

15-354: Midterm

October 14, 2008

Name:

Andrew ID:

Instructions

- Fill in the box above with your name and your Andrew ID. **Do it, now!**
- Clearly mark your answers in the allocated space. If need be, use the back of a page for scratch space. If you have made a mess, cross out the invalid parts of your solution, and circle the ones that should be graded.
- Scan the test first to make sure that none of the 12 pages are missing. The problems are of varying difficulty and are not necessarily sorted according to increasing difficulty. You might wish to pick off the easy ones first.
- You have 80 minutes. Good luck.

1	20	
2	20	
3	20	
4	20	
5	20	
Total	100	

Problem 1: Integer Register Machine (20 pts.)

According to our definition of a register machine, a register can only hold natural numbers. Suppose we allow registers to hold arbitrary integers instead, and we have instructions for unconditional increment, unconditional decrement and test for 0.

- **inc r k**: increment register R_r , goto k ,
- **dec r k**: decrement register R_r , goto k ,
- **zero r k l**: if register R_r is 0, goto k , otherwise goto l .

Call such a contraption an *integer register machine (IRM)*.

- A. Show that a function $f : \mathbb{N} \rightarrow \mathbb{N}$ that is RM-computable is also IRM-computable.

B. Show that a function $f : \mathbb{N} \rightarrow \mathbb{N}$ that is IRM-computable is also RM-computable.

Problem 2: Turing Machines (20 pts.)

Some finite state machines require huge state sets (e.g. in natural language processing or verification), which fact causes endless headaches in applications. How about Turing machines?

To be more precise, fix some tape alphabet Γ . Is it true that in order to implement some arbitrary computable function f on a Turing machine using alphabet Γ one needs arbitrarily many states? Explain your answer.

Problem 3: Word Shuffle (20 pts.)

The *word shuffle* operation, in symbols \parallel , is a map from $\Sigma^* \times \Sigma^*$ to $\mathfrak{P}(\Sigma^*)$ defined by

$$\begin{aligned}\varepsilon \parallel y = y \parallel \varepsilon &= \{y\} \\ xa \parallel yb &= (x \parallel yb) a \cup (xa \parallel y) b.\end{aligned}$$

As usual, we can extend the operation to languages by

$$K \parallel L = \bigcup \{x \parallel y \mid x \in K, y \in L\}$$

For example,

$$aa \parallel bbb = \{aabb, ababb, abbab, abbba, baabb, babab, babba, bbaab, bbaba, bbbaa\}$$

Also, $a^* \parallel b^* = \{a, b\}^*$.

- A. Let $M_1 = \langle Q_1, \Sigma, \tau_1; I_1, F_1 \rangle$ and $M_2 = \langle Q_2, \Sigma, \tau_2; I_2, F_2 \rangle$ be two NFAs accepting regular languages K and L , respectively. Show how to construct a finite state machine $M = \langle Q, \Sigma, \tau; I, F \rangle$ that accepts the shuffle language $K \parallel L$. Explain what you are doing.

$Q =$

$\tau =$

$I =$

$F =$

B. Now suppose that M_1 and M_2 are in fact DFAs. Give an upper bound on the size of a DFA that accepts $K \parallel L$ in terms of n_1 and n_2 , the sizes of M_1 and M_2 , respectively.

Problem 4: Wurzelbrunft and Ochsenfiesl vs. Collatz (20 pts.)

Wurzelbrunft and Ochsenfiesl are two grad students at a little known university. They both became fascinated by the Collatz problem as undergraduates. Their approach is to study the complexity of the *Collatz set*

$$S = \{x \in \mathbb{N} \mid \text{orbit of } x \text{ contains } 1\}.$$

Wurzelbrunft thinks he has a proof that S is decidable. He also claims that his result, together with the well-known fact that S is infinite, immediately implies the Collatz conjecture. Ochsenfiesl, on the other hand, thinks he has a proof that S is undecidable. He claims his result implies that the Collatz conjecture is false.

If you were their PhD advisor, what professional, well-reasoned advice would you give to them? A rant won't do, you need to provide arguments.

A. Wurzelbrunft:

B. Ochsenfiesl:

Problem 5: Computing Transients and Periods (20 pts.)

For the following suppose we are given a C program that computes a function $f : A \rightarrow A$ where $A = \{0, 1, \dots, 10^{15} - 1\}$. For simplicity let's assume that the computation takes one step.

If you write pseudo-code to describe your algorithms below, make sure to provide ample comments; no credit otherwise. You can use any auxiliary data structure you like, just say clearly what the data structure is. Write t for the transient of point, and p for the period.

- A. Suppose that the transients of all points under f are at most 3 (three). Give a fast and memory efficient algorithm to compute the transient and period of a point $x \in A$. State the running time of your algorithm in terms of t and p .

- B. Now suppose that the periods of all points under f are at most 3 (three). Give a fast and memory efficient algorithm to compute the transient and period of a point $x \in A$. State the running time of your algorithm in terms of t and p .