

CSc/Math 473

EXAMPLE final exam and Solutions

Do eight (8) problems. All problems have equal weight. Write all answers on this examination, and submit it in the envelope at the end of the exam.

TIME = 2 hours.

You can do ONE additional problem for extra credit. If you do so, circle the extra credit problem you attempted in the table below.

NAME											
Problem	1	2	3	4	5	6	7	8	9	10	Σ
Maximum	10	10	10	10	10	10	10	10	10	10	80+10
Score											

1. Irregularity

Let  $L$  be the following language over  $\Sigma = \{a, b, c\}$ :

$$L = \{ucv \mid u, v \in \{a, b\}^* \text{ and } |u| = |v|\}$$

Show that  $L$  is not a regular language.

**Solution:** If  $L$  is regular, then  $L_1 = L \cap a^*cb^*$  is also regular. But  $L_1 = \{a^n cb^n \mid n \geq 0\}$ , which is known to be irregular (or can be shown to be irregular by an easy pumping argument.) Thus  $L$  cannot be regular. ■

2. True or False

Be very careful to read what is actually asserted. (Short explanations follow each T,F response).

(a) F The Turing-recognizable languages are closed under complementation.

Counterexample is  $A_{TM}$

(b) T The Turing-decidable languages are closed under complementation.

This is a fundamental theorem about decidable sets.

(c) F The following problem is unsolvable: given context-free grammar  $G$ , determine whether or not  $L(G) = \emptyset$ .

The algorithm for testing emptiness was given in a homework assignment and discussed in class. It is the same as the question of whether of start symbol  $S$  is *co-reachable*.

(d) T Any language in  $\Sigma^*$  with a context-free complement is decidable.

Since all CFLs are decidable, their complements are also decidable.

(e) T The set  $\{ \langle M \rangle \mid M \text{ is a TM} \}$  is decidable.

In fact, given our encoding rules, every string over 0 and 1 is the encoding of some TM. In any case, this question is easily decidable using syntactic rules.

(f) F The set  $\{ \langle M \rangle \mid \langle M \rangle \in L(M) \}$  is decidable.

Diagonalization shows that this language cannot be decidable, since its complement can be shown to be undecidable.

(g) T It is decidable, given a TM  $M$  and configurations  $C_1 = uqav$  and  $C_2 = ua'q'v$ , whether  $C_1 \vdash_M C_2$ .

This can be decided by inspection of the transition function of  $M$ .

(h) F It is undecidable, given a TM  $M$  and configurations  $C_1 = uqav$  and  $C_2 = q'\epsilon$ , whether  $C_1 \vdash_M^* C_2$ .

This is undecidable, since  $q'$  could be the accept state, and a decider for this question would decide whether  $q_0w$  leads to an accepting state.

(i) T It is decidable, given a CF grammar  $G$  and a string  $w$ , whether  $w \in L(G)$ .

This decision problem was discussed in class; the algorithm uses facts about the parse tree when the grammar is in Chomsky Normal Form.

(j) T It is decidable, given a right-linear grammar  $G$  and a string  $w$ , whether  $w \in L(G)$ .

A special case of (i).

### 3. Hierarchy

In each part below, describe (using set notation) a specific language that has the given property. A response like " $A_{TM}$ " is not sufficient: you must define the set of strings precisely.

(a) not Turing-recognizable but with a Turing-recognizable complement

$$\overline{A_{TM}} = \{ \langle M, w \rangle \mid w \notin L(M) \}$$

(b) Turing-recognizable but not Turing-decidable

$$A_{TM} = \{ \langle M, w \rangle \mid w \in L(M) \}$$

(c) Context-free but not Regular

$$\{ a^n b^n \mid n \geq 0 \}$$

(d) Turing-recognizable but not Context-free

$$\{ a^n b^n c^n \mid n \geq 0 \}$$

(e) Turing-recognizable with a non-Turing-recognizable complement

$$A_{TM} = \{ \langle M, w \rangle \mid w \in L(M) \}$$

### 4. Relational Calculus

Let  $R \subseteq A \times A$ .  $R$  is a *quasi-order* if it is both irreflexive ( $R \subseteq \overline{I_A}$ ) and transitive ( $R \circ R \subseteq R$ ).

Prove:  $R$  is a quasi-order if and only if  $R \cap R^{-1} = \emptyset$  and  $R = R^+$ .

**Solution:** If  $R$  is a quasi-order, then because  $R^{-1} \subseteq \overline{(I_A)^{-1}}$ , it follows that  $R \cap R^{-1} = \emptyset$ . Transitivity ( $R \circ R \subseteq R$ ) implies  $(\forall i \geq 0) R^i \subseteq R$  and hence  $R = R^+$ .

In the other direction, if  $R \cap R^{-1} = \emptyset$  then  $(a, a) \notin R$  for all  $a$ , and so  $R$  is irreflexive. And if  $R = R^+$ , then  $R \cup R^2 \cup \dots \subseteq R$ , showing that  $R^2 \subseteq R$ , which is transitivity. ■

## 5. Determination

Construct a DFA equivalent to the NFA given below, using the Rabin-Scott algorithm. Show any intermediate work. Label all the DFA states appropriately. Here  $\epsilon$  denotes the empty string.

**Solution:** See NFA and DFA figures in file [www.cs.arizona.edu/classes/cs473/fall107/prob5fig.pdf](http://www.cs.arizona.edu/classes/cs473/fall107/prob5fig.pdf)

## 6. Unsolvable Problem

A TM is said to be *tidy* if, whenever it halts for some input, it halts with its tape completely blank.

Show that the following problem is unsolvable:

*Given:* A Turing Machine  $M$

*Question:* Is  $M$  a tidy TM?

**Solution:** Reduce  $A_{TM}$  to the (Un)Tidy problem. We know we can assume that if a TM  $M$  halts, it halts only in state  $q_{accept}$  (build in transitions from other "blocked" transitions to a "loop" state). Given  $\langle M, w \rangle$ , the reduction function alters  $M$  (using  $w$ ) to a new machine  $M' = C(\langle M, w \rangle)$  so that, when  $M$  would halt on  $w$  by entering  $q_{accept}$ , then  $M'$  mimics it—but before halting in  $q_{accept}$ ,  $M'$  writes some garbage on the tape and then halts. So  $M$  is untidy iff  $M$  accepts  $w$ . (If  $M'$  does not halt, then *by definition* it is tidy!). Now for the reduction. If we assume there is a decider for tidiness (and hence untidiness), we can use that "untidiness" decider on  $M'$ , and thus decide whether  $M$  accepts  $w$ . So we can construct a decider for  $A_{TM}$  assuming a decider for (Un)tidiness. We conclude there can be no decider for tidiness (or untidiness).

## 7. Incrementing by TM

Design a TM with tape alphabet  $\Sigma = \{0, 1, \#\}$  that, given a binary representation of a number, computes the binary representation of its successor. For example, if  $M$  is started in configuration

#s110101111#

then it halts in configuration

#110110000h#

(Here for emphasis the character scanned by the read/write head is designated by an underscore.)

You may give either a detailed TM diagram or an informal description of the TM at an appropriate level of detail. Comment your code to make clear what is intended.

**Solution:** The TM scans right until it reaches #, and backs up one cell. If this cell contains 0, then it replaces it with 1 and moves R to halt over #.

If the cell contains 1, it overwrites 0, and the head shifts left, overwriting 1's with 0's until the first 0 is encountered, which is overwritten by 1 and the read/write head is then re-positioned to # on the right.

If no 0 is encountered in scanning left, then # is encountered. If this happens, all the 0,1 data to the right of # is shifted right, and 1 is inserted in the cell after #. Then the head is repositioned right to the first # on the right.

## 8. Diagonalization

In this problem, you are asked to prove that the "diagonal language"

$$D = \{ \langle M \rangle \mid \langle M \rangle \notin L(M) \}$$

cannot be a Turing-recognizable language. (This is the language consisting of all TMs that do not recognize themselves!)

(a) Proof by contradiction. Assume  $D$  is Turing-recognizable. Then there must be some particular TM that recognizes  $D$ . Call it  $M^*$ . Write down in symbols what the foregoing fact means in terms of  $L(M)$  and  $\langle M \rangle$  (this part has been done for you.)

$$D = L(M^*) = \{ \langle M \rangle \mid \langle M \rangle \notin L(M) \}$$

(b) Using the definition of  $D$ , write out exactly what is in the set  $L(M^*)$ :

$$L(M^*) = \{ \langle M \rangle \mid$$

$$L(M^*) = \{ \langle M \rangle \mid \langle M \rangle \notin L(M) \}$$

(c) Is  $\langle M^* \rangle$  a member of  $L(M^*)$ ? Don't answer yet. Let us explore this. First suppose that  $\langle M^* \rangle \in L(M^*)$ . What can we then conclude from the definition of  $L(M^*)$  in part (b)?

$$\langle M^* \rangle \notin L(M^*)$$

(d) Next suppose that  $\langle M^* \rangle \notin L(M^*)$ . What can we now conclude from the definition of  $L(M^*)$  in part (b)?

$$\langle M^* \rangle \in L(M^*)$$

(e) Parts (c) and (d) together form a blatant contradiction. Explain why.

They imply that

$$\langle M^* \rangle \in L(M^*) \Leftrightarrow \langle M^* \rangle \notin L(M^*)$$

which is obviously "absurd" (a patent contradiction.)

(f) Given that a contradiction has been derived, what assumption that we made earlier *must* be false?

The only assumption we made was to assume that  $D$  was Turing-recognizable. So that assumption must be false.

(g) What is the overall conclusion that can be derived from this *reductio ad absurdum* argument?

That  $D$  must not be Turing-recognizable.

## 9. State Diagram

Construct the state diagram of a DFA that accepts

$$L = (00 + 1)^* \cap ((000)^*(11)^*)$$

Explain each step in your construction in a short phrase that allows the reader to understand how your construction works, and why it is correct.

**Solution:** One can solve this problem by (a) constructing DFA for each of the regular expressions and (b) performing the "cross-product" construction on the two DFAs to get a DFA for the intersection set of strings.

The easier way is to solve this by using your knowledge of regular expressions. The expression  $((000)^*(11)^*)$  says that words in the intersection must be of the form  $(000)^m(11)^n$  for  $m, n \geq 0$ , with all 0s preceding all 1s. The expression  $(00 + 1)^*$  says that words in the intersection must further have a number of 0s that is even (and there is no restriction on the length of the run of 1s). So the final description of the intersection language  $L$  is  $L = \{ 0^{6i} 1^{2j} \mid i, j \geq 0 \}$ .

It is easy to write down a DFA from this (6 states for 0s and 2 states for 1s) and this will not be given here.

## 10. Decision Problem

Is there an algorithm to decide, given a Turing-recognizable set  $L$ , whether or not  $L \cap (01)^* \neq \emptyset$ ? If so describe the algorithm. If not, prove there is no such algorithm.

**Solution:** Given the language  $L = L(M)$  recognized by TM  $M$ , there is no algorithm to decide whether  $L \cap (01)^* \neq \emptyset$ . The property  $P(L) \equiv [L \cap (01)^* \neq \emptyset]$  is a property of the language recognized by the TM  $M$ . It is a non-trivial property, since the property holds for  $L = (01)^*$  and does not hold for  $L = \emptyset$ . So from Rice's Theorem, the property  $P$  is undecidable for Turing-recognizable sets.