CSE200 Lecture Notes – Immerman-Szelepcsényi Theorem

Lecture by Russell Impagliazzo
Notes by Jiawei Gao

February 25, 2016

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 $S(n) \geq \log n$,

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

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

In linguistics there is a thesis saying all human languages are context-sensitive. CSL, the set of context-sensitive languages, is exactly $\mathsf{NSPACE}(n)$. By Theorem 1, $\mathsf{CSL} =$ 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 $L \in \mathsf{NSPACE}(S(n))$ there is $\overline{L} \in \mathsf{NSPACE}(S(n))$, 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.

> Problem: (s,t)-UNCONN
> - Input: Graph G, nodes s, t.
> - Output: If t is not reachable from s, output "Yes", otherwise output "No".

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 $count_\ell$ to be the number of vertices v that are reachable from s by a path of length at most ℓ. Initially, $count_1 = d(s) + 1$.

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

Algorithm $R(s, v, \ell, count)$

```
for each u ∈ V − {v} do
    b ← guess if u is reachable from s in ℓ steps
    newcount ← newcount + b
    if b = 1 then
        guess a path from s to u of length ≤ ℓ.
        if we don't reach u then reject.
α:
    if newcount = count then return "0".
    if newcount = count − 1 then
        guess a path to v
        if we find it then return "1" else reject.
    else reject.
```

The nondeterministic algorithm computing $R(s, v, \ell, count)$ works as follows: for each vertex u other than v, we guess if there is a path from s to u of length ℓ. Every time we make a wrong guess, we reject. At point α, all our previous paths were checked to be correct. If $newcount = count$, then it means all paths of length ℓ are already found previously in $V - \{v\}$, thus v cannot be reached from s in ℓ steps. If $newcount = count - 1$, 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 $newcount < count - 1$, 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 ℓ 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 $count_\ell$ from $count_{\ell-1}$ using R as a subroutine.

Algorithm $\#G(s, \ell, count_{\ell-1})$

```
count_ℓ ← 0
for each v do
    for each u do
        if (u, v) ∈ E and R(s, u, ℓ − 1, count_{ℓ−1}) then
            count_ℓ ← count_ℓ + 1
            break;
```

return $count_\ell$

Finally, we can reuse the variables for $count_\ell$.

Algorithm (s,t)-UNCONN

$count \leftarrow 1$
for $\ell \leftarrow 1$ **to** n **do**
  $count \leftarrow \#G(s, \ell, count)$
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 $\Pr[p(x_1,\ldots,x_n) = 0] \le D/|S|$.