Non-deterministic Space

NSPACE(\(f(n)\)) - languages decidable by a multi-tape NTM that touches \(\leq f(n)\) squares of work tape along any computation path.

Robust non-deterministic space classes:

\(NL = NSPACE(\log n)\)

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

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

\[\log n \times \log t(n) \times |Q| \times 2^{O(t(n))}\]

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:

\(t(n) = \log n\) \(\Rightarrow NSPACE(t(n)) = NL \subseteq P\)

\(t(n) = n^k\) \(\Rightarrow NSPACE(n^k) \subseteq EXP \Rightarrow NSPACE \subseteq \text{EXP} \).
This suggests 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 ∈ NL:**

- $V = S$
- Clock = 0

**Algorithm:**

1. 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.

2. $V = v'$
3. 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 $t$: $s = v_0 \rightarrow v_1 \rightarrow v_2 \ldots \rightarrow v_n = t$

There is a sequence of guesses that will lead to $t$.

**Now prove:** $∀ L \in NL \rightarrow \exists x \cdot STCONN$

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

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

$x \in L \iff \exists$ path from $s$ to $t$ in $G_x$
On input $\alpha$, there will be $\log^2 n \times |C^{th}| \times |Q|$ 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: $S = \text{start config.}$, $t = \text{t}$.

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

Switch to: $\text{NPSPACE} = \text{PSPACE}$

$(\text{NL} \subseteq \text{SPACE} (\log^2 n))$

Immerman-Szelepcsényi (’87/’88): $\text{NL} = \text{co-NL}$

These are the opposite of what we believe to hold for time complexity classes. ($P = \text{NP}$, $\text{NP} = \text{co-NP}$)
Savitch's Theorem: \( \text{STCONN} \in \text{SPACE}(\log^2 n) \)

Corollary: \( \text{NL} \subseteq \text{SPACE}(\log^2 n) \)

Corollary: \( \text{NPSPACE} = \text{PSPACE} \)

The configuration graph for a poly-space NTM \( M \) has size \( 2^{c \log n} \). The algorithm that solves \( \text{STCONN} \) in space \( \log^2 n \) does not 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 \( 2^{c \log n} \) in \( \log^2 (2^{c \log n}) = \Theta(c \log n) \).

We can answer queries of the form: \((c,c')\) is there a single move of \( M \) that transforms \( c \) into \( c' \)?

**Proof that \( \text{STCONN} \in \text{SPACE}(\log^2 n) \):**

**Input:** \( G = (V,E) \) \& \( s, t \in V \).

**Recursive algorithm:**

\[ \text{PATH}(x, y, i) \] // is there a path from \( x \) to \( y \) of length \( \leq 2^i \)

\[ \text{if } i = 0 \text{ return } (x = y \lor (x, y) \in E); \]

\[ \text{for all nodes } z: \]

\[ \text{if } \text{PATH}(x, z, i-1) \land \text{PATH}(z, y, i-1) \text{ return true; } \]

\[ \text{else return false.} \]

**Return** \( \text{PATH}(s, t, \log n) \)

**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 record.

- $(x, y, i)$ $(x, z, i-1) \rightarrow$ next cell $(z, y, i-1)$
- $(x, y, i)$ $(z, y, i-1) \rightarrow$ pop record & return results.

ST-NON-CONN

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

Is there no path from $s$ to $t$ in $G$?

$L \in NL$ $\quad$ $\quad$ $ST-$ NON-CONN (no)

$N \quad \quad \quad \rightarrow (s, s, t)$ with a path for $s$ to $t$ $\quad$ $ST-$ NO-CONN (YES)

$Y \quad \quad \quad \rightarrow (s, s, t)$ with no path for $s$ to $t.$

$co-L$

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

Immerman–Szelelősényi: $ST-$ NON-CONN $\notin NL$.

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

Counter $= 0$.

Current node $= s$

While (current node $\neq t$ $\land$ counter $< n$)

Guess $v$

if $(current$ $node, v) \in E$

Current node $\rightarrow v$

Counter $+= 1$

else reject.

after loop:

if current node $= t$ accept

else reject.
Each computation path in the tree is a sequence of node labels: 
$(V_{i_1}, V_{i_2}, \ldots, V_{i_n})$. Compute in steps if $(V_{i_j}, V_{i_{j+1}}) \in E$. 
Reject if $t$ never reached. 
Accept if $t$ is reached.

 dois reasons to stop:
+ 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
   end to $t$. 


Now how to compute \( R \)?

\[
R(i) = \#\text{nodes reachable from } s \text{ in } \leq i \text{ steps.}
\]

\[
R(0) = 1 \quad \text{(node } s)\text{.}
\]

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 \( R(n) \) \( O(\log n) \) bits.

Initialize \( R(1) = 0 \)

For each \( v \in V \) guess if \( v \) is reachable from \( s \) in \( \leq i+1 \) steps.

If guess = "yes"

Use NL procedure to verify path from \( s \) to \( v \).
There will be an accepting path if yes guess is right.

If guess is right, increment \( R(i+1) \).

If guess = "no"

Use NL procedure to verify there is no path from \( s \) to \( v \) that uses \( \leq i+1 \) edges.
There will be an accepting path if no guess is right.

Computation guesses a subset of the nodes.

\[
\begin{bmatrix}
V_1 & V_2 & \cdots & V_n \\
Y/N & Y/N & \cdots & Y/N
\end{bmatrix}
\]

Computation continues only if each yes guess is in fact reachable from \( s \).

Only accept if \( R \) nodes have been reached.
Let $S_i$ be the nodes reachable by path of length $\leq i+1$.

**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 $v$
  
  $(\text{current node}, v) \in E$?
  
  No $\Rightarrow$ reject

  Yes
  
  current node $\leftarrow v$
  
  count $\leftarrow$ count + 1.

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 $v = 1 \ldots n$

Non-deterministically guess if $v$ is reachable from $s$ in $\leq i$ steps.

If guess = YES

    $c_{uv} = q$
    $c = 0$

    While ($c < i$ and $c_{uv} \neq w$)
        Non-deterministically guess a node $w$
        If $(c_{uv}, w)$ is an edge
            $c_{uv} = w$
            $c++$
        Else reject.
        If $(c_{uv} \neq w)$
            Reject.

    Counter ++;
    If there is an edge $(u, v)$
        Reject.

If Counter $\neq R(i)$
    Only accept if all $R(i)$ nodes reachable from $s$ in $i$ steps were actually reached.