UVa - cs302: Theory of Computation Spring 2008

Problem Set 5 - Undecidability

Due: Tuesday, 1 April (2:02pm)

This problem set focuses on material from Classes 14-18 (through 28 March) and Sipser's book Chapter 4 and Chapter 5 (through Section 5.1). Answer all 6 questions. For full credit, answers must be concise, clear, and convincing, not just correct. Please **staple** your answer sheets before class.

Honor Policy (changed from PS4). As with previous problem sets, you may discuss and work on the problems with anyone you want. After your discussions, you must destroy any notes from the meetings and write up your own solutions based on your own understanding. Unlike in previous problem sets, this policy also applies to notes from problem-solving sessions that pertain to questions on the problem sets. You may preserve notes from these meetings that pertain to other problems or general questions, but should not use notes from these meetings that include answers to specific questions from the problem set.

Problem 1: Random Access Memory. Random access memory (misnamed, since it is not at all random) allows a program to directly reference specified memory locations. Assume each memory location can store a single byte (8 bits) value. Show that a Turing Machine can simulate random access memory. Your machine should be able to simulate these two instructions:

1. **store** $< location >$ — write the value represented by the current square on the tape into location $< location >$. The $< location >$ is any 32-bit integer, and $< value >$ is any 8-bit value (represented by a single square on the tape).
2. **load** $< location >$ — read the value in the location $< location >$ and write it onto the current square on the tape. At the end of a load instruction, the square under the tape head should contain the read value. The read value should be the last value that was stored in $< location >$ (using a **store** instruction), or 0 if no value has been stored in $< location >$.

a. Provide an implementation-level description of a Turing Machine that can simulate the **load** and **store** instructions. Your description should explain how you represent memory on the tape and provide implementation-level descriptions of how you simulate the two instructions.

b. Is the programming language consisting of just the **load** and **store** instructions above a *universal programming language*? (That is, is it possible to express all possible algorithms using this language.) If it is, prove it (by explaining how you could implement a universal Turing Machine using the **load/store** language. If it is not, explain convincingly why not, and describe the simplest modifications needed to make the language a universal programming language.

Problem 2: Language Sizes. Consider the language,

$$BIGGER_{DFA} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } |L(A)| > |L(B)|\}$$

The notation $|L(M)|$ means the size of the language described by the machine M. The size of a language is the number of strings in the language. For purposes of this question, you should assume the definitions about set sizes from Definition 4.12.

Is $BIGGER_{DFA}$ decidable? Either prove that it is decidable (for example, by providing a high-level description of a Turing Machine that can decide it), or prove that it is undecidable (for example, but showing that a known undecidable problem can be reduced to it).

Problem 3: Closure Properties. For each part, provide a clear **yes** or **no** answer, and support your answer with a brief and convincing proof.

a. If A is a Turing-recognizable language, is the complement of A a Turing-recognizable language?

b. If A is a Turing-decidable language, is the complement of A a Turing-decidable language?

c. If A and B are Turing-recognizable languages, is $A \cap B$ a Turing-recognizable language?

d. If A and B are Turing-decidable language, is $A \cap B$ a Turing-decidable language?

Problem 4: Undecidability. Prove that each of the following languages is undecidable. (Hint: show that you can reduce a known undecidable problem to the problem of deciding the given language.)

a. $L_{INF} = \{< M > | M \text{ describes a TM that accepts infinitely many strings}\}$

b. $L_{HelloWorld} = \{J | J \text{ is a Java program that prints out "Hello World"}\}$

Problem 5: Unmodifiable-Input Turing Machine. (Based on a question by Ron Rivest.) Consider a one-tape Turing Machine that is identical to a regular Turing machine except the input may not be overwritten. That is, the symbol in any square that is non-blank in the initial configuration must never change. Otherwise, the machine may read and write to the rest of the tape with no constraints (beyond those that apply to a regular Turing Machine).

$HALT_{UTM} = \{< M, w > | M$ is an unmodifiable-input TM and M halts on input $w\}$

a. What is the set of languages that can be recognized by an unmodifiable-input TM? (Support your answer with a convincing argument.)

b. Is $HALT_{UTM}$ decidable (by a regular TM)? (Support your answer with a convincing proof.)

Problem 6: Minds and Machines. Many people find the suggestion that a human mind is no more powerful than a Turing Machine to be disturbing, but there appear to be strong arguments supporting this position. For example, consider this argument:

> The brain is a collection of 100 billion neurons. Each neuron is a cell that has inputs (known as *dendrites*) and outputs (synapses that emit neurotrasmitter chemicals). The output depends on the inputs in a deterministic way that could be simulated by a Turing Machine. The connections between neurons could also be simulated by a Turing Machine. Since all components of the brain could be simulated by a Turing Machine, the brain itself could be simulated by a Turing Machine. Hence, a human mind is no more powerful than a Turing Machine.

Write a short essay that counters this argument (although many books have been written on this question, you should limit your response to no more than one page). If you reject the premise of this question either because you do not find it disturbing to think of your mind as a Turing Machine, or you feel that the only way to counter this argument is to resort to supernatural (e.g., religious) notions, you may replace this question with Sipser's Problem 5.13.