6.045 Pset 3: “The Gödel-Turing Mindblower”

Assigned: Wednesday, February 23, 2011
Due: Wednesday, March 9, 2011

To facilitate grading, remember to solve each problem on a separate sheet of paper! Also remember to write your name on each sheet.

1. **Decidable and Recognizable**

   (a) Recall that a language L is *decidable* if there exists a Turing machine M such that $M(x)$ accepts for all $x \in L$ and $M(x)$ rejects for all $x \notin L$. Also, M is *recognizable* if there exists a Turing machine M such that $M(x)$ accepts for all $x \in L$ and $M(x)$ either rejects or loops for all $x \notin L$. Show that L is decidable *if and only if* L and $\overline{L}$ are both recognizable.

   (b) Recall that the language $HALT = \{\langle M \rangle : M(\ )\text{ halts}\}$ is not decidable. Show that $HALT$ is recognizable (and that therefore, the decidable languages are a strict subset of the recognizable languages).

   (c) Show that $\overline{HALT}$ is *not* recognizable.

   (d) Show that every recognizable language L is Turing-reducible to $HALT$.

2. **Enumerators.** A Turing machine M *enumerates* a language L if, when M is run forever, M outputs a list of strings $x_1, x_2, x_3, \ldots$ containing all and only the strings in L. (The strings in L can be output in any order, and repeats are allowed.) Also, L is *enumerable* if there exists a Turing machine that enumerates L.

   (a) Show that L is enumerable if and only if L is recognizable.

   (b) Show that L is enumerable *in strictly increasing order, with no repeats*, if and only if L is decidable.

3. **Nondeterministic Turing Machines.** Recall from class that a nondeterministic Turing machine (NDTM) M is just a Turing machine that can make nondeterministic transitions—analogous to an NDFA or an NPDA. Given an input x, the evolution of $M(x)$ corresponds to a (possibly-infinite) *tree*, where each path from the root vertex downward corresponds to a possible history of M's computation. Each path can either be infinite (which corresponds to running forever) or finite (which corresponds to halting), and each finite path can either accept or reject at the leaf vertex.

   (a) Say that an NDTM M *decides* a language L if (i) $M(x)$ has at least one accepting path and no rejecting paths for every input $x \in L$, and (ii) $M(x)$ has at least one rejecting path and no accepting paths for every input $x \notin L$. Show that L is decidable by an NDTM, if and only if L is decidable by an ordinary Turing machine.

   (b) Say that an NDTM M *recognizes* a language L if (i) $M(x)$ has at least one accepting path for every input $x \in L$, and (ii) $M(x)$ has no accepting paths for every input $x \notin L$. Show that L is recognizable by an NDTM, if and only if L is recognizable by an ordinary Turing machine.

   (c) Given an NDTM M, say that $M(x)$ *halts* if every one of its computation paths is finite. Also, let L be the language consisting of $\langle M \rangle$ for every NDTM M such that $M(\ )$ halts. Show that L is recognizable. [*Hint:* Use König's Lemma.]

(d) Briefly explain why you *needed* König's Lemma for part c.

4. **Busy Beaver.** Recall that the *Busy Beaver function*, or $BB(n)$, is defined to be the maximum number of steps made by any n-state Turing machine that eventually halts (when run on an initially-blank tape).

    (a) Show that the function $BB(n)$ is Turing-reducible to $HALT$.

    (b) Let $C : \mathbb{N} \to \mathbb{N}$ be any integer function such that $C(n) \geq BB(n)$ for all n. Show that $HALT$ is Turing-reducible to C—so in particular, C is not computable.

    (c) *[Extra credit]* Show that there is not even a computable function C such that $C(n) \geq BB(n)$ for infinitely many values of n.

5. **Fun with Gödel.** Let F be some formal axiomatic system. You can assume F is *sound* (that is, it only proves true statements), and also that F is strong enough for Gödel's Incompleteness Theorem to apply to it. Let $G(F)$ be the Gödel sentence of F (that is, a mathematical encoding of "This sentence is not provable in F.") Also, let M be a Turing machine that generates all possible F-proofs, one by one, and halts if and only if it finds a proof of $G(F)$.

    (a) Does M halt? Why or why not?

    (b) Show that the question of whether or not M halts is independent of F.

    (c) Suppose M has k states. Show that, for all $n \geq k$, the value of $BB(n)$ is not provable in F.

6. **The Church-Turing Thesis in Action.** A *deterministic queue automaton* (DQA) is defined the same way as a deterministic pushdown automata (DPDA), except that it has a queue instead of a stack. In other words, a DQA is a deterministic finite automaton augmented with an unbounded queue, together with the operations of (a) pushing a symbol onto the "back" of the queue, and (b) popping the symbol at the "front" of the queue. Show that DQAs are equivalent in power to Turing machines: that is, any given language L is decidable by a DQA if and only if it's decidable by a Turing machine.

7. **Kolmogorov Complexity.** Let $s(n)$ be the number of possible n-state, single-tape Turing machines over the 3-symbol alphabet $\{0, 1, \#\}$.

    (a) Show that $s(n) \leq (6n+2)^{3n}$.

    (b) Say that a Turing machine M *generates* the string $x \in \{0,1\}^*$ if $M(\,) = x$: that is, if given an initially blank tape, M halts with $\cdots 0\#x\#0 \cdots$ written on its tape. Then let $K(x)$, or the *Kolmogorov complexity* of x, be the minimum number of states of any Turing machine that generates x. Show that $K(x) \leq n + O(1)$ for every n-bit string x.

    (c) Using part a, show that for every sufficiently large n, there exists an n-bit string x such that $K(x) \geq n^{0.99}$. [*Hint:* How many n-bit strings are there?]

    (d) Suppose $K(x)$ were a computable function. Using part c, show that there would exist a Turing machine that took any positive integer n as input (encoded in binary using $\log n$ bits), and that output a string $x \in \{0,1\}^n$ such that $K(x) \geq n^{0.99}$.

    (e) *[Extra credit]* Using part d, show that $K(x)$ is not a computable function.

6.045J / 18.400J Automata, Computability, and Complexity
Spring 2011