

NL

## NL-Completeness

A **logspace** transducer is a TM with a read-only input tape, a write-only output tape, and a read/write work tape, in which only  $O(\log n)$  tape cells of the work tape can be used for input  $w$  of length  $n$ .

A logspace transducer  $M$  **computes** a function  $f$  if for every  $w$ ,  $M$  on  $w$  halts with  $f(w)$  on the output tape.

**Note:**  $f(w)$  can occupy at most  $O(n^k)$  space for some  $k$ , since there are only  $2^{O(\log n)}$  configurations of the transducer.

A language  $A$  is **logspace reducible** to  $B$ , write  $A \leq_L B$ , if there is a logspace computable mapping reduction from  $A$  to  $B$ .

A language  $A$  is **NL-complete** if  $A \in \text{NL}$  and every  $B \in \text{NL}$  is logspace reducible to  $A$ .

## Properties of logspace reductions

**Theorem.** *If  $A \leq_L B$  and  $B \in L$  then  $A \in L$ .*

(Comment: compute just 1 symbol of the reduction function, as needed. Recording the symbol location takes  $O(\log n)$  space.)

**Theorem.** *If  $A \leq_L B$  and  $B \in NL$  then  $A \in NL$ .*

**Corollary.** *If  $A$  is NL-complete and  $A \in L$  then  $L = NL$ .*

## NL-complete Problem

**Theorem.** *PATH is NL-complete.*

**Proof**  $PATH \in NL$ . Given an instance  $(G, s, t)$  of  $PATH$  with  $n$  nodes, repeat the following  $n - 1$  times with  $x = s$  at the beginning:

- Nondeterministically select a node  $y$  from  $1, \dots, n$ ,
- If  $(x, y)$  is in  $G$ , then set  $x$  to  $y$ . If not, reject.
- If  $y = t$ , then accept.

This method correctly decides whether  $(G, s, t) \in PATH$  and requires  $O(\log n)$  space.

## PATH is NL-complete (cont'd)

Let  $A$  be decided by a nondeterministic  $c \log n$  space machine  $N$ . We may assume that  $N$  has the unique accepting configuration for each input. Let  $x$  be an input of some length  $n$ . Define the graph  $G$  as follows:

- The nodes of  $G$  are the configurations of  $M$  on  $x$ . Here each configuration is the concatenation of the state, head positions, and the work tape contents. (Size  $\sim O(\log n)$ .)
- $s$  is the initial configuration
- $t$  is the accepting configuration.
- For every pair of nodes  $u$  and  $v$ , there is an arc from  $u$  to  $v$  if and only if  $v$  is one of the next possible configurations of  $u$ .

Then  $(G, s, t) \in PATH$  if and only if  $x \in A$ .

## Computation of $(G, s, t)$ in logspace

Let  $\ell$  be the encoding length of each configuration, in binary.  
Block 1 prints the adjacency matrix, block 2 prints indices of  $s, t$ .

```
for  $u = 0^\ell, \dots, 1^\ell$  do
  for  $v = 0^\ell, \dots, 1^\ell$  do
    if  $u$  and  $v$  are configurations then
      if  $u \Rightarrow v$  then output 1 else output 0
 $C \leftarrow 0$ ;
for  $u = 0^\ell, \dots, 1^\ell$  do
  if  $u$  is a configuration then
     $C \leftarrow C + 1$ ;
    if  $u$  = the initial config. then output " $s = C$ "
    if  $u$  = the accepting config. then output " $t = C$ "
```

The algorithm works in  $O(\ell) = O(\log n)$  space.



## NL = coNL

**Theorem.**  $\overline{PATH} \in \text{NL}$ .

**Proof** Let  $(G, s, t)$  be as in the  $PATH$  problem, where  $G$  has  $n$  nodes. For each  $i$ ,  $0 \leq i \leq n$ , define  $A_i$  to be the set of all nodes reachable from  $s$  within  $i$  steps and  $c_i = \|A_i\|$ .

Given  $c_i$  it is possible to nondeterministically enumerate all the nodes in  $A_i$  with the following  $\text{ENUMERATE}(i, c_i)$ , in  $\log n$  space:

1. Set counter  $d$  to 0;
2. for  $j = 0, \dots, n$  do the following:
  - (a) guess an  $s$ -to- $j$  path of length at most  $i$  (node by node);
  - (b) if successful increment  $d$  and output  $j$ ;
3. if  $d = c_i$  output “SUCCESSFUL”; otherwise, output “FAILURE”

### Computing $c_{i+1}$ knowing $c_i$

1. Set counter  $e$  to 0;  $\{\textit{becomes } c_{i+1}, \textit{ if successful}\}$
2. For  $j = 0, \dots, n$  do the following:  $\{\textit{Is } j \in A_{i+1} \textit{? If so, add 1 to } e\}$ 
  - (a) Set a variable  $r$  to *false*.
  - (b) Call  $\text{ENUMERATE}(i, c_i)$ . For each node  $u$  output by  $\text{ENUMERATE}$ , *check if*  $u \Rightarrow j$ ; *if so, set*  $r$  *to* *true*.
  - (c) If  $\text{ENUMERATE}$  has output “FAILURE” at the end output “FAILURE”.  
  
Otherwise, *increment*  $e$  *if and only if*  $r = \textit{true}$ .
3. Output  $e$ .



## Testing Unreachability

1. Set  $c_0$  to 1.
2. For  $i = 0, \dots, n - 1$ , compute  $c_{i+1}$  from  $c_i$ .  
(Stop without accepting in case of failure.)
3. (Check if  $t \notin A_n$  by calling  $\text{ENUMERATE}(n, c_n)$ .) Accept if the enumeration is “SUCCESSFUL” and  $t$  is not output.

The method uses only  $O(\log n)$  space.

