COS 487
Fall 2003

Problem Set 8 Solutions

Problem 1:

Let SPACE1($f(n)$) and SPACE2($f(n)$) be the space complexity classes for the single and two-tape models respectively.

Let M be a single-tape Turing machine. We can construct a two-tape Turing machine with the same space bound - the machine simply copies the input to the work tape, returns the head to the beginning of the work tape, and then simulates M.

For the other direction, let M be a two-tape Turing machine. The equivalent single tape machine uses the first n tape positions to hold the input and the rest of the tape as a work tape. Special markers are used to indicate the head position in each tape, and the Turing machine updates both tapes as required. The single tape machine uses space $O(n+f(n))$, which is $O(f(n))$ since $f(n)$ is at least linear.

Problem 2:

Player two has a winning strategy. Player one's only first move is 2. After player two chooses 4, player one can only choose 5. Player two then wins after player two chooses 6.

Problem 3:

We give a coNPSPACE machine that recognizes $EQ_{REX.}$ Convert each regular expression into an equivalent NFA. An NFA with n states has an equivalent DFA with 2^n states. Furthermore, two different DFA's with at most 2^n states can be run in parallel with 2^{2n} states. Therefore, they will differ on some input of length less than 2^{2n+1}. (If they differ on an input of greater length, we can pump down by the LCM of the two pumping lengths). We can maintain a counter running from 1 to $2^{n^\wedge 2}$ in polynomial space. For each time step, we nondeterministically guess the next input symbol, simulate both NFAs with this next input symbol, and increment the counter, accepting if the two NFAs disagree. Since coNPSPACE = PSPACE, we are done.

Problem 4:

STRONGLY-CONNECTED is in coNL = NL. We co-nondeterministically select each vertex pair and run the PATH algorithm, accepting if there is a path between all pairs.

Given an instance <G, s, t>, we add a directed edge from each vertex to s and from t to each vertex. This requires only log space since we need only count through the number of vertices. If a path existed from s to t, the output is strongly connected (all vertices connected through a path containing *s* and *t*). If no path exists, then there is no path from *s* to *t* in the output, and the graph is not strongly connected.