University of Virginia - cs3102: Theory of Computation	Spring 2010
Exam 2 - Comments	**15 April 2010**

Average: 71.5
Distribution: 90-100: 11; 80-89: 6; 70-79: 15; 60-69: 10; below 60: 14

Problem 1: Language Classification.

Classify each of the following languages as: (R) Regular, (CF) Context-Free, but not regular, (TD) Turing-Decidable, but not context-free, (TR) Turing-Recognizable, but not Turing-decidable, or (None) None of the above.

For full credit, your answer should include a brief argument supporting your answer (but a detailed proof is not needed).

a. (5) The set of strings generated by the replacement grammar (S is the start variable):

$$\begin{array}{lcl} S & \rightarrow & AS \mid SB \\ A & \rightarrow & 0 \mid S \\ B & \rightarrow & 1 \mid S \end{array}$$

Answer: (R) Regular. The language generated by the grammar is empty, since there is no way to derive a string that contains only nonterminals. A DFA can recognize the empty language (always reject), so the language is regular.

b. (5) $R_{TM} = \{\langle M, w\rangle \mid M$ is a description of a TM and w is not accepted by $M\}$

Answer: (None) None of the above. This language is similar to the complement of A_{TM}, which we know is not Turing-recognizable (Corollary 4.23). It is not identical to $\overline{A_{TM}}$ since A_{TM} is defined as $\{\langle M, w\rangle \mid M$ is a description of a TM and w is accepted by $M\}$. Its complement includes R_{TM} as well as all strings that are not of the form $\langle M, w\rangle$ (that is, they start with strings that are not descriptions of TMs). The set of strings that are not descriptions of TM (we'll calls this NOT_{TM} is definitely Turing-decidable since this is a simple syntactic property of the string. So $R_{TM} = \overline{A_{TM}} \cup NOT_{TM}$. The union of a Turing-unrecognizable and Turing-decidable language is not necessarily Turing-unrecognizable (see Problem 3b), so this isn't quite what we need. But, if we had a machine that can recognize R_{TM} would could use it to build a machine that recognizes $\overline{A_{TM}}$, which we know is impossible. This is easy: our machine first checks if its input is not a valid TM description, if so, it accepts. If not, it simulates the machine that recognizes R_{TM} and outputs the result.

Intuitively, the reason this language is not Turing-recognizable is because for a language to be Turing-recognizable it must *eventually* accept all strings that are in the language. To accept a string in R_{TM}, we need to know that M does not accept the string,

which means M either rejects it or runs forever. This cannot be determined by simulating M.

c. (5) $POWER = \{1^x * *1^y = 1^z \mid x, y, z \in \mathcal{N}, z = x^y\}$

Answer: (TD) Turing-Decidable, but not context-free. We know it can be decided by a Turing machine, using a similar approach to how we decided the $MULT$ language in class, so it must be in Turing-decidable. We know it is not context-free because it requires keeping track of more than one unbounded value. A more formal proof would use the pumping lemma for CFLs.

d. (5) *UVA-WINS* = $\{y \mid y$ is a four-digit string representing a year between 2000 and 2999 and UVa wins an NCAA soccer championship (Men's or Women's) in that year $\}$

Answer: (R) Regular. The language is finite, so there must be *some* DFA that recognizes it. We don't happen to know exactly which one yet, but we know there must be some DFA that recognizes the language, so it must be regular.

Problem 2: Decidability. For each subpart, answer whether or not the language is Turing-decidable. A full credit answer must include a convincing argument supporting your answer.

a. (10) *REPEATS-STATE*$_{TM}$ = $\{\langle M, w\rangle | M$ is a description of a TM and M enters some FSM state more than once in processing input w. $\}$

Answer: REPEATS-STATE$_{TM}$ is Turing-decidable. The key insight is the number of FSM states in a TM is finite, so the maximum number of steps a TM may run without re-entering an FSM state more than once is $2|Q| - 1$. We can decide the language with a TM. It is easiest to describe it as a multi-tape TM (which we know can be simulated by a single-tape TM), where we use the second tape to keep track of the entered states. Initially, we write each label in Q on the second tape twice. Then, simulate M on w. For every step, M transitions into some state q. Look for q on the second tape. If it is found, cross it off (write an X); if not, it has been entered twice, so *accept*. If M accepts or rejects, then M finishes without entering any state twice, so *reject*. This is a decider, since it is guaranteed to finish in no more than $2|Q|$ simulated steps.

b. (10) *REPEATS-START*$_{TM}$ = $\{\langle M, w\rangle | M$ is a description of a TM and M will enter q_0 more than once in processing input w. $\}$

Answer: REPEATS-STATE$_{TM}$ is **not** Turing-decidable. We prove this using a reduction from $HALTS_{TM}$ to *REPEATS-STATE*$_{TM}$.

Assume *REPEATS-STATE*$_{TM}$ is decidable. Then, there exists a TM M_{RS} that decides it. We could use it to construct a machine, $H(\langle M, w\rangle)$, that decides $HALTS_{TM}$:

(a) Construct a machine M' that adds a state q_{start} to M that is the start state of M'. Add transition rules from q_{start} to q_0 that write the same symbol they read and move left. (Note that on a one-way infinite tape TM, this leaves the tape in exactly the same configuration, and only changes the FSM's state to q_0.) Replace all the transitions in M to q_{accept} or q_{reject} with transitions to q_{start}. This means M' behaves identically to M, except it re-enters its start state only if M would accept or reject since there are no other transitions to q_{start}.

(b) Simulate $M_{RS}(\langle M', w\rangle)$ and output the result.

Thus, we have a contradiction and *REPEATS-STATE*$_{TM}$ must be undecidable.

c. (10) *REPEATS-CONFIGURATION*$_{TM} = \{\langle M, w\rangle | M$ is a description of a TM and M will repeat some configuration in processing $w\}$

Hint: Recall that the configuration of a Turing machine is the entire contents of the tape, the current FSM state, and the tape head position. For full credit, your answer must include a correct reduction proof.

Answer: REPEATS-CONFIGURATION$_{TM}$ is **not** Turing-decidable. We prove this using a reduction from $HALTS_{TM}$ to *REPEATS-CONFIGURATION*$_{TM}$ (abbreviated RC_{TM}. The most tricky part to realize is that there are ways a TM could never halt but also never repeat a configuration. This means, if $\langle M, w\rangle \in RC_{TM}$ we know $\langle M, w\rangle \notin HALTSTM$, but if $\langle M, w\rangle \notin RC_{TM}$ we do not know whether or not it halts. To deal with this, our reduction proof construction needs to create a machine M' that only repeats a configuration if M halts.

Assume RC_{TM} is decidable. Then, there exists a TM M_{RC} that decides it. We could use it to construct a machine, $H(\langle M, w\rangle)$, that decides $HALTS_{TM}$:

(a) First, simulate $M_{RC}(\langle M, w\rangle)$. If it accepts, *reject*. (If a configuration repeats, we know the machine can never halt since it must do exactly what it did last time it was in the same confiruation.)

(b) Now we know either M halts, or it doesn't halt without repeating a configuration. So, construct a machine M' that repeats a configuration if M halts. This is easy: replace transitions to q_{accept} and q_{reject} with transitions to a new state q_{repeat} where $\delta(q_{repeat}, a) = (q_{repeat}, \sqcup, L)$ for all $a \in \Gamma$. This eventually leads to a configuration repeating (when the tape head reaches the left edge of the tape).

(c) Simulate $M_{RC}(\langle M', w\rangle)$ and output the result.

Thus, we have a contradiction and RC_{TM} must be undecidable.

Problem 3: Language Classes.

a. (10) Define a *type of machine* that *recognizes* the set of languages in the *Mystery Class* ellipse (see original exam for picture).

Answer: Discussed in Class 23.

b. (5) If A is a Turing-recognizable language and $B \subseteq A$, is B necessarily a Turing-recognizable language? (Prove or disprove.)

Answer: **No.** Here's a counterexample: $B = \overline{HALTS_T M}$, $A = \Sigma^*$. B is a subset of A since A is the set of all strings. B is not Turing-recognizable, but A is Turing-recognizable (it can even be decided by at DFA).

c. (5) If A and B are Turing-decidable languages is the concatenation of A and B a Turing-decidable language? The concatenation of A and B is the language $\{xy | x \in A, y \in B\}$. (Prove or disprove.)

Answer. **Yes.** Since A and B are decidable, there exist TMs M_A and M_B that decide them. Note that we cannot just link together machines that decide A and B since M_A will be have differently on input xy than it would on input x. Instead, we have to try all possible ways of splitting the input into x and y. The number of possible ways is finite (the length of the input), so it is no problem to try them all. Here's a pythonish sketch of the algorithm:

```
decideConcat(w, MA, MB):
   for i in range(length(w)):
      if ((MA(w[:i]) and MB(w[i:])): return True
   return False
```

Problem 4: Janus Machine. (15) Consider a Janus Machine which is similar to a Turing machine except it has two heads which we will call the alpha and zeta heads. Initially, the alpha head starts at the left edge of the tape and the zeta head starts at the leftmost blank square (that is, one square after the end of the input). On each step, the input symbols under both heads are read, each head can a symbol on the tape, and each head can move one square left or right. Thus, the transition function for a Janus machine has three inputs and

$$\delta : Q \times \Gamma \times \Gamma \to Q \times \Gamma \times \{\mathbf{L}, \mathbf{R}\} \times \Gamma \times \{\mathbf{L}, \mathbf{R}\}$$

Prove that the set of languages that can be decided by a Janus Machine is identical to the set of language that can be decided by a Turing machine. (A super-full credit answer would also resolve an important ambiguity in the computing model for the Janus machine.)

Answer: Discussed in Class 23.

Problem 5: Fading Tape Machine. Consider a machine similar to a Turing machine, with an infinite tape, single tape head, and FSM controller, but where the symbols written on the tape are written with an ink that fades over time. The applies to both the original input written on the tape, and all symbols written on the tape by the machine.

A formal description of a Fading Tape machine is identical to that of a TM except for the addition of $K \in \mathcal{N}$, a natural number that gives the number of steps it takes for ink to fate. After K steps without being written, the symbol in a square becomes indistinguishable from a blank. Note that the value of K can be any natural number, but it is part of the description of the machine (that is, the value of K cannot depend on the input).

We can formalize our Fading Tape machine by adding a step-counter to the machine and thinking of each write as including a time-stamp. The configuration for a Fading Tape machine is the left side of the tape (note that each tape square now contains a pair consisting of the tape symbol and time stamp), the current FSM state, the current step count, and the right side of the tape. Thus, the extended transition function for a Fading Tape machine is,

$$\delta^* : (\Gamma, \mathcal{N})^* \times Q \times \mathcal{N} \times (\Gamma, \mathcal{N})^* \rightarrow (\Gamma, \mathcal{N})^* \times Q \times \mathcal{N} \times (\Gamma, \mathcal{N})^*$$

A partial definition of δ^* is below (we have omitted the rules for dealing with the edge of the tape, but these are similar to the rules for a standard TM).

$\forall u, v \in (\Gamma, \mathcal{N})^*, a, b \in \Gamma, n_i \in \mathcal{N}, q \in Q$:
  if $q \in \{q_{accept}, q_{reject}\}$:
    $\delta^*(u, q_F, n, v) = (u, q_F, n, v)$
  else:
    if $\delta(q, \sqcup) = (q_r, c, \mathbf{L})$ and $n_s - n_b > K$ or $b = \sqcup$:
      $\delta^*(u(a, n_a), q, n_s, (b, n_b)v) = \delta^*(u, q_r, n_s + 1, (a, n_a)(c, n_s + 1)v)$
    if $\delta(q, \sqcup) = (q_r, c, \mathbf{R})$ and $n_s - n_b > K$ or $b = \sqcup$:
      $\delta^*(u, q, n_s, (b, n_b)v) = \delta^*(u(c, 0), q_r, n_s + 1, v)$
    if $\delta(q, b) = (q_r, c, \mathbf{L})$ and $n_s - n_b \leq K$:
      $\delta^*(u(a, n_a), q, n_s, (b, n_b)v) = \delta^*(u, q_r, n_s + 1, (a, n_a)(c, n_s + 1)v)$
    if $\delta(q, b) = (q_r, c, \mathbf{R})$ and $n_s - n_b \leq K$
      $\delta^*(u, q, n_s, (b, n_b)v) = \delta^*(u(c, 0), q_r, n_s + 1, v)$

Note: the original question incorrectly used $(c, 0)$ instead of $(c, n_s + 1)$ as the output symbol. This doesn't actually change the answers, but if all the time stamps are written as 0 in the definition, then after step K the machine cannot write anything except blanks (so the entire tape would be permanently effectively blank after step K).

a. (10) To define the computing model for the Fading Tape machine, we need to specify its initial configuration. Provide a suitable description of the initial configuration to fill in the missing space in the computing model definition below.

   A Fading Tape Machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}, K)$ accepts a string $w \in \Sigma^*$ iff

   $$\delta^*(\textit{initial configuration needed}) = (\gamma_L, q_{accept}, n_{any}, \gamma_R)$$

   for some $\gamma_L, \gamma_R \in (\Gamma, \mathcal{N})^*$ and $n_{any} \in \mathcal{N}$.

Note: the original question omitted n_{any} from the right side. This is incorrect, since we defined the configuration of the FTM to include the step counter, even though it has no impact on the acceptance decision.

Answer: The initial configuration should be similar to that of a TM, except with timestamps added to all the input squares and the initial time step set to 0:

$$(\epsilon, q_0, 0, ((w_0, 0), (w_1, 0), \ldots, (w_{n-1}, 0)))$$

b. (10) Define precisely the power of a Fading Tape machine. Support your answer with a very convincing argument.

Answer: The FTM is less powerful than a DFA: it can only recognize languages that are distinguished by finite prefixes. We know it is not more powerful than a DFA since its memory is finite. Since there is a limit on the number of non-blank tape squares, we can represent the state of an FTM by a finite set of $(position, \Gamma)$ pairs, where all other squares on the tape must be blank. The number of configurations is finite, so we could represent each configuration with a DFA state. In fact, it is less powerful that a DFA since the input also fades! For example, it could not recognize a language like 0^*1 wich is regular, since the number of 0s could exceed K (no matter what K is, there is some longer string in the language), so the 1 will have faded before the tape head can reach it. This means it can only recognize language that can be recognizes by looking at some finite number of the first symbols in the string: all finite languages and all languages that are concatenations of finite languages with Σ^*.