PATH is NL-Complete

This material was presented in class on February 3, 2015. The lecture began with definitions 4.5, 4.16, and 4.19, all of which are presented clearly in the textbook and hence won't be repeated here.

Claim 1 *PATH is in NL.*

Proof. A PATH instance is a triple (G, s, t), where G is a directed graph, and $\{s, t\} \subseteq V(G)$. The yes instances are those in which there is a path from s to t in G. Note that, if $V(G) = \{1, 2, \ldots, n\}$, the instance (G, s, t) is of length $c \cdot n^2$, for some positive constant c, assuming that we encode G as an $n \times n$ matrix of bits in which the $(i, j)^{th}$ bit is a 1 if and only if the arc (i, j) is in $A(G)$. (Note "arc" instead of "edge" and $A(G)$ instead of $E(G)$, in order to emphasize that G is a *directed* graph. PATH is a totally different, easier problem for undirected graphs.) So we seek a nondeterministic algorithm that decides PATH in space $O(\log(c \cdot n^2)) = O(\log n)$. Here is one such algorithm:

```
PATH(G, s, t)
{
  i ← 0;
  u ← s;
  WHILE(i ≤ n)
  {
    IF (u = t) THEN OUPUT(ACCEPT) AND HALT;
    GUESS u′ ∈ V(G);
    IF ((u, u′) ∈ A(G)) THEN u ← u′;
    i ← i + 1;
  }
  OUTPUT(REJECT) AND HALT;
}
```

Things to notice about this algorithm:

- If there is a path from s to t, then there must be one of length less than or equal to n, because there are only n nodes in G.
- We cannot simply guess a path of length at most n in one fell swoop, because that would require $\Omega(n \log n)$ bits of workspace. Thus, we guess one node at a time and verify that all of the requisite arcs are there.
- It is clear that the values of the variables i, u, and u' require $O(\log n)$ workspace. Not as apparent, but still not hard, is that the bit on the input tape that tells us whether $(u, u') \in A(G)$ can be read in space $O(\log n)$ using a counter.

∎

Claim 2 *Every set in NL is logspace-reducible to PATH.*

Proof. Let S be a set in NL and M be a nondeterministic logspace machine that recognizes S. We must exhibit a logspace reduction f from S to PATH, *i.e.*, an implicitly logspace-computable f such that $x \in S$ if and only if $f(x) \in$ PATH.

The directed graph G in $f(x)$ is the configuration graph $G^{M,x}$; the nodes s and t in $f(x)$ are the START and ACCEPT configurations $C^{M,x}_{\text{START}}$ and $C^{M,x}_{\text{ACCEPT}}$ in $V(G^{M,x})$. By definition of "configuration graph," we have that $x \in S$ if and only if $f(x) \in$ PATH; so it remains to prove that f is logspace-computable.

There is a constant c that depends on M but not on x such that the number of bits required to encode any configuration $C \in V(G^{M,x})$ is $c \log n$, where $n = |x|$. The number $|V(G^{M,x})|$ of configurations is $2^{c \log n} = n^c$, and $G^{M,x}$ can be written down explicitly (as an adjacency matrix) using n^{2c} bits. Therefore, the length $|f(x)|$ of the target instance $(G^{M,x}, C^{M,x}_{\text{START}}, C^{M,x}_{\text{ACCEPT}})$ is polynomial in $|x|$ (specifically, $n^{2c} + 2c \log n$, where $|x| = n$), and we can clearly determine in space logarithmic in $|x|$ whether $i \leq f(|x|)$.

It remains to show that each bit in $f(x)$ can be computed in space logarithmic in $|x|$. The configurations $C^{M,x}_{\text{START}}$ and $C^{M,x}_{\text{ACCEPT}}$ can each be written down explicitly in space $c \log n$, where $n = |x|$; so it is clear how to determine whether each of the last $2c \log n$ bits of $f(x)$ is a 1 in space $O(\log n)$. To determine the $(C, D)^{th}$ bit of the adjacency matrix in $f(x)$, we use the following logspace procedure: Write C on a work tape, $C' = \delta_0(C)$ on a second work tape, and $C'' = \delta_1(C)$ on a third work tape, where δ_0 and δ_1 are the transition functions of M; then output 1 if and only if $D = C'$ or $D = C''$. Say M has k writable tapes and (logarithmic) space complexity s. Recall that

$$C = (q, P_0, P_1, \ldots, P_k, \gamma_{1,1}, \gamma_{1,2}, \ldots, \gamma_{1,s}, \ldots, \gamma_{k,1}, \gamma_{k,2}, \ldots, \gamma_{k,s}),$$

where q is an element of the state set of M, P_0 is the position of M's input tape head, P_w is the position of the w^{th} writable-tape head in M, $1 \leq w \leq k$, and $\gamma_{w,j} \in \Gamma$ is the symbol in the j^{th} cell of the w^{th} writable tape. Similarly, let

$$C' = (q', P'_0, P'_1, \ldots, P'_k, \gamma'_{1,1}, \gamma'_{1,2}, \ldots, \gamma'_{1,s}, \ldots, \gamma'_{k,1}, \gamma'_{k,2}, \ldots, \gamma'_{k,s})$$

and

$$C'' = (q'', P''_0, P''_1, \ldots, P''_k, \gamma''_{1,1}, \gamma''_{1,2}, \ldots, \gamma''_{1,s}, \ldots, \gamma''_{k,1}, \gamma''_{k,2}, \ldots, \gamma''_{k,s}).$$

Computation of C' (resp. C'') can be done in space $O(|C| + |C'|) = O(\log n)$ (resp. $O(|C| + |C''|) = O(\log n)$), because each of its components can be looked up in a (constant-sized) table that specifies the transition function δ_0 (resp. δ_1). ∎