Harvard University Extension School
Computer Science E-121

Problem Set 6

Due November 4, 2016 at 11:59pm

Problem set by Walter Thornton

Collaboration Statement: I completed this work alone and only with course materials

When writing proofs, avoid the inclusion of redundant or irrelevant information. Sometimes these details can obscure the argument and we will take off half points when these issues arise.

PROBLEM 1 (6 points, suggested length of 1/2 page)

Let $L = \{\langle D, k\rangle$: if $k \in \mathbb{N}, D$ is a DFA that accepts exactly k strings; if $k = -1, D$ is a DFA that accepts an infinite number of strings$\}$. Show that L is recursive.

Hint: Show how to find a p such that if D accepts any string of length at least p, then D accepts infinitely many strings.

Solution.

We set p the number of states in the DFA. If D accepts any strings over length p, then by the pigeon hole principle (pset1s 7b) D accepts an infinite number of strings as it necessarily contains a cycle.
Construct a decider D for L
D = on input $\langle D, k\rangle$
1. Count number of states in DFA D, set to p.
2. If length of string is greater then p, then if $k = -1$ accept.
3. Else, Simulate D on all strings of length less than or equal to p. If number of accepting strings is k, accept, else reject.
Since D always accepts or rejects on k, L is recursive.

PROBLEM 2 (3+3 points, suggested length of 1/4 page each)

For each of the following languages, determine whether it is decidable or undecidable. If it is undecidable, determine whether it recognizable, co-recognizable, or neither. In each case, M is an arbitrary Turing machine.
(A) $L_k = \{\langle M\rangle : L(M)$ contains a string of length nk, $k > 1, n > 0$ (some multiple of k)$\}$
(B) $\{\langle M, w\rangle : M$ eventually visits the 10th square twice on $w\}$

Solution.

(A) decidable
a DFA can be constructed to accept multiples of k. A decider can be built to simulate the DFA

and accept or rejects as the DFA does.
(B) undecidable, recognizable
if the machine reaches the same configuration twice, then it may loop if w is not in M, and may not halt

The following definition will be useful for the questions 3 and 4. A function $f : \Sigma^* \to \Sigma^*$ is *computable* if there is a Turing machine M that, on every input $w \in \Sigma^*$, halts with the tape containing just the string $f(w)$ (followed by blank symbols). In this case, we say that M *computes* the function f.

PROBLEM 3
(4 points, suggested length of no more than six or seven lines. Redundant information will be penalized.)

Let $\Sigma = \{a, b\}$. For every set $A \subseteq \Sigma^*$ there is a function $\alpha_A : \Sigma^* \to \{a, b\}$ defined by $\alpha_A(x) = a \leftrightarrow x \in A$. Prove that L is decidable if and only if α_L is computable.

Solution.

If α_A is computable then some TM M, on every input $x \in$, halts with $\alpha_A(x)$ on its tape. From the contrapositive, we know that if $x \ni A$ then $\alpha_A(x) = b$ N is a decider for M
N =on input x:
1. Compute $\alpha_A(x)$
2. Run M on $\alpha_A(x)$
Since M always halts with outputs a or b, then L is decidable.

PROBLEM 4 (4 points, suggested length of 1/3 page)

A *homomorphism* is defined as follows:

> Let Σ and Δ be alphabets. Consider a function h from Σ to Δ^*. Extend h to a function from Σ^* to Δ^* by applying it to each symbol in its input. More formally:
>
> $$\begin{aligned} h(\varepsilon) &= \varepsilon \\ h(w\sigma) &= h(w)h(\sigma), \text{ for any } w \in \Sigma^*, \sigma \in \Sigma \end{aligned}$$
>
> Any function $h : \Sigma^* \to \Delta^*$ defined in this way from a function $h : \Sigma \to \Delta^*$ is called a **homomorphism**.

Note that homomorphisms can "erase": $h(w)$ may be ε, even though w is not.
Prove that the Turing-recognizable languages are closed under *homomorphism*.

Solution.
Imagine an enumerator E_1 that prints every w of L
Have another enumerator E_2, using the enumeration of E_1 as input,
that prints every $h(w)$ of $h(L)$

$h(L)$ is the homomorphism of L, since this can be derived through enumeration,
and a language is re iff some enumerator enumerates it,
then both $h(L)$ and L are re and are closed under homomorphism.

PROBLEM 5 (3+3 points, suggested length of 1/2 page)

Consider the following modifications to the definition of a Turing Machine. Specify the class of decidable languages under each model and concisely justify your answers. Constructions may be described informally so long as it's clear they could be implemented.

(A) A TM_A is defined with an additional natural number k. The machine operates on a read-only input tape and a writable work tape, with the restrictions that the input tape head cannot move left, and the work tape contains only k cells.

(B) A TM_B is defined with an additional set S of natural numbers, two special states q_{lookup} and q_{yes}, and a second tape with alphabet $\{0, 1\}$ for writing nonnegative binary integers. When the machine transitions to state q_{lookup}, it is automatically sent to q_{yes} (ignoring transitions from q_{lookup}) if the second tape contains a number i that is in S.

As an example, a TM_B can determine if the number five if in its set S by writing the binary representation of five (101) to the second tape and transitioning to q_{lookup}. If five is in S, the machine will take its next transition from q_{yes} rather than q_{lookup}.

Solution.

(A)
TM_A is actually an LBA, therefore it is decidable.(theorem 5.9 pg 222)
The read only input tape is irrelevent since the input can be trasnferred to the work tape. Since the work tape is bounded by k, it is a LBA. We can simulate this TM for limited time qkg^q, if it does not accept by then, reject.
(B) undecidable, r.e. because we knoe nothing about what happens in TM_B after q_{lookup}if i is not in S