

CS 159: Homework 1
Professor John Savage
Assigned: January 30, 2009,
Due: Monday, February 9, 2009

Question 1

The halting problem is defined by the language $\{ \langle [M], x \rangle \}$ where $[M]$ is a description of a Turing machine M and x is an input to that machine such that $[M]$ halts on x . $HALT$ is undecidable, meaning no Turing machine exists that accepts only the strings in $HALT$.

A Turing machine is said to “require k tape cells” if, when started on a blank tape, its head eventually reaches the k^{th} cell. The **busy beaver problem** is defined as follows:

- Given an integer n , output an integer k , in binary, such that there is no Turing machine with n or fewer states that both halts when started on a blank tape and requires at least k tape cells.

Use the fact that $HALT$ is undecidable to prove that no TM can solve this problem for all n .

Question 2

A) In class we defined the complexity class **NP** as follows:

A language L is in **NP** if and only if there exists some Turing Machine M_L such that

1. if $x \in L$, there exists a string u such that $M_L(x, u)$ accepts in polynomial time,
2. if $x \notin L$ there is no u such that $M_L(x, u)$ accepts.

In this definition, M_L need only halt when $M_L(x, u)$ accepts. Suppose we add the requirement that M_L halts in polynomial time on all inputs. Prove that this additional requirement does not change the set of languages in **NP**.

B) Later we defined the complexity class **coNP** as follows:

A language coL is in **coNP** if and only if the complement of that language, L is in **NP**.

Define **coNP** directly using Turing Machines and certificates.

Question 3

An instance of 3COLORING is a graph $G(V, E)$. A graph is a “yes” instance of 3COLORING if and only if each vertex can be labeled either “red”, “green” or “blue” such that no two adjacent vertices receive the same color.

In class we give a reduction from 3SAT to INDEPENDENT SET (see slide 10). Using a similar construction, reduce 3SAT to 3COLORING and prove that 3COLORING is **NP**-Complete.

HINT: As with INDEPENDENT SET, your reduction should involve a triangle for each clause. It should also involve a pair of connected nodes for each x_i and \bar{x}_i , all of which are connected to a single additional vertex.

Question 4 (OPTIONAL)

In class (Lecture 2, slide 10) we stated that DFSMs and NFSMs recognize the same sets of languages. Given a NFSM with states $q_i \in Q$, it is possible to construct a DFSM with states $q'_i \in 2^Q$. A full explanation of this can be found in Chapter 3 of *Models of Computation*.

Any language recognized by an FSM can be represented as a “regular expression”. Given an alphabet, \mathcal{A} , a regular expression R is either a symbol in \mathcal{A} , the concatenation of two regular expressions (denoted $R = R_1 R_2$), the union of two regular expressions (denoted $R = R_1 + R_2$), or the closure of a regular expression (denoted $R = R_1^*$). The meaning of these operations should be clear from the following examples, in which $\mathcal{A} = \{0, 1\}$.

- $R = 0 + 10$: The language consisting of the binary strings “0”, or “10”.
- $R = (0 + 1)^* 10$: The language consisting of all binary strings ending in “10”.
- $R = 1(00 + 11)^* 1 + 000$: The language consisting of “000” or a binary string that begins and ends with a 1, and otherwise contains only pairs of 0’s and 1’s.

1. Write an NFSM that recognizes each of the above examples. Please describe your FSM using a state transition diagram and be sure to shade in any accept states. When you are done, briefly described a systematic approach for converting a regular expressions to NFSMs.
2. Regular expressions are used in many programming languages, but they are often described using concatenation, union and closure, along with additional operations. One common additional operation is complementation where R^c denotes the set of strings not described by R . Argue that complementation is convenient, but not necessary. Try to keep your argument simple by first noting the equivalence between FSMs and regular expressions.

Question 5 (OPTIONAL)

In class, we noted that a single tape Turing machine is no more powerful than a multitape Turing machine. Describe how a two-tape turing machine, M , can be simulated with a single-tape Turing machine, M' , over a larger alphabet. Now describe how M' can be converted to a single-tape Turing machine, M'' , that operates over the alphabet $\{0, 1\}$.

Both of your constructions should be clear, but not overly detailed. You should definitely not provide a state transition diagram. In both cases, use should also use big-O notation to bound the time and space overhead associated with your constructions. In other words, given the time, T , and space, S , required by M , bound the time and space required by M' and M'' .