CS 381 Homework 11 solutions

November 9, 2000

Problem 27.1 Give an algorithm to decide for two strings α and β of non-terminals and terminals of a CFG G, whether $\alpha \rightarrow_G^* \beta$.

Assume G is in Chomsky Normal Form. Then we have rules of two types: $A \rightarrow BC$ and $D \rightarrow e$. Now we do the following:

1. $S \leftarrow \{\alpha\}$
2. for $i = 1$ to $|\beta| - |\alpha|$ do
   (a) for each item γ in S, for each non-terminal symbol N in γ, and for each rule $N \rightarrow OP \in G$, we apply the rule to γ and add that string to S'.
   (b) $S \leftarrow S'$

We now have a very large set S, consisting of all possible strings of terminals and non-terminals derivable from α by application of rules of the type $A \rightarrow BC$. We know the set is finite, because $|\alpha|$ is finite, $|G|$ is finite, and each time we apply a rule of that type, we make our string one symbol longer than it was, so we need to do this $|\beta| - |\alpha|$ times, which is finite.

Now we can, for each string γ in S, compare each symbol γ_i to the corresponding symbol β_i:

- If $\gamma_i = \beta_i$, we examine $i + 1$.
- If γ_i is a terminal and β_i is a non-terminal, then we reject this string.
- If γ_i is a non-terminal and β_i is a terminal, we must check if there is a rule of the form $\gamma_i \rightarrow \beta_i$. If so, we go on to $i + 1$; if not, we reject this string.

If one of the strings in S is not rejected, then we have found a way to transform α into β.

Problem 28.1

b) $\{ww^R | w \in \{0,1\}^*\}$. The machine does the following on input x:

1. Mark off and remember the first unmarked symbol in x: if it is a 0, replace it with $\not{0}$; if it is a 1, replace it with $\not{1}$.
2. Move to the last unmarked symbol on the tape.
3. Compare the last unmarked symbol on the tape. If it is the same as the remembered symbol from step 1, mark that symbol off as in step 1. If it is different, reject the string.
4. Move to the first unmarked symbol on the tape. If there are no more unmarked symbols, accept the string. Otherwise, go to step 1.

c) The set of strings with equal number of 0's and 1's. The machine does the following on input x:

1. Mark off the first unmarked symbol and remember if it is a 0 or a 1. If there are no unmarked symbols, accept the string.

2. Move right along the tape until we reach a 0, if we marked a 1 in step 1, otherwise until we reach a 1. If we advance past the end of the input, reject the string.
3. Mark off this symbol.
4. Move left to the beginning of the tape.
5. Go to step 1.

Problem 28.2 Design a Turing Machine to compute n^2. We assume our input to be in unary.

1. Place the symbol # at the end of the input.
2. For each symbol in the input, copy the entire input to the end of the tape.
3. If we want the tape to only contain n^2, we must copy everything after the # to the beginning of the tape and erase the rest of the tape.

Note that we will be using several symbols to do the copying and marking.

Problem 28.3 Construct a TM to accept $\{a^n | n \text{ is a power of } 2\}$.

1. If there is a single unmarked a on the tape, accept.
2. Staring with the second a, mark off every second a to the end of the tape. If the tape ended in an unmarked a, reject, as there were an odd number of a's.
3. Go to step 1.

Problem 28.4 Prove that an r.e. set is recursive iff there exists an enumeration machine that enumerates it in increasing order.

We'll prove this in two directions, with a slight rewording to make it clearer what we're trying to accomplish.

First: Given a recursive set S, show $\exists$ and enumeration machine that enumerates it in increasing order. Since S is recursive, we know there exists a total TM M deciding S. Now consider the following TM M': For each string in Σ^*, in lexicographic order, run M on that string. If it accepts, print out the string.

M' is an enumerator for S that enumerates in increasing order. It is clear that it enumerates in increasing order because we examine strings in increasing order. It enumerates S because, since M is total, we always have a membership answer on each string we look at in finite time, so it will eventually print any given string in S in finite time.

Second: Given an enumerator for a r.e. set S that enumerates in increasing order, show that S is recursive. Let's call the enumerator M'. Now consider the following description of a TM M: On input x it does the following: Run the enumerator M' until we see the string x or some string y that is larger than x, or the enumerator halts.

If the enumerator prints x, accept x.

If the enumerator prints a string $y > x$ without printing x, reject x, as if $x \in S$, the enumerator would have printed it before y, as it is enumerating in increasing order.

If the enumerator halts before printing x, then the set S is finite and x is not in it.

The created machine M is total, because it always accepts or rejects its input in finite time.