Computational Complexity Theory
Fall 2025

NL-completeness and NL = coNL
September 12, 2025

Lijie Chen
University of California, Berkeley
lijiechen@berkeley.edu

Today's plan & why $\mathbf{NL} = \mathbf{coNL}$ is surprising

- Quick recap: **NL** and space-bounded nondeterminism
- Logspace-reductions and NL-completeness
- Two NL-complete problems
- Overview of the Immerman–Szelepcsényi proof that $\mathbf{NL} = \mathbf{coNL}$

Today's plan & why **NL** $=$ **coNL** is surprising

- Quick recap: **NL** and space-bounded nondeterminism
- Logspace-reductions and NL-completeness
- Two NL-complete problems
- Overview of the Immerman–Szelepcsényi proof that **NL** $=$ **coNL**

Why is **NL** $=$ **coNL** *surprising?*

In contrast to time complexity (where **NP** $\stackrel{?}{=}$ co-**NP** is open), *nondeterministic space* **is** closed under complement. The proof uses *inductive counting* to reason about reachability without storing large sets.

Non-deterministic space complexity

Definition (Non-deterministic space complexity)

- For a (non-deterministic) multi-tape Turing machine M and input x, the **space complexity** of M on x is the maximum number of tape cells visited by any head during the computation, over all possible sequences of transitions.

Non-deterministic space complexity

Definition (Non-deterministic space complexity)

- For a (non-deterministic) multi-tape Turing machine M and input x, the **space complexity** of M on x is the maximum number of tape cells visited by any head during the computation, over all possible sequences of transitions.
- We say M uses space $S(n)$ if for every input x of length n, M uses at most $S(n)$ space.

Non-deterministic space complexity

Definition (Non-deterministic space complexity)

- For a (non-deterministic) multi-tape Turing machine M and input x, the **space complexity** of M on x is the maximum number of tape cells visited by any head during the computation, over all possible sequences of transitions.
- We say M uses space $S(n)$ if for every input x of length n, M uses at most $S(n)$ space.
- The input tape is **read-only** and does not count towards space usage.

Non-deterministic space complexity

Definition (Non-deterministic space complexity)

- For a (non-deterministic) multi-tape Turing machine M and input x, the **space complexity** of M on x is the maximum number of tape cells visited by any head during the computation, over all possible sequences of transitions.
- We say M uses space $S(n)$ if for every input x of length n, M uses at most $S(n)$ space.
- The input tape is **read-only** and does not count towards space usage.
- There is a designated **output tape** which is **write-only** and does not count towards space usage.

Non-deterministic space complexity

Definition (Non-deterministic space complexity)

- For a (non-deterministic) multi-tape Turing machine M and input x, the **space complexity** of M on x is the maximum number of tape cells visited by any head during the computation, over all possible sequences of transitions.
- We say M uses space $S(n)$ if for every input x of length n, M uses at most $S(n)$ space.
- The input tape is **read-only** and does not count towards space usage.
- There is a designated **output tape** which is **write-only** and does not count towards space usage.
- Only the **work tapes** count towards space complexity.

Recall: **NL** (nondeterministic logspace)

Fix a standard multi-tape TM model with a read-only input tape.

$\mathbf{NL} = \mathsf{NSPACE}(\log n)$.

- Space counts only the work tapes; the output tape is write-only.
- Deterministic logspace: $\mathbf{L} = \mathsf{SPACE}(\log n)$.
- Canonical complete problem: directed s–t reachability (STCONN).

Logspace-reductions and NL-completeness

Definition (Logspace many-one reduction)

A function f is a logspace reduction if f is computed by a deterministic TM using $O(\log n)$ space, and

$$x \in L \iff f(x) \in L'.$$

Definition (NL-complete)

A language A is *NL-hard* if every $L \in \mathbf{NL}$ reduces to A via a logspace many-one reduction.
It is *NL-complete* if $A \in \mathbf{NL}$ and A is NL-hard.

Remark

We often use configuration graphs of logspace NTMs to prove hardness.

Recall: Configuration graph

- For a fixed machine M and input x, a *configuration* encodes the state, heads, and work-tape contents.

Recall: Configuration graph

- For a fixed machine M and input x, a *configuration* encodes the state, heads, and work-tape contents.
- If M uses $s(n)$ space, the number of distinct configurations is

$$N = 2^{O(s(n))}.$$

Recall: Configuration graph

- For a fixed machine M and input x, a *configuration* encodes the state, heads, and work-tape contents.
- If M uses $s(n)$ space, the number of distinct configurations is

$$N = 2^{O(s(n))}.$$

- Build the directed graph $G_{M,x}$ whose nodes are configurations and whose edges represent one valid move.

Recall: Configuration graph

- For a fixed machine M and input x, a *configuration* encodes the state, heads, and work-tape contents.
- If M uses $s(n)$ space, the number of distinct configurations is

$$N = 2^{O(s(n))}.$$

- Build the directed graph $G_{M,x}$ whose nodes are configurations and whose edges represent one valid move.
- M accepts x iff there exists a path from the start configuration c_{start} to some c_{acc}.

Recall: Configuration graph

- For a fixed machine M and input x, a *configuration* encodes the state, heads, and work-tape contents.
- If M uses $s(n)$ space, the number of distinct configurations is

$$N = 2^{O(s(n))}.$$

- Build the directed graph $G_{M,x}$ whose nodes are configurations and whose edges represent one valid move.
- M accepts x iff there exists a path from the start configuration c_{start} to some c_{acc}.
- Any accepting path has length at most N (no need to repeat configurations).

NL-complete example #1: STCONN

Problem

Input: directed graph $G = (V, E)$, nodes $s, t \in V$.
Question: is there a path from s to t?

- Membership: guess the path node-by-node; keep only the current node and a counter $\leqslant |V|$ in $O(\log n)$ space.
- Hardness: see the whiteboard!

NL-complete example #2: NFA non-emptiness

Problem
Input: an NFA $\mathcal{A}$.
Question: is $L(\mathcal{A}) \neq \emptyset$?

- Membership: guess a path from a start state to some accepting state; store only the current state and step counter.
- Hardness: see the whiteboard!

Alternative definition of NL

A non-deterministic $O(\log n)$-space Turing machine makes a sequence of $O(2^{O(\log n)}) = O(\text{poly}(n))$ choices on the fly.

An alternative definition of NL treats such a sequence of choices as a witness; this is similar to the proof-verifier viewpoint of NP.

Definition (Alternative definition of NL)

A language L is in **NL** if and only if there exists a constant c and a deterministic $O(\log n)$-space Turing machine $M(x, w)$ that takes x and a witness w such that:

- $M(x, w)$ has *streaming access* to the witness w.

Alternative definition of NL

A non-deterministic $O(\log n)$-space Turing machine makes a sequence of $O(2^{O(\log n)}) = O(\text{poly}(n))$ choices on the fly.

An alternative definition of NL treats such a sequence of choices as a witness; this is similar to the proof-verifier viewpoint of NP.

Definition (Alternative definition of NL)

A language L is in **NL** if and only if there exists a constant c and a deterministic $O(\log n)$-space Turing machine $M(x, w)$ that takes x and a witness w such that:

- $M(x, w)$ has *streaming access* to the witness w.
- For every $x \in L$, there exists a witness w with $|w| \leqslant n^c$ such that $M(x, w)$ accepts.

Alternative definition of NL

A non-deterministic $O(\log n)$-space Turing machine makes a sequence of $O(2^{O(\log n)}) = O(\text{poly}(n))$ choices on the fly.

An alternative definition of NL treats such a sequence of choices as a witness; this is similar to the proof-verifier viewpoint of NP.

Definition (Alternative definition of NL)

A language L is in **NL** if and only if there exists a constant c and a deterministic $O(\log n)$-space Turing machine $M(x, w)$ that takes x and a witness w such that:

- $M(x, w)$ has *streaming access* to the witness w.
- For every $x \in L$, there exists a witness w with $|w| \leqslant n^c$ such that $M(x, w)$ accepts.
- For every $x \notin L$, for all witnesses w with $|w| \leqslant n^c$, $M(x, w)$ rejects.

The Immerman–Szelepcsényi theorem

Theorem

For $s(n) \geqslant \log n, \quad NSPACE(s(n)) = co\text{-}NSPACE(s(n)).$
In particular, $\quad \mathbf{NL} = \mathbf{coNL}.$

Proof: see the whiteboard!