MATH/CSCI 387 Homework 7

Sima Nerush

Discussed problems with Harrison and Riley

Problem 1. Show that any PSPACE-hard language is also NP-hard. (Remember that "NP-hard" requires the same thing as NP-completeness, except that the language does not have to be in NP. PSPACE-hard is defined similarly.)

For a language L_1 to be NP-Hard requires that for all A in NP, $A \leq_p L_1$.

For a language L_2 to be Pspace-hard requires that for all A in Pspace, $A \leq_p L_2$.

Proof: Let $L \in$ PSPACE-hard. This implies that any $A \in$ PSPACE is no harder than L. Therefore, any $B \in$ NP is also no harder than L. This shows any $B \in$ NP can be reduced to L. Thus, L is also NP-hard.

Problem 2. The game Gomoku is played by two players on an n $\times$ n board. The players alternate placing pieces, with one placing red and the other placing blue. (The pieces must be placed on open spaces.) The winner is the first player to achieve a line of 5 consecutive markers (in a row, column, or diagonal). A position consists of a description of what stones are on the board and whose turn it is. Let $GOMOKU$ be the set of positions from which red can force a win. Show that $GOMOKU \in PSPACE$.

Formulated as a language:

$GOMOKU = \{p \mid p \text{ is a position where red can force a win}\}$

Define a PSPACE algorithm M for $GOMOKU$:

1. Get a position as an input.
2. If it is a leaf and a winning position of Red, accept.
3. If it is a leaf and a winning position for Blue, reject.
4. If It is Red's turn:
    1. Make a recursive call of M on the position corresponding to all possible next turns.
    2. Erase the contents of the tape, storing only the results of the recursive calls.
    3. If one of them accepts, accept. Else, reject.
5. If It is Blue's turn:
    1. Make a recursive call of M on the position corresponding to all possible next turns.
    2. Erase the contents of the tape, storing only the results of the recursive calls.
    3. If all of them accept, accept. Else, reject.

Analysis:

Recursion can have at most n^2 depth, because at every turn it takes at most n^2 subsequent turns to fill up the board.

Each level of recursion adds $O(n^2)$ characters to the tape because it takes $O(n^2)$ symbols to represent a board.

It also takes $O(n^2)$ time to check whether a board position is a win. (This implies that the check can also be done in polynomial space).

Problem 3. Show that $PSPACE$ is closed under the star operation.

Given a string and a polynomial time decider for L, we want to determine whether our string is in L^*.

Define an algorithm M for L^*:

1. Get a string as input:
2. If $s \in L$, return true.
3. for every possible splitting into parts l and r of s:
    1. Recursively call M on part l of the string. If it accepts, erase the calculations on the tape and recursively call M on part r of the string. If it accepts, accept.
    2. Erase the calculations on the tape.
    3. reject.

Analysis:

Recursion can have at most n depth, because there are n characters in the input string, and we cannot split beyond this point.

Each level of recursion adds some polynomial number of characters to the tape because we have a polynomial space decider for L.