Assignment 4

Due Friday April 7th, 2016

1. [10 marks] Short answers about Turing machines

Answer each of the following questions in *one or two* sentences.

(a) [2 marks] Explain the difference between a Turing recognizable and a Turing decidable languages.

(b) [2 marks] Explain the Halting problem and its significance.

(c) [2 marks] Consider the set of languages recognizable by a determnistic Turing Machine. Now consider a multi-tape Turing machine, that is, a Turing machine with multiple taples. Could a multi-tape Turing machine possibly recognize more languages than a single-tape Turing Machine? Why or why not?

(d) [2 marks] Under what circumstance does a non-deterministic Turing machine output `accept`?

(e) [2 marks] In class we mentioned that the set of all possible Turing Machines is countably infinite, and that the set of all languages is uncountably infinite. Use these facts to reason why there *must* be languages that cannot be recognized by a Turing machine.

2. [10 marks] Computation tree of a non-deterministic Turing Machine

The following transitions implicitly define a non-deterministic Turing machine (with the alphabet $\Sigma = \{r, s, t\}$):

$\delta(q_0, r) = (q_1, a, R)$
$\delta(q_0, r) = (q_2, a, R)$
$\delta(q_0, s) = (q_1, b, R)$
$\delta(q_0, t) = (q_2, c, R)$
$\delta(q_1, r) = (q_0, a, R)$
$\delta(q_1, s) = (q_1, b, R)$
$\delta(q_1, t) = (q_0, c, R)$
$\delta(q_2, r) = (q_2, a, R)$
$\delta(q_2, s) = (q_3, b, R)$
$\delta(q_2, t) = (q_1, c, R)$

where q_0 is the start state and q_3 is the accepting state.

(a) [5 marks] Create a state diagram to represent the Turing machine.

(b) [5 marks] Given input `rstrsr` , list the final tape contents for each computational path, and indicate which paths accept.

3. [6 marks] Complexity Classes

For each of the complexity classes covered in the lectures, i.e., **P**, **NP**, **NP**-complete, **NP**-hard, **BPP** and **BQP**, state whether the problems below are known to be in that class or not. For each positive answer (i.e., *yes, problem ____ is in class ____*.), give the name of an algorithm from that class that solves it:

(a) [3 marks] **PRIMES**, the problem of determining if a given integer is a prime number.

(b) [3 marks] **FACTOR**, the problem of determining if an integer contains a factor less than some bound b.

4. [4 marks] Boolean Satisfiability

(a) [2 marks] **Boolean Satisfiability (SAT)**. Consider the following boolean equation:

$$((x_1 \land x_2) \lor (x_3 \land \neg x_4)) \land (\neg x_1 \lor \neg x_2) \land (x_2 \lor \neg x_3)$$

Is this equation satisfiable? If so, assign the variables to satisfy the equation. If not, explain why not.

(b) [2 marks] Recall **SAT** was the first problem proven to be **NP**-complete (by Cook and Levin). What would the consequence be of finding an efficient algorithm to solve **SAT**?

5. [10 marks] NP-Completeness

Let S be a set of integers. Recall SUBSETSUM is the problem of deciding whether there is some subset of S which sums to zero. Now consider the related problem PARTITION which asks whether S can be partitioned (i.e., split) into two subsets such that they have the *same* sum. For example, $\{1, 2, 3, 4\}$ contains such a partition: $\{1, 4\}$ and $\{2, 3\}$.

Given that SUBSETSUM is **NP**-complete, prove PARTITION is **NP**-complete. **Hint**: Suppose the elements of a set S sum to some integer s. Define a new set $S' = S \cup \{-s\}$ and use it in your proof.

6. [10 marks] Undecidability

Recall the Halting problem:

$$\text{HALT} = \{\langle M, w\rangle : M \text{ halts on string } w\}.$$

We already know this problem is undecidable, but maybe it's still possible to build a decider for a simpler problem, like deciding if a machine M will accept a *specific* input, e.g., '`a`'. As we will find out, this language is undecidable *as well*. Let:

$$\text{L} = \{\langle M\rangle : M \text{ accepts the string 'a'}\}.$$

Prove L is undecidable.

Hint: Toward a contradiction assume there exists a Turing machine C that decides L. It accepts the description of a Turing machine M as input and returns `True` if the machine accepts `a`, and returns `False` otherwise. Now consider the following machine:

```
A(<M,w>):
    Construct a new Turing machine B with the following behavior:
        B(x):
            Simulate M on w
            Halt and output accept

    return <B>  //Return a string describing Turing machine B.
```

Now write an algorithm `Halt-solver` that accepts a string `<M,w>` and outputs accepts if `M` halts on input `w` and outputs rejects otherwise. By making calls to A and C, show how `Halt-solver` can decide the halting problem.