Nondeterministic Space is Closed Under Complement

This theorem was proved independently by Neil Immerman and Róbert Szelepcsényi in 1987. Immerman was an experienced researcher in computational complexity who built his proof based on recent papers that had proved related but significantly weaker statements. Szelepcsényi was an undergraduate student who solved the problem from a list of starred problems in a course on proving that the set of *context-sensitive languages* is closed under complement, which was an equivalent open problem. Though the ideas were closely related, the proof we give is very much like Szelepcsényi's proof.

Using the NL-completeness of $PATH$, the question is equivalent to the following theorem.

Theorem [Immerman-Szelepcsényi 1987] $\overline{PATH} \in NL$ (and therefore $NL = co\text{-}NL$).

Proof. Suppose that we are given an input $\langle G, s, t\rangle$ where $G = (V, E)$ is a directed graph with vertices s and t. We wish to verify that there is *no path* in G from s to t using a nondeterministic small space algorithm. Let $n = |V|$. The standard NL algorithm for $PATH$ consists of guessing and verifying a path from s of length at most n, one vertex at a time.

The key new idea to show that there isn't a path is the following: Suppose that we had access to a number $Count = \#\{v \in V \mid \text{there is a path from } s \text{ to } v \text{ in } G\}$.

Then the following nondeterministic algorithm would correctly determine that there is no path from s to t since it would find $Count$ other vertices reachable from s in G:

NoPath($s, t, n, Count$):
$reach \leftarrow 0$
for all $v \in V$ with $v \neq t$ **do**
   Guess whether or not v is reachable from s by a path of length $\leq n$
   if Guess is yes **then**
      Guess and verify a path of length at most n from s to v, one vertex at a time
      if path is found **then**
         $reach \leftarrow reach + 1$
      else
         reject
      end if
   end if
end for
if $reach = Count$ **then**
   accept
else
   reject
end if

This algorithm only needs to maintain $reach$, $Count$ and a constant number of other vertices at a time and hence it takes only $O(\log n)$ space.

So all we need to do is to produce the value $Count$ using an $O(\log n)$ space nondeterministic algorithm. We will do this inductively, by showing how to complete $Count_i$, the count of the number of vertices reachable from s using at most i hops for each i from 0 to n. For convenience we will write the separate values of $Count_i$ for all i but at any point in time the algorithm will at most maintain two values $Count_i$ and $Count_{i+1}$ in memory.

First observe that $Count_0 = 1$ since s is the one node reachable in 0 hops from s. Now suppose that we have correctly computed $Count_i$. Then we can use NoPath with appropriate parameters and v in place of t to verify that there is no path of length at most i from s to a vertex v. The basic idea for computing $Count_{i+1}$ is that we can guess which vertices should contribute to this count and verify the difficult case that we have guessed that there is no path of length $i+1$ from s to v, by verifying that no predecessor u of v in G is reachable by a path of length at most i. The following code implements this idea:

```
ComputeCount(G, s, t):
Count_0 ← 1
for i = 0 to n − 1 do
   Count_{i+1} ← 0
   for all v ∈ V do
      Guess whether or not v is reachable from s in ≤ i + 1 hops
      if Guess is yes then
         Guess and verify a path of length at most i + 1 from s to v, one vertex at a time
         if path is found then
            Count_{i+1} ← Count_{i+1} + 1
         else
            reject
         end if
      else if Guess is no then
         for all u ∈ V with (u, v) ∈ E do
            if NoPath(s, u, i, Count_i) rejects then
               reject
            end if
         end for
      end if
   end for
end for
return Count_n.
```

□

By applying the above construction to the configuration graph $G_{M,x}$ with $s = C_0 = (q_0 x,)$ and $t = C_{accept}$ for any $S(n)$-space bounded TM M, we obtain the following:

Corollary 1: For every $S(n) \geq \log_2 n$, $A \in NSPACE(S(n)) \Leftrightarrow \overline{A} \in NSPACE(S(n))$.

Corollary 2: The set of languages accepted by linear-bounded automata (equivalently, the set of context-sensitive languages) is closed under complement.