Non-deterministic Space

Note Title 4/16/2013

NSPACE $(f(n))$ – languages decidable by a multi-tape NTM that touches $\leq f(n)$ squares of work tape along any computation path.
this means all paths must terminate.

Robust non-deterministic space classes:

$$NL = NSPACE(\log n)$$
$$NPSPACE = \bigcup_{k \geq 1} NSPACE(n^k)$$

First the relationship between these classes + time complexity classes. Recall that on input x, $|x| = n$ a $t(n)$ space bounded device (det or non-det) has at most

$$\underbrace{\log n \times \log t(n)}_{\text{head locations}} \times \underbrace{|Q|}_{\text{state}} \times \underbrace{\Sigma^{t(n)}}_{\text{work tape contents.}} \text{ configurations.}$$

Can build a configuration graph where each node is one of these configurations. Edges correspond to one step in the computation. For a deterministic TM, the out-degree is 1 (or 0 if it's a halting state.
In a non-deterministic computation, out-degree is bounded by some constant.
The problem boils down to connectivity:
Is there a path from the starting configuration to an accepting configuration.
This question can be answered in time poly in the size of the graph:

$t(n) = \log n$ $\quad$ $NSPACE(t(n)) = NL \subseteq P$
$t(n) = n^k$ $\quad$ $NSPACE(n^k) \subseteq EXP \ \forall k \Rightarrow NPSPACE \subseteq EXP.$

This suggest a problem that could be complete for NL:

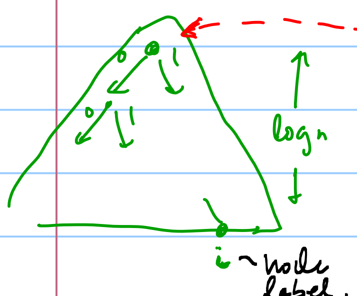
ST-Connectivity (STCONN)

Input: $G = (V, E)$; $s, t \in V$

Is there a path from s to t in G?

Theorem: STCONN is NL-complete (under log space reductions)

STCONN $\in$ NL:



$v = s$

clock $= 0$

Use non-determinism to "guess" a node label v'.

If $(v, v') \notin E$ reject.

Else increment clock

if $v' = t$ accept.

if clock $= n$ reject.

$v \leftarrow v'$

Repeat

If there is no path from s to t, every computation path will hit a dead end in the graph or the timer will run out. (i.e. never reach t)

If there is a path from s to t: $s = v_0 \to v_1 \to v_2 \cdots \to v_e = t$

there is a sequence of guesses that will lead to t.

Now prove $\forall L \in NL$ $\quad L \propto_\ell$ STCONN

There is an NTM M that uses $\log n$ cells of work tape that decides L.

Will describe logspace R: $x \longrightarrow G_x, s, t$.

$x \in L$ iff $\exists$ path from s to t in G_x

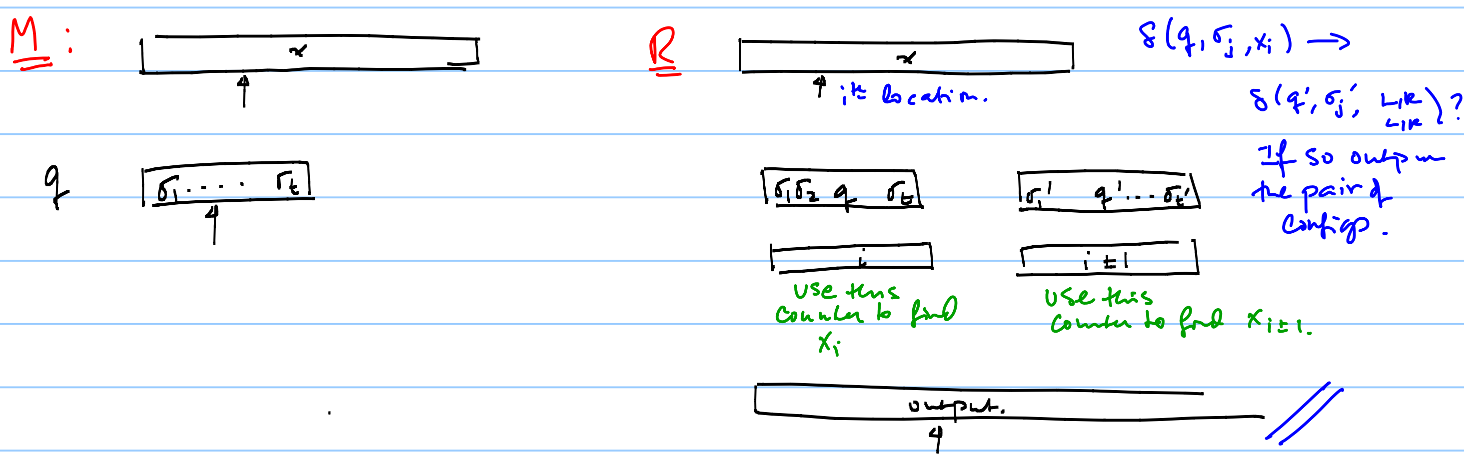
On input x, there will be
$\log^2 n \times |\Sigma|^{t(n)} \times |Q|$ configurations of M.
Each can be specified using $O(t(n))$ bits.

R:

Run through all possible pairs of configurations on the work tape. For each pair determine if if one can be reached from the other in one step of M. If so: output the edge.

Add new node "t": add edge from each accepting configuration to t.

Output: s = start config, t = "t".



Now we will prove two surprising theorems about non-deterministic space classes.

Savitch '70: NPSPACE = PSPACE
(NL $\subset$ SPACE$(\log^2 n)$)

Immerman-Szelepscény ('87/'88): NL = co-NL

These are the opposite of what we believe to hold for time complexity classes. (P=NP, NP = co-NP)

Savitch's Theorem $STCONN \in SPACE(\log^2 n)$

Corollary: $NL \subseteq SPACE(\log^2 n)$

Corollary: $NPSPACE = PSPACE$

The configuration graph for a poly-space NTM M has size 2^{cn^k}. The algorithm that solves STCONN in space $\log^2 n$ doesn't need to construct the graph explicitly. It only requires answers to queries: is (i,j) an edge? So, we can solve the connectivity problem on a graph of size 2^{cn^k} in $\log^2(2^{cn^k}) = O((cn^k)^2)$. We can answer queries of the form: (c, c') is there a single move of M that transforms c into c'?

Proof that ST-CONN $\in$ SPACE$(\log^2 n)$:

Input: $G = (V,E)$ & $s, t \in V$.
recursive algorithm:

PATH (x, y, i) // is there a path from x to y of length $\leq 2^i$

```
    if i=0 return (x == y ∨ (x,y) ∈ E);
    For all nodes z:
        if PATH(x,z,i-1) ∧ PATH(z,y,i-1) return true
        else return false.
```

Return PATH$(s, t, \log n)$

Space used: (depth of recursion) $\times$ (size of stack record) depth = $\log n$.

Stack record: (x, y, i) $O(\log n)$ space.

Can figure out what to do next from current & previous record

ex: $(x,y,i)\ (x,z,i-1) \longrightarrow$ next call $(z,y,i-1)$

: $(x,y,i)\ (z,y,i-1) \longrightarrow$ pop record & return results.

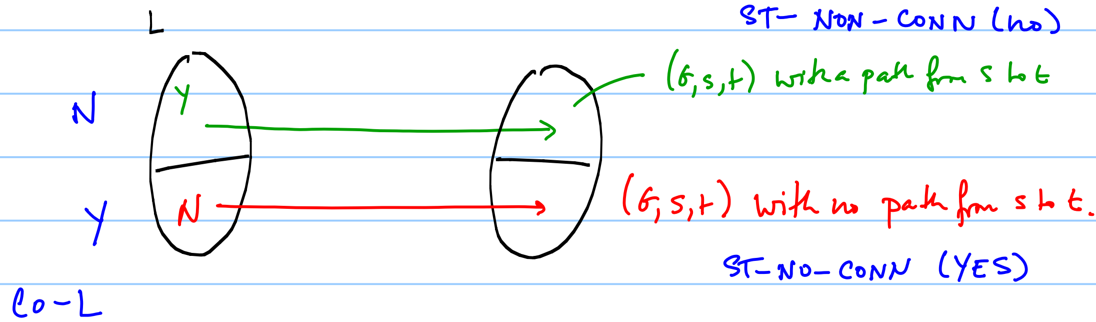
//

ST-NON-CONN

Input: $G = (V, E)$ s, t.

Is there no path from s to t in G?

$L \in NL$



ST-NON-CONN is complete for co-NL.

Immerman - Szelepcsényi: ST-NON-CONN $\in$ NL.

Review non-deterministic log-space algorithm for ST-CONN:

```
Counter = 0.
current node = s
while (currentnode != t + counter < n)
    guess v
    if (currentnode, v) ∈ E
        current node ← v
        counter ++;
    else reject.
```

after loop:

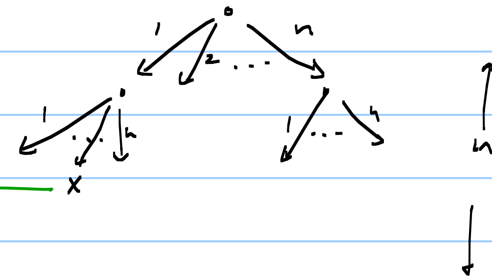
if current node = t
accept

else reject.

Each computation path in the tree is a sequence of node labels: $(v_{i_1}, v_{i_2}, \ldots, v_{i_n})$. Computation stops if $(v_{i_j}, v_{i_{j+1}}) \notin E$.

Reject if t never reached.

Accept if t is reached.

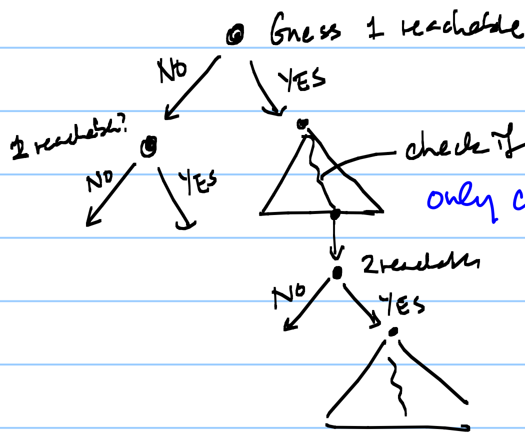


two reasons to stop:
t reached (accept).
path does not follow an edge (reject).

Now suppose we know the number of nodes reachable from s. Call this # R. One way to show that t is not reachable from s is to find R distinct nodes that are reachable from s, none of which are t:

counter = 0
For each $v = 1, \ldots, n$
  Non-deterministically guess if v is reachable from s.
  If guess = yes
    Solve STCONN (s,v) using logspace
      Guess path from s to v. If guess doesn't lead to v, reject.
      If path does lead to v
        If $(v == t)$ reject
        else counter++;
If (counter = R) accept
Else reject.

only way to accept is to reach R distinct nodes not equal to t.



only computations w/ valid paths to 1 continue.

Computation guesses a subset of the nodes.

$$\begin{bmatrix} v_1 & v_2 & \cdots & v_n \\ Y/N & Y/N & & Y/N \end{bmatrix}$$

Computation continues only if each YES guess is in fact reachable from s.

Only accept if R nodes have been reached.

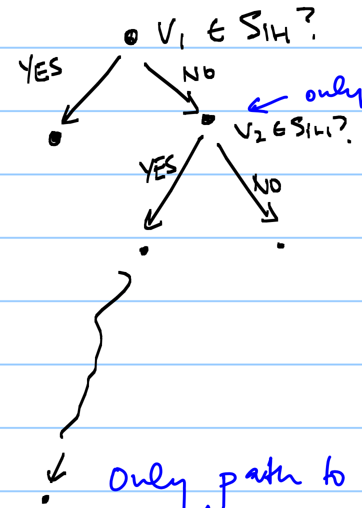
Now how to compute R?

$R(i)$ = # nodes reachable from s in $\le i$ steps.
$R(0) = 1$ (node s).

Compute $R(i+1)$ from $R(i)$ using non-determinism.
uses only $\log n$ bits
Eventually have $R(n) = R$.
Only need to keep $R(i)$ to get $R(i+1)$ $O(\log n)$ bits.

Initialize $R(i+1) = 0$
For each $v \in V$ guess if v is reachable from s in $\le i+1$ steps.
If guess = "yes"
Use NL procedure to verify path from s to v.
There will be an accepting path iff Yes guess is right.
If guess is right, increment $R(i+1)$.
If guess = "no"
Use NL procedure to verify that there is no path from s to v that uses $\le i+1$ edges.
There will be an accepting path iff No guess is right

Let S_{i+1} be the nodes reachable by path of length $\leq i+1$



← only the right guess will survive.

Only path to survive has all the right guesses.
$\Rightarrow$ has the right count for $R(i+1)$

NL procedure to verify $\exists$ path from s to v of length $\leq i+1$:

This is basically the alg for ST-CONN:

current node $= s$.
count $= 0$.
while (count $\leq i$ and current node $\neq t$)
  guess v
    (currentnode, v) $\in E$?
      NO $\rightarrow$ reject
      YES
        current node $\leftarrow v$
        count ++.
If current node $= t$
    accept
Else reject.

NL procedure to verify that there is no path from s to v in $\leq i+1$ steps.

Use $R(i)$ and the procedure outlined above to reach every node reachable from s in $\leq i$ steps. For each v' reached, verify that it has no edge to v.

Reject if v is reached
Accept otherwise.

Here's an expanded specification of this procedure:

```
Counter = 0
For each u = 1 ... n
    Non-deterministically guess if u is reachable from s in ≤ i steps.
    If guess = YES
        curr = s
        c = 0
        while (c < i and curr ≠ u)
            Non-deterministically guess a node w
            If (curr, w) is an edge
                curr ← w
                c++
            Else reject.
        If (curr ≠ u)
            reject.
        Counter ++;
        If there is an edge (u,v)
            reject.
If Counter ≠ R(i)
    reject.
```

Computation continues only if YES guess was correct.

Check for length i+1 path to v through u.

only accept if all R(i) nodes reachable from s in i steps were actually reached.