Models of Computation (Sheet #5)

Marius Gavrilescu

1. (a) We'll build a NFA for this language. Let the states be q_0 and q_1.

   We have a transition from state q_0 to q_0 when the symbol read is one of $\begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\1\end{pmatrix}, \begin{pmatrix}1\\0\\1\end{pmatrix}$

   We have a transition from state q_0 to q_1 when the symbol read is $\begin{pmatrix}0\\0\\1\end{pmatrix}$.

   We have a transition from state q_1 to q_0 when the symbol read is one of $\begin{pmatrix}1\\1\\0\end{pmatrix}$.

   We have a transition from state q_1 to q_1 when the symbol read is one of $\begin{pmatrix}0\\1\\0\end{pmatrix}, \begin{pmatrix}1\\0\\0\end{pmatrix}, \begin{pmatrix}1\\1\\1\end{pmatrix}$

   The start state, and only accepting state, is q_0.

   (b) First we'll show it is not regular.

   Take the strings $s_a = 01^{2a}01^a$. For every two such strings s_a and s_b with $a \neq b$ we have $s_a \not\equiv s_b$. By the Myhill-Nerode theorem we know this language is not regular.

   Then we'll make a context-free grammar for this language:

   $S \to 1S1|0A$
   $A \to 0|1A1$

   where the starting symbol is S. As this grammar is context-free and it generates the language, we know our language is context-free.

   (c) Assume the language is context-free. Then for some p we can divide any word w of the language into five pieces $uxyzv$ such that $ux^iyz^iw \in L \forall i \geq 0$, $|xz| > 0$ and $|xyz| \leq p$.

   Take the string $01^p; 10^p; 1^{p+1}$. Now because $|xyz| \leq p$, we know x and z will only affect one or two adjacent parts of the string, but not all three of them. If x and z affect a single part, we can either make that part bigger or smaller by setting $i = 0$ or $i = 2$, and the resulting string is no longer in the language. If they affect the leftmost two parts, then we can make both the left parts bigger or smaller, so the resulting string also is no longer in the language. Finally, if they affect the rightmost two parts then $x = 0^a$ and $z = 1^b$. Setting $i = 0$ will decrease both parts by different amounts.

   So there is no way to divide the string above as required by the pumping lemma, which means our language is not context-free.

   (d) This language is context-free. We can make a NPDA that pushes 1s on the bottom row onto the stack, and pops 1s on the top and middle rows from the stack. That NPDA should also check that each row is of the form 0^x1^y.

   The language is not regular because strings $s_i = (0, 0, 1)^i$ are all in different equivalence classes.

2. No. It is not closed under complementation. We know a problem is decidable if and only if the problem and its complement are both RE. If RE was closed under complementation, then every RE problem was decidable. $HALT_{TM}$ is known to be RE, but undecidable, which contradicts our assumption.

3. (a) For a given y we can enumerate all elements x of A and run R on (x, y). If we get an accept from R we accept. This is guaranteed to try every possible x eventually because we know R will always halt. Thus the $\exists xR$ problem is RE.

   (b) Let M be a TM for C.
   Let $A = Z$ and $R = \{(x, y)|$ after running C on y for x steps, C accepted$\}$.
   R is decidable, as we can run C on y for x steps and see if it accepts. $\exists xR = C$.

4. (a) For any string $x_1x_2 \dots x_k \in \sum_1^*$ we have $h(x_1x_2 \dots x_k) = h(x_1)h(x_2) \dots h(x_k)$. So it is sufficient to define $h : \sigma_1 \to \sigma_2$.

   (b) Take the regular language for L and replace every terminal t by $h(t)$. The resulting expression matches $h(L)$.

   (c) Take the grammar for the language and apply the homomorphism to every terminal symbol.

   (d) Take the grammar for the language and apply the inverse homomorphism to every terminal symbol. If the result is a set with one element, this is the same as above. If the result is an empty set, delete this word from the grammar. If the result is a set with more than one elment, duplicate the word appropriately (if it is the LHS of a rule, make new identical rules; if it is on the RHS of a rule, add the new words as conjunctions) to cover each element of the result.
   So for example if our alphabet is $\{a, b, c\}$ and we have $h(a) = h(b) = x, h(c) = y$, then the grammar:
   $S -> xxS$
   $xS -> y|x$
   translates to:
   $S -> aaS|abS|baS|bbS$
   $aS -> c|a|b$
   $bS -> c|a|b$

5. (a) Decidable. Simply run the TM on a^{127} for 127^{127} steps and see if it has halted or not.

   (b) Undecidable. Assume it was decidable and let D be a decider for it. For a given TM a with start state x_0 make a new state x_s and set $\delta(x_s, \vdash) = \delta(x_0, \vdash)$. Then change all mentions of the accepting and rejecting states to x_s and make x_s the new starting state.
   The TM we just constructed has the same behaviour as a. Now we can make a new machine b_X that discards its input and simulates a on input X. Finally we can implement $HALT_a$ by running D on b_X, where X is the input. Thus $HALT_{TM}$ is decidable, which is false. Thus our assumption is incorrect.

   (c) Undecidable. Assume it was decidable and let D be a decider for it. For a given TM a, make a TM M_a that runs a on its input and accepts when a finishes running.
   Now we can implement $HALT_{TM}$ by running D on M_{TM} and accepting if D accepts, rejecting otherwise. This means $HALT_{TM}$ is decidable, which is false. Thus our assumption is incorrect.