not in the $TRIPLES$. A better choice would be to start with 1^{3p} which we do know is in the language.

Step 5 is also broken since it picks a specific choice for y. For our proof to be correct, it needs to work for *all possible* choices for y.

d. (*Bonus*; don't work on this one until you finish the rest of the exam)

In the 2010 annual Latkes (fried potato cakes) v. Hamentash (triangular cookies) debate at MIT, Michael Sipser presented the proof described below (taken from *Faculty fling fake facts in food fight*, The Tech, 26 February 2010, with numbers added):

> Sipser wrapped things up for Team Hamentash with the HamenTheorem, which proves by contradiction that the hamentash is better than the latke. (1) First, the proof assumes latkes are best. (2) Then by obviousness, he claimed that hamentashen are better than nothing, and (3) by first assumption, claimed that nothing is better than latkes. (4) Therefore, Sipser argued that the HamenTheorem proved that hamentashen are better than latkes.

Since everyone knows latkes are better, the proof must be flawed. Explain the flaw in Sipser's proof.

> *Answer:* The best answer to this would be:
>
> Nothing!
>
> There is nothing technically wrong with the proof, but there is a problem with the way Sipser uses the informal English word "Nothing". In step 2, it means "not having anything". In step 3, it means "no one thing".

Problem 5: Leaky PDAs.

Consider a new machine model known as a *LeakyPDA*. A LeakyPDA is similar to a deterministic PDA except that the stack has a limited depth. A LeakyPDA is specified by the 7-tuple $(Q, \Sigma, D, \Gamma, \delta, q_0, F)$ where $D \in \mathcal{N}$ is a natural number giving the maximum stack depth and the other elements have the same meaning as for a deterministic PDA. When the stack contains D elements, if a transition pushes another element on the stack, the bottom element of the stack is removed. That is, the result of following the transition rule,

$$\delta(q_i, a, \epsilon) \rightarrow (q_t, h_p)$$

starting with a stack $\gamma_0\gamma_1, \ldots, \gamma_{D-2}\gamma_{D-1}$ is the stack $h_p\gamma_0\gamma_1, \ldots, \gamma_{D-2}$.

(5.4 / 10) **Revision Opportunity!** Precisely describe the computing power of a LeakyPDA. For full credit, your answer should include a convincing proof supporting your answer.

> *Answer:* You can submit a correct revised answer for this one. Hint: there is a correct three-letter answer to this, but you need more than 3 letters for the proof.

Problem 6: Language Class Differences.

a. (7.9 / 10) If A and B are regular languages, is $A - B$ always a regular language? Recall that the difference of two sets is defined as

$$A - B = \{s : s \in A \wedge x \notin B\}$$

(For full credit, your answer should include a convincing proof supporting your answer.)

Answer: One way to prove this is to use the property that $A - B = A \cap \bar{B}$. Since we already proved the regular languages are closed under intersection and complement, this proves that $A - B$ is regular.

Another way to prove it would be to show how to construct a DFA that recognizes $A - B$:

Since A and B are regular languages, there exists DFA $M_A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ and $M_B = (Q_B, \Sigma, \delta_B, q_{0B}, F_B)$ that recognize A and B respectively.

We can produce a machine M_{A-B} that recognizes the language $A - B$ by simulating both machines and accepting when M_A would accept and M_B would reject.

$M_{A-B} = (Q_A \times Q_B, \Sigma, \delta_{AB}, (q_{0A}, q_{0B}), F_{AB})$ where:

$$\delta_{AB}(q_a, q_b, a) = (\delta_A(q_a, a), \delta_B(q_b, a))$$
$$F_{AB} = \{(q_a, q_b) | q_a \in F_A \wedge q_b \notin F_B\}$$

b. (5.3 / 10) **Revision Opportunity!** If A and B are context-free languages, is $A - B$ always a context-free language? (For full credit, your answer should include a convincing proof supporting your answer.)

Answer: Many people answered by using the $A - B = A \cap \bar{B}$ and claiming that because intersection is not closed for context-free languages this proves that $A - B$ is not a context-free languages. This is not a valid proof. Just because a language can be described as the intersection of two context-free languages doesn't mean that language is not context-free. For example, $A = A \cap A$. Thus, if this proof technique were valid we could use it to prove there are no context-free languages!

A good revised answer would show that $A - B$ is not necessarily a context-free language by identifying two context-free languages whose difference is not context-free (hint: review problem 5c from PS3).