1 Immerman-Szelepcsényi Theorem

The theorem states the nondeterministic classes of space complexity are closed under complement.

**Theorem 1** (Immerman-Szelepcsényi Theorem). For any function $S(n) \geq \log n$,

$$NSPACE(S(n)) = \text{co-NSPACE}(S(n)).$$

This theorem was surprising when it was discovered. Before it, people conjectured $NL \neq \text{co-NL}$. But the theorem shows $NL = \text{co-NL}$.

In linguistics there is a thesis saying all human languages are context-sensitive. CSL, the set of context-sensitive languages, is exactly $NSPACE(n)$. By Theorem 1 CSL = co-CSL (the complement of a context-sensitive language is also context-sensitive).

Recall that an accepting computation of a TM is a path in the configuration graph from the start configuration node to the accepting configuration node. (Here we assume there is only one accepting configuration) So a TM doesn’t accept means there are no paths from the start configuration node to the accepting configuration node.

To prove that for any $L \in NSPACE(S(n))$ there is $\overline{L} \in NSPACE(S(n))$, we need to come up with a nondeterministic algorithm that certifies $t$ is not reachable from $s$. Since the size of graph is $2^{O(S(n))}$, our nondeterministic algorithm should use space logarithmic to the graph size.

<table>
<thead>
<tr>
<th>Problem: $(s, t)$-UNCONN</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Input: Graph $G$, nodes $s, t$.</td>
</tr>
<tr>
<td>• Output: If $t$ is not reachable from $s$, output “Yes”, otherwise output “No”.</td>
</tr>
</tbody>
</table>

We construct an algorithm in NL that decides $(s, t)$-UNCONN.

Let $N$ be a nondeterministic algorithm. We say $N$ computes $f(x)$ if

(A) every non-rejecting run of $N$ outputs $f(x)$
(B) on every \( x \), at least one path accepts.

Define value \( \text{count}_\ell \) to be the number of vertices \( v \) that are reachable from \( s \) by a path of length at most \( \ell \). Initially, \( \text{count}_1 = d(s) + 1 \).

To compute \( \text{count}_\ell \), we define \( R(G, s, v, \ell, \text{count}) = \begin{cases} 1, & \text{if } v \text{ is reachable from } s \text{ in } \leq \ell \text{ steps.} \\ 0, & \text{otherwise.} \end{cases} \), where \( \text{count} \) is the number of nodes reachable from \( s \) by paths of length no more than \( \ell \).

\[\begin{align*}
\text{Algorithm } R(s, v, \ell, \text{count}) \\
&\text{for each } u \in V - \{v\} \text{ do} \\
&\quad b \leftarrow \text{guess if } u \text{ is reachable from } s \text{ in } \ell \text{ steps} \\
&\quad \text{newcount} \leftarrow \text{newcount} + b \\
&\quad \text{if } b = 1 \text{ then} \\
&\quad\quad \text{guess a path from } s \text{ to } u \text{ of length } \leq \ell. \\
&\quad\quad \text{if we don't reach } u \text{ then reject.} \\
\end{align*}\]

\(\alpha:\)
\[\begin{align*}
&\text{if } \text{newcount} = \text{count} \text{ then return } \text{“0”.} \\
&\text{if } \text{newcount} = \text{count} - 1 \text{ then} \\
&\quad \text{guess a path to } v \\
&\quad \text{if we find it then return } \text{“1” else reject.} \\
&\text{else reject.}
\end{align*}\]

The nondeterministic algorithm computing \( R(s, v, \ell, \text{count}) \) works as follows: for each vertex \( u \) other than \( v \), we guess if there is a path from \( s \) to \( u \) of length \( \ell \). Every time we make a wrong guess, we reject. At point \( \alpha \), all our previous paths were checked to be correct. If \( \text{newcount} = \text{count} \), then it means all paths of length \( \ell \) are already found previously in \( V - \{v\} \), thus \( v \) cannot be reached from \( s \) in \( \ell \) steps. If \( \text{newcount} = \text{count} - 1 \), then it is possible that the only uncounted path reaches \( v \). Thus we guess a path to \( v \) and check if it is the case. If \( \text{newcount} < \text{count} - 1 \), then it means for some \( u \) reachable from \( s \), we didn't make the guess that \( u \) is reachable.

As long as \( v \) is reachable from \( s \) within \( \ell \) steps, there is always a sequence of correct guesses that finally returns with “1". On the other hand, if \( v \) is not reachable, then all sequences of guesses are either wrong and thus rejected, or correct and thus returns with “0”.

Each variable in the algorithm uses memory \( \log n \). So the space complexity is \( \log n \).

Next we show the algorithm that computes \( \text{count}_\ell \) from \( \text{count}_{\ell - 1} \) using \( R \) as a subroutine.

\[\begin{align*}
\text{Algorithm } \#G(s, \ell, \text{count}_{\ell - 1}) \\
&\text{count}_\ell \leftarrow 0 \\
&\text{for each } v \text{ do} \\
&\quad \text{for each } u \text{ do} \\
&\quad\quad \text{if } (u, v) \in E \text{ and } R(s, u, \ell - 1, \text{count}_{\ell - 1}) \text{ then} \\
&\quad\quad\quad \text{count}_\ell \leftarrow \text{count}_\ell + 1 \\
&\quad\quad\text{break;}
\end{align*}\]
Finally, we can reuse the variables for $count_\ell$.

**Algorithm** ($s, t$)-UNCONN

```
count ← 1
for $\ell ← 1$ to $n$ do
  count ← $\#G(s, \ell, count)$
  if $R(s, t, n, count) = 1$ then return false
return true
```

Because the number of variables is constant, and the space complexity of each variable is $O(\log n)$, the whole algorithm is in NL.

## 2 Probabilistic computation

Say we have an equation $(x + y + z)^{137} - (x + 2y - z)^{137} = z^{137}$. To test if it's valid (i.e. true on all values of $x, y, z$), we can plug in random values for $x, y, z$ and see if the equation holds.

**Theorem 2** (Schwartz-Zippel-DeMillo-Lipton Lemma). If $p(x_1, \ldots, x_n)$ is a non-zero polynomial of degree $D$, and let $x_1, \ldots, x_n$ be uniformly and independently selected from set $S$, then $\Pr[p(x_1, \ldots, x_n) = 0] \leq D/|S|$. 