

Quiz 2: Solutions

March 30, 2005

Vinod Vaikuntanathan

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

Problem	Score
1	
2	
3	
4	
5	
6	
Total	

Name: _____

Problem 1: True or False (20 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: There exists a Turing machine that enumerates a set S of (encodings of) decider Turing machines, such that S includes Turing machines that decide infinitely many different decidable languages.

True. It is easy to enumerate a set of Turing machines $\{T_i\}_{i=1}^{\infty}$ such that for each i , T_i recognizes the language $L_i = \{0^i\}$.

2. True or False: There exists a Turing machine that enumerates a set S of (encodings of) decider Turing machines, such that S includes at least one Turing machine that decides each decidable language.

False. Disprove by diagonalizing (Recall that this was Problem 2 of homework 5).

3. True or False: There exists a Turing machine that enumerates an infinite set S of (encodings of) decider Turing machines, such that every machine that S outputs is “minimal”. (Here “minimal” means that there is no other smaller decider Turing machine that decides the same language.)

False. This is an application of Recursion Theorem. Consider a machine R that first gets its own description (via Recursion Theorem), and proceeds to run the enumerator E till E outputs a machine M' that has a longer description than R . Then it continues to simulate M' . $L(R) = L(M')$, and therefore M' is not minimal. This is a contradiction.

4. True or False: Rice’s Theorem immediately implies that

$$\{\langle M \rangle \mid M \text{ is a Turing machine and } L(M) \subseteq 0^*1^*\}$$

is undecidable.

True. This is a non-trivial property of languages. A machine M_1 that decides the empty language ϕ has the property, but a machine M_2 that decides Σ^* does not.

Name: _____

5. True or False: If L_1 is a decidable language and L_2 is a Turing recognizable language, then $L_1 - L_2$ must be Turing-recognizable.

False. Does not hold if $L_1 = \Sigma^*$ and L_2 is A_{TM} . $L_1 - L_2$ is \bar{A}_{TM} , which is not recognizable.

6. True or False: If L_1 is a Turing recognizable language and L_2 is a decidable language, then $L_1 L_2$ must be Turing-recognizable.

True. L_2 is, in particular, recognizable. Then, this follows from the closure of Turing-recognizable languages under concatenation (this was again a homework problem !!)

7. True or False: A three-dimensional Turing machine is like an ordinary Turing machine except that its “tape” storage consists of a three dimensional “tape”, where each tape cell is a unit cube. In one step, the single tape head can move north, south, east, west, up, or down.
The class of languages recognized by three-dimensional Turing machines is exactly the Turing-recognizable languages.

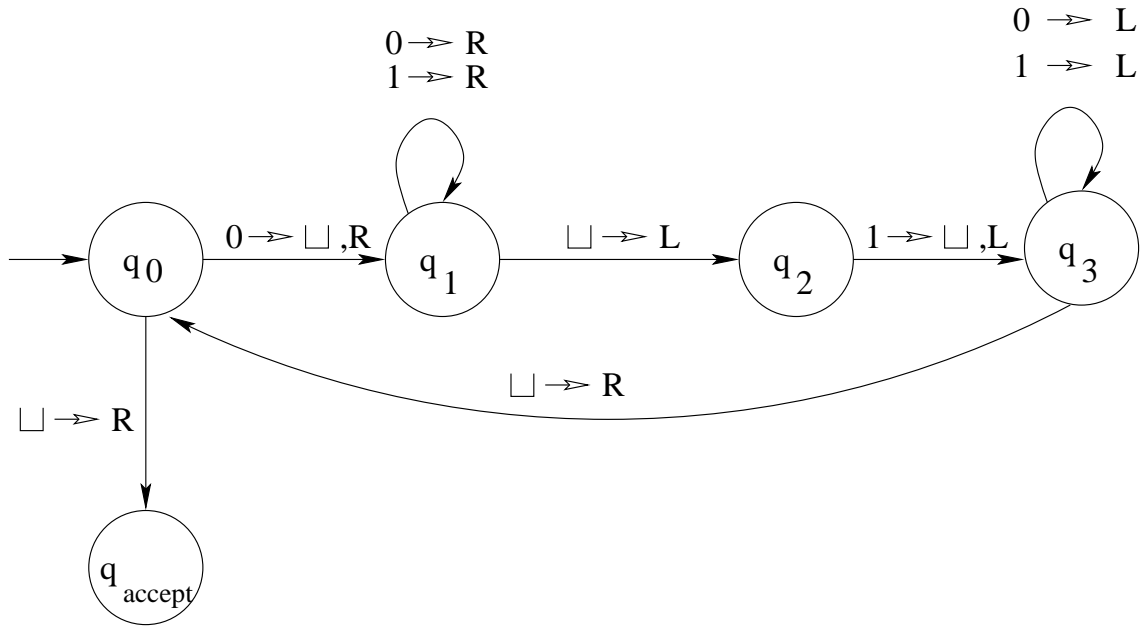
True. One can simulate this by a single-tape machine.

8. True or False: The class of languages recognized by three-stack machines is exactly the Turing recognizable languages.

True. Three stacks is equivalent to two stacks which is as powerful as Turing machines.

Name: _____

Problem 2: (25 points) Consider the following formal description of a Turing Machine M , where $Q = \{q_0, q_1, q_2, q_3, q_{accept}, q_{reject}\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \sqcup\}$. Assume that any unspecified transitions go to q_{reject} .



- (5 points) Write out the accepting computation history of M on input 01, in the form given in class and in Sipser's book. Describe the behavior represented in this computation history in words.

q_0 01
 \sqcup q_1 1
 \sqcup 1 q_1 \sqcup
 \sqcup q_2 1 \sqcup
 q_3 \sqcup \sqcup \sqcup
 \sqcup q_0 \sqcup \sqcup
 \sqcup \sqcup q_{accept} \sqcup

Name: _____

2. (5 points) What language does M recognize? (Give a precise definition.)

$$L = \{0^n 1^n \mid n \geq 0\}.$$

3. (10 points) Give the set of tiles for the *modified* Post Correspondence Problem, for this particular machine M and input 01. Indicate which is the initial tile.

We have started things off by listing the tiles needed for completing the match from the point where an accepting state is encountered. You must define the initial tile and the tiles needed to represent all the moves.

The initial tile : $\begin{pmatrix} \# \\ \#q_001\# \end{pmatrix}$

Tiles for the right moves : $\begin{pmatrix} q_0 0 \\ \sqcup q_1 \end{pmatrix} \begin{pmatrix} q_1 0 \\ 0 q_1 \end{pmatrix} \begin{pmatrix} q_1 1 \\ 1 q_1 \end{pmatrix} \begin{pmatrix} q_3 \sqcup \\ \sqcup q_0 \end{pmatrix} \begin{pmatrix} q_0 \sqcup \\ \sqcup q_{accept} \end{pmatrix}$

Tiles for the left moves :

$$\begin{pmatrix} 0q_1 \sqcup \\ q_2 0 \sqcup \end{pmatrix} \begin{pmatrix} 1q_1 \sqcup \\ q_2 1 \sqcup \end{pmatrix} \begin{pmatrix} \sqcup q_1 \sqcup \\ q_2 \sqcup \sqcup \end{pmatrix} \\ \begin{pmatrix} 0q_2 1 \\ q_3 0 \sqcup \end{pmatrix} \begin{pmatrix} 1q_2 1 \\ q_3 1 \sqcup \end{pmatrix} \begin{pmatrix} \sqcup q_2 1 \\ q_3 \sqcup \sqcup \end{pmatrix} \\ \begin{pmatrix} 0q_3 0 \\ q_3 0 0 \end{pmatrix} \begin{pmatrix} 1q_3 0 \\ q_3 1 0 \end{pmatrix} \begin{pmatrix} \sqcup q_3 0 \\ q_3 \sqcup 0 \end{pmatrix} \\ \begin{pmatrix} 0q_3 1 \\ q_3 0 1 \end{pmatrix} \begin{pmatrix} 1q_3 1 \\ q_3 1 1 \end{pmatrix} \begin{pmatrix} \sqcup q_3 1 \\ q_3 \sqcup 1 \end{pmatrix}$$

The alphabet tiles: $\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} \sqcup \\ \sqcup \end{pmatrix}, \begin{pmatrix} \# \\ \# \end{pmatrix}, \begin{pmatrix} \sqcup \# \\ \sqcup \# \end{pmatrix}$

The “clean-up” tiles : $\begin{pmatrix} 0 q_{accept} \\ q_{accept} \end{pmatrix}, \begin{pmatrix} 1 q_{accept} \\ q_{accept} \end{pmatrix}, \begin{pmatrix} \sqcup q_{accept} \\ q_{accept} \end{pmatrix}, \begin{pmatrix} q_{accept} 0 \\ q_{accept} \end{pmatrix},$

$$\begin{pmatrix} q_{accept} 1 \\ q_{accept} \end{pmatrix}, \begin{pmatrix} q_{accept} \sqcup \\ q_{accept} \end{pmatrix}, \begin{pmatrix} q_{accept} \# \\ \# \end{pmatrix}$$

Name: _____

4. (5 points) Write the accepting computation history you wrote for part (b) twice, one above the other, and mark the boundaries of the MPCP tiles involved in the match. You may skip the part involved in terminating the computation—just mark the tiles up to the first occurrence of the accept state.

$$\begin{array}{cccccccccccccccc} \left(\begin{array}{c} \# \\ \#q_001\# \end{array} \right) & \left(\begin{array}{c} q_00 \\ \sqcup q_1 \end{array} \right) & \left(\begin{array}{c} 1 \\ 1 \end{array} \right) & \left(\begin{array}{c} \# \\ \# \end{array} \right) & \left(\begin{array}{c} \sqcup \\ \sqcup \end{array} \right) & \left(\begin{array}{c} q_11 \\ 1q_1 \end{array} \right) & \left(\begin{array}{c} \# \\ \sqcup\# \end{array} \right) & \left(\begin{array}{c} \sqcup \\ \sqcup \end{array} \right) & \left(\begin{array}{c} 1q_1\sqcup \\ q_21\sqcup \end{array} \right) & \left(\begin{array}{c} \# \\ \# \end{array} \right) \\ \left(\begin{array}{c} \sqcup q_21 \\ q_3\sqcup\sqcup \end{array} \right) & \left(\begin{array}{c} \sqcup \\ \sqcup \end{array} \right) & \left(\begin{array}{c} \# \\ \# \end{array} \right) & \left(\begin{array}{c} q_3\sqcup \\ \sqcup q_0 \end{array} \right) & \left(\begin{array}{c} \sqcup \\ \sqcup \end{array} \right) & \left(\begin{array}{c} \sqcup \\ \sqcup \end{array} \right) & \left(\begin{array}{c} \# \\ \# \end{array} \right) & \left(\begin{array}{c} \sqcup \\ \sqcup \end{array} \right) & \left(\begin{array}{c} q_0\sqcup \\ \sqcup q_{accept} \end{array} \right) & \left(\begin{array}{c} \sqcup \\ \sqcup \end{array} \right) & \left(\begin{array}{c} \# \\ \# \end{array} \right) \end{array}$$

Name: _____

Problem 3: (10 points) Suppose we have a Turing Machine M that, on each input string x , either halts with an output string on its tape or loops forever. Describe briefly how to construct an enumerator Turing machine E that enumerates the outputs produced by M on all the inputs.

This is a simple dove-tail construction. The enumerator E does the following: In the first “phase”, it runs the machine M on the string ϵ for 1 step. In the second “phase”, it runs M on ϵ for 2 steps, and on 0 for 1 step. More generally, in the k^{th} phase, it runs M on the i^{th} string (in the lexicographic order) for $k - i$ steps. This ensures that if M outputs something on input x , it does so in a finite amount of time, and eventually E outputs it too.

Name: _____

Problem 4: (20 points) Let $EVENODD = \{\langle M \rangle \mid M \text{ accepts all strings of even length and does not accept any strings of odd length}\}$

1. (2 points) Does Rice's Theorem apply to $EVENODD$? Why or why not? If it does apply, then what does this imply about $EVENODD$?

Rice's theorem does apply to $EVENODD$, because this is clearly a language property and it is non-trivial. This is because a machine M_1 that accepts exactly the even length strings has this property, and a machine M_2 that accepts nothing, does not.

This means that $EVENODD$ is undecidable.

2. (9 points) Is $EVENODD$ Turing-recognizable? Prove your answer. You may use any results proved in class or in Sipser's book, but if you do, then cite the results explicitly.

No, $EVENODD$ is not Turing recognizable.

We show this by reducing \bar{A}_{TM} to $EVENODD$. Given $\langle M, w \rangle$, we construct a machine $\langle M' \rangle$ such that $\langle M, w \rangle \in \bar{A}_{TM}$ if and only if $\langle M' \rangle \in EVENODD$.

M' first checks if its input is of even length. If so, it accepts.

Else, it runs M on w and accepts if M accepts, and rejects if M rejects.

If M indeed accepts w , then $L(M') = \Sigma^*$ and therefore, $M' \notin EVENODD$.

On the other hand, if M does not accept w , then $L(M')$ is precisely the set of all even length strings.

Therefore, $M' \in EVENODD$.

Name: _____

3. (9 points) Is the complement of *EVENODD* Turing-recognizable? Again, prove your answer. Again, you may use any results proved in class or in the book, but cite them explicitly.

No, the complement of *EVENODD* is not Turing recognizable.

We show this by reducing \bar{A}_{TM} to the complement of *EVENODD*, or equivalently, A_{TM} to *EVENODD*.

Given $\langle M, w \rangle$, we construct a machine $\langle M' \rangle$ such that $\langle M, w \rangle \in A_{TM}$ if and only if $\langle M' \rangle \in \text{EVENODD}$.

M' first checks if its input is of odd length. If so, it rejects.

Else, it runs M on w and accepts if M accepts, and rejects if M rejects.

If M indeed accepts w , then $L(M')$ is precisely the set of all even-length strings and therefore, $M' \in \text{EVENODD}$.

On the other hand, if M does not accept w , then $L(M') = \phi$. Therefore, $M' \notin \text{EVENODD}$.

Name: _____

Problem 5: (15 points) Let L be the following language of Turing machine descriptions:
 $\{\langle M \rangle : M \text{ is a Turing machine with input alphabet } \{0, 1\} \text{ and } M \text{ accepts every string consisting of just zeros (it may accept other strings)}\}$

Prove that L is undecidable using the Recursion Theorem. Do this by filling in the following proof outline:

Suppose for the sake of contradiction that L is decidable.

Let D be the decider for L .

Define a Turing machine R :

R : On input w do:

 Obtain the description of R itself.

This is possible because of the Recursion theorem.

Run D on input $\langle R \rangle$

If D accepts then reject.

If D rejects then accept.

If R accepts all strings consisting of only zeros, then D accepts $\langle R \rangle$,

because D is a decider for L and $\langle R \rangle \in L$.

But this implies that, R rejects all strings (and in particular rejects all strings of the form 0^*),

which is impossible.

On the other hand, if R does not accept all strings consisting of only zeros, then

D rejects $\langle R \rangle$, because D is a decider for L , and $\langle R \rangle \notin L$

But this implies that, R accepts all strings, (and in particular accepts all strings of the form 0^*),

which is also impossible.

Therefore, we have a contradiction, and L cannot be decidable.

Name: _____

Problem 6: (10 points) Consider a new kind of machine, a k -Queue Machine. A k -Queue Machine has the same general structure as a k -Counter Machine or a k -Stack Machine. However, it has k queues for storage instead of k counters or stacks. Initially, each queue q is empty.

It supports the following operations (Let Γ be the alphabet of queue symbols):

1. *enqueue*: takes a symbol in Γ , and adds it to the end of queue q .
2. *dequeue*: removes the symbol at the front of queue q , if q is nonempty. If q is empty, this operation does nothing.
3. *empty*: a boolean, which returns 1 if queue q is currently empty, 0 otherwise.

Briefly outline an argument that the acceptance problem for 2-Queue-Machines is undecidable.

We do this by showing that a k -queue machine can simulate a k -counter machine.

This, in particular, shows that the acceptance problem for a k -counter machine can be reduced to the acceptance problem for a k -queue machine. Since we know that the acceptance problem for a 2-counter machine is undecidable, the acceptance problem for a 2-queue machine is undecidable, too.

We need to show how to simulate the *increment*, *decrement* and *isZero* functions on a counter, using a queue.

increment: is implemented by enqueueing a symbol on the queue.

decrement: is implemented by dequeueing a symbol from the queue.

isZero: is implemented by invoking the function *empty* on the queue, and returning true if and only if *empty* returns true.

It is trivial to see that this correctly implements the functionality of a counter. Since the acceptance problem for a 2-counter machine is undecidable, we conclude that the acceptance problem for a 2-queue machine is undecidable too.

Name:

Scratch Work