6.045J/18.400J: Automata, Computability and Complexity Prof. Nancy Lynch

Recitation 6: Decidability and Undecidability

March 15, 2007 *Elena Grigorescu*

Problem 1: These are the key concepts from lecture this week:

1. Undecidability - p. 172-176 have great example proofs.
2. Reductions - p. 171-172 will help with the terminology (e.g., "reduce A from B", etc.)
3. Computation history - p. 176, 179, 185 give the definition and some examples.
4. Diagonalization method - p. 160-168. This concept is both elegant and difficult; make sure you understand it.

Problem 2: Show that the following languages are undecidable:

1. $DECID_{TM} = \{< M > \mid M$ halts on any input in accept or reject$\}$
2. $L = \{< M >$: M is a Turing machine and M accepts exactly the strings in Σ^* whose length is a power of $2\}$.

Problem 3: Show that the following language is undecidable:

$$EQ_{TM} = \{< M, N > \mid M \text{ and } N \text{ are TMs such that } L(M) = L(N)\}$$

Reduce from both E_{TM} and A_{TM}. Recall that EQ_{DFA} was decidable.

Solution 3: In class we saw how to reduce E_{TM} to EQ_{TM}. Here we will reduce from A_{TM} to prove that EQ_{TM} is undecidable. Let D be a TM that decides EQ_{TM}. We could then construct a decider S for A_{TM} as follows.

S="On input $< M, w >$, an encoding of a TM M and a string w,

1. Construct TM R_1 from M and w and TM R_2 as detailed below.
2. Run D on $< R_1, R_2 >$.
3. If D accepts, reject; otherwise, accept."

R_1="On input x,

1. Run M on w.
2. If M accepts, accept"

Notice that R_2 is the TM that we constructed when we proved EQ_{TM} was undecidable by reducing from E_{TM} (i.e., $L(R_1) = \emptyset$).
R_2="On input x,

1. reject."

Thus, we contrive that $L(R_1) = \emptyset$ if and only if M rejects w, while $L(R_2) = \emptyset$ always. Since, by assumption, we have a decider D that tells us if these two machines recognize the same language, we know that if D rejects R_1 and R_2, then this implies that M accepts w.

Problem 4: (From Sipser, problems 5.17 and 5.18)
Consider the Post Correspondence Problem over small alphabets.

1. Show that the problem is decidable over the unary alphabet $\{0\}$.
2. Show that the problem is undecidable over the binary alphabet $\{0, 1\}$ (bPCP).

Solution 4: 1. **Sketch of Proof:** We prove it is decidable, by giving an algorithm that decides it. Each

domino d_i in the set has a top portion of 0^{k_i} and a bottom portion of 0^{m_i} for some $k_i, m_i \geq 0$. Lets consider the values $c_i = k_i - m_i$:

1. If c_i for some domino, accept. [That single domino is a match.]
2. If $c_i > 0$ for all dominos, reject.
3. If $c_i < 0$ for all dominos, reject.
4. If $c_i > 0$ and $c_j < 0$ for $i \neq j$, then accept. [You can even these out.]

■

2. **Sketch of Proof:** We prove it is undecidable by reducing from PCP (over an arbitrary alphabet Σ).

Assume a TM D that decides $bPCP$. Build a TM S to decide PCP.

S="On input $< d_1, d_2, \ldots, d_k >$, where each d_i is a domino,

1. Count the number of different symbols on the dominos: $|\Sigma|$.
2. Assign to each unique symbol a unique (iterative) m-bit (or you could also reduce this to a $log(m)$-bit) value. Front-pad with zeros.
3. Construct new dominos $< d'_1, d'_2, \ldots, d'_k >$ using the binary encoding.
4. Run D on $< d'_1, d'_2, \ldots, d'_k >$.
5. If D accepts, accept; otherwise, reject."

We know that the number of symbols counted in step 1 is finite, since the number and content of each domino is finite. By giving unique binary encodings *of equal length* to each domino, the problem reduces nicely. Observe that this would not necessarily be true if our encoding for each unique symbol was allowed to be of different lengths. ■