6.045J/18.400J: Automata, Computability and Complexity Prof. Nancy Lynch, Nati Srebro

Quiz 2 Solutions

March 31, 2004 *Susan Hohenberger*

Please write your name in the upper corner of each page.

Name: ______________________________

Problem 1:

True or False (15 points). Full credit will be given for correct answers. If you include justification for your answers, you may obtain partial credit for incorrect answers.

1. True or False: The Post Correspondence Problem for a 3-letter alphabet is decidable.

> **False**; PCP for any finite alphabet can be mapping-reduced to PCP for a 2-letter (binary) alphabet, and PCP is undecidable.

2. True or False: The set A of all finite subsets of $\{0,1\}^*$ (that is, $A = \{S \subseteq \{0,1\}^* | \ S \text{ is finite}\}$) is a countable set. (You can think of A as the set of all *finite* languages.)

> **True**; consider that all finite languages are regular, and each regular language has at least one corresponding DFA. We saw in class that the number of DFAs (or even TMs!) is countably infinite.

3. True or False: Rice's Theorem implies that $\{\langle M\rangle | \ M$ is a Turing machine and M accepts the string 01 and M does not accept the string 10$\}$ is undecidable.

> **True**; this is a non-trivial property of of the language. The description of a machine that accepts only 01 would have this property, while that of a machine that rejects all inputs would not.

4. True or False: The intersection of two Turing-recognizable languages must be Turing-recognizable.

> **True**; given two recognizers R_1, R_2, we build a recognizer for the intersection of their languages, R_3 as follows: on input x, run R_1 and R_2 on x. If R_1 and R_2 both accept x, accept; else loop.

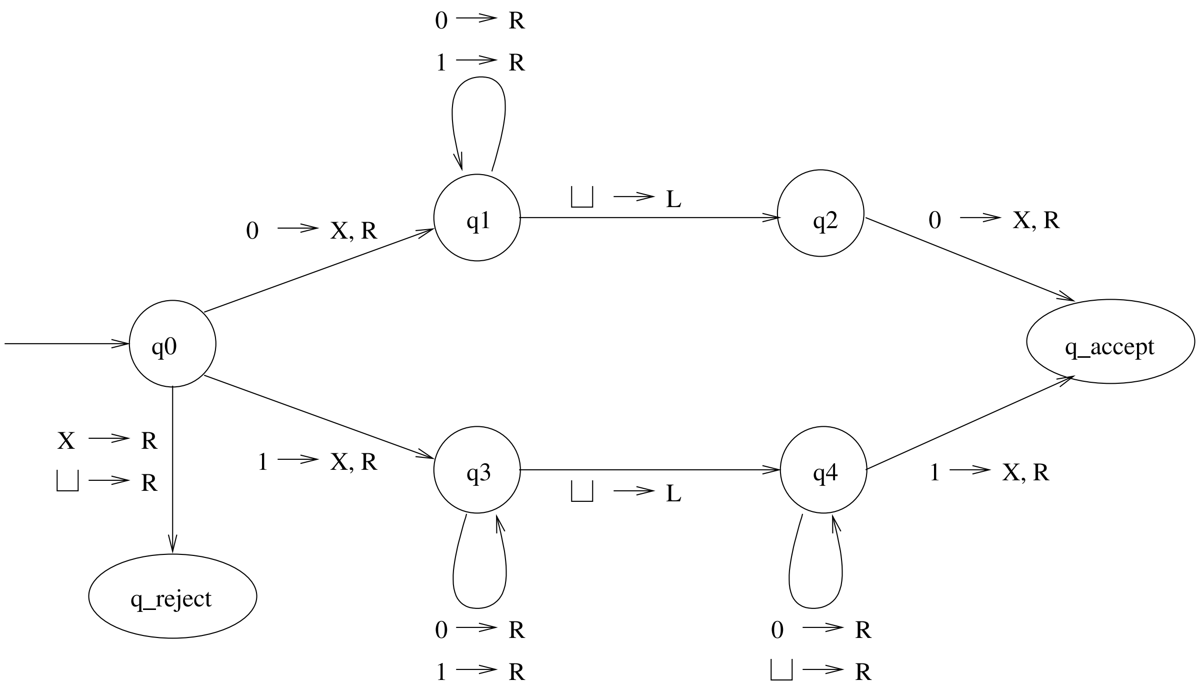
Name: ______________________________

5. True or False: If a language L is recognized by a Turing machine, then L is recognized by a one-stack, one-counter machine (i.e., a machine that has one stack and one counter).

True; from Handout 10, we know that a two-counter machine is equivalent to a TM, and a stack (e.g. over a unary stack alphabet) can simulate a counter.

Name:

Problem 2: **(20 points)** Consider the following formal description of a Turing Machine M, where $Q = \{q_0, q_1, q_2, q_3, q_4, q_{reject}, q_{accept}\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, X, \sqcup\}$. Assume that any unspecified transitions go to q_{reject}.



1. **(6 points)** Write out the accepting computation history of M on input 00.

1. $q_0 00$
2. $X q_1 0$
3. $X 0 q_1 \sqcup$
4. $X q_2 0 \sqcup$
5. $X X q_{accept} \sqcup$

Name: ______________________________

2. **(6 points)** Describe in a few words the behavior of M on input 10.

It loops forever. (Specifically, it takes the $\sqcup$ transition in q_4 repeatedly, walking down the right-side of the infinite tape.

3. **(8 points)** What language does M recognize?

$L(M) = 0(0 \cup 1)^*0 \cup 1(0 \cup 1)^*1$;
all the strings in $\{0, 1\}^*$ that begin and end with the same symbol.

Name: ______________________________

Problem 3: **(25 points)** Let $FIN = \{\langle M\rangle |\ M$ accepts only a finite number of strings$\}$.

1. **(5 points)** Does Rice's Theorem apply to FIN? Why or why not?

Rice's Theorem applies.

1. $|L(M)|$ being finite is a property of the language.

2. This property is non-trivial. The description of a TM that accepts $\emptyset$ is in $L(M)$, while the description of a TM that accepts Σ^* is not.

Prove the following two results about FIN. You may use any results proved in class or in Sipser's book, but if you do, then cite the results explicitly.

2. **(10 points)** FIN is not Turing-recognizable. (Hint: Use mapping reducibility.)

We show that A_{TM} is mapping-reducible to $\overline{FIN}$, and apply Corollary 5.23 from Sipser.
The reducing function f works as follows.

F="On input $\langle M, w\rangle$ where M is a TM and w is a string,
1. Construct TM R using $\langle M, w\rangle$ as below.
2. Output $\langle R\rangle$."

R="On input x,
1. Run M on w.
2. If M accepts, accept.
3. If M rejects, reject."

If M accepts w, $L(R) = \Sigma^*$ and thus $|L(R)| = \infty$. However if M fails to accept w, either by looping forever or rejecting, $L(R) = \emptyset$ and thus $|L(R)| = 0$.

Name: ______

3. **(10 points)** $\overline{FIN}$ is not Turing-recognizable, that is, FIN is not co-recognizable. (Hint: Use mapping reducibility.)

We show that A_{TM} is mapping-reducible to FIN, and apply Corollary 5.23 from Sipser. The reducing function f works as follows.

F="On input $\langle M, w\rangle$ where M is a TM and w is a string,
1. Construct TM R using $\langle M, w\rangle$ as below.
2. Output $\langle R\rangle$."

R="On input x,
1. Run M on w for x steps.
2. If M accepts, reject; otherwise accept."

If M accepts w, then it must do so in a finite number of steps; thus, in this case, only a finite number of inputs will be accepted by R. However if M fails to accept w, either by looping forever or rejecting, $L(R) = \Sigma^*$ and thus $|L(R)| = \infty$.

Name:

Problem 4: **(20 points)** Define a Turing machine M to be *almost-minimal* if there does not exist another Turing machine M' such that $L(M') = L(M)$ and $|\langle M' \rangle| < \frac{1}{2}|\langle M \rangle|$. That is, there is no machine that recognizes the same language and has an encoding whose size is less than half of the size of $\langle M \rangle$.

Prove that there is no enumerator that outputs a set S consisting of an infinite number of almost-minimal machine descriptions. (Hint: Use the recursion theorem.)

Assume for contradiction that such an enumerator E exists. Now consider the following TM:

R="On input x,

1. Obtain, via the recursion theorem, own description $\langle R \rangle$.
2. Run E, letting it enumerate the machines in S until it produces some machine M_i such that $|\langle M_i \rangle| > 2|\langle R \rangle|$.
3. Simulate M_i on x.
4. If M_i accepts, accept; else reject."

Clearly $L(R) = L(M_i)$ and $|\langle R \rangle| < \frac{1}{2}|\langle M_i \rangle|$. So R shows that M_i is not almost-minimal. But that contradicts the assumption that E is producing only almost-minimal machines.

Name: ______________________________

Problem 5: **(20 points)** On your first day on your new job at SmartCompilers.com your boss assigns you the task of adding the following new feature to the company's development suite:

"Analyze the source code of a function, and verify that no array will ever be accessed out of bounds. Output TRUE if the code is safe in this respect, and FALSE if for some inputs, the function will access an array out of bounds."

For example, the following program should fail the verification:

```
void sloppy_code(int k) {
    int array[10];
    for (int i=0; i<k; i++) { array[i]=i; }
}
```

Program $sloppy\_code$ should fail, because on input $k = 15$, the array will be accessed at position 11 (i.e., $array[11] = 11;$), which is outside its declared range of 10.

In your answers, you may assume that the behavior of the function is fully determined by its source code and its inputs (i.e., it does not make any outside calls to libraries or other functions). Also assume for simplicity that the only variables a function uses are (possibly unbounded) integers, and integer arrays.

1. **(5 points)** State your boss' request as a language decision problem.

$SAFE = \{\langle F\rangle |\ F$ is a function that does not permit an array to be accessed out of bounds$\}$

2. **(7 points)** In order to prove to your boss that this task is impossible, show a mapping reduction from some language we know is undecidable to $SAFE$ (Hint: consider using $E_{TM} = \{\langle M\rangle |\ L(M) = \emptyset\}$, or if you prefer you can even use E_{2CM}, the analogous language for 2-counter machines):

We will reduce $\underline{\ E_{2CM}\ }$ to $SAFE$ by mapping each input of the form $\underline{\ \langle Q\rangle\ }$ to input of the form $\langle R\rangle$, where the function R is described below:

R="On input x,

1. Declare two integers a, b and an array $S[5]$.
2. Simulate Q on x using only a and b as counters.
3. If Q accepts x, access S out of bounds (e.g., $S[6] = 0$)."

Name:

3. **(8 points)** Is $SAFE$ recognizable? Co-recognizable? Neither? Both? Explain.

Co-recognizable only.

We can recognize $\overline{SAFE}$ as follows. On input $\langle M \rangle$, simulate M on every string in Σ^* in parallel. If M ever tries to access an array out of bounds, accept.

Since $SAFE$ is undecidable, it cannot also be recognizable.