The Immerman–Szelepcs'enyi Theorem

Édouard Bonnet Codex 5.6 and 6

Abstract

We prove NL = coNL by inductive counting on finite configuration graphs. The proof constructs a nondeterministic Turing machine that generates paths and census witnesses sequentially, halts on every branch, and uses logarithmic space. The argument includes correctness of the count sequence and space bounds for the compiled counting machine.

The languages are sets of finite binary words. The complexity classes and machine models are those of lax-434930. The annotations connect each definition, statement, and proof below to its formal counterpart.

1 Configurations and counting certificates

Definition 1 (Bounded reachability)**.** A directed graph on N vertices is given by its Boolean adjacency matrix. A walk may repeat vertices. The recursive reachability test divides the length bound in two and enumerates possible middle vertices.

Definition 2 (Bounded machine configurations)**.** A configuration using at most s work cells consists of a control state, an input-head position, a work-head position, and s work symbols. Cells beyond this prefix are blank.

Definition 3 (The bounded configuration graph)**.** Vertices are configurations whose work tape has a fixed finite bound. Edges are machine transitions. Acceptance is reachability from the initial configuration to a terminal accepting configuration. The search predicate replaces reachability by the recursive finite graph test.

Definition 4 (Inductive counting certificates)**.** The kth reachable layer contains vertices at distance at most k from the source. A census lists distinct reachable vertices and their path witnesses. Given the correct census size, a negative classification checks that no listed vertex has an edge to the target. A counting step classifies every vertex and counts the positive answers. Write R_k for the vertices reachable from the source a in at most k steps. A census S for count c satisfies $|S| = c$ and $S \subseteq R_k$. A negative classification of v also requires $v \neq a$ and no edge from S to v. A trace starts at count one and supplies a valid counting step for each $k < N$. A nonreachability certificate ends with a census omitting the target.

2 Correctness of inductive counting

Lemma 5 (Exact censuses enumerate the whole layer)**.** *A census of reachable vertices with the correct cardinality contains every vertex of the layer. Distinctness is enforced by the finite-set representation.*

Proof. Every member of the census belongs to the reachable layer. This inclusion and equality of the two finite cardinalities imply equality of the finite sets. □

Lemma 6 (The next reachable layer)**.** *A vertex belongs to the next layer precisely when it is the source or has an incoming edge from the current layer. In symbols,* $v \in R_{k+1} \iff v = a$ *or* $\exists u \in R_k,\ G(u, v) = 1$.

Proof. A walk of length at most $k + 1$ is either the zero-length walk at the source or has a last edge whose prefix has length at most k. Conversely, append the indicated edge to a witnessing prefix walk. □

Lemma 7 (Correctness of vertex classification)**.** *With the correct current count, a vertex has a valid positive or negative classification exactly according to its membership in the next layer.*

Proof. A positive classification is precisely a bounded walk into the next layer. For a negative classification, exactness of the census says that all vertices in the current layer have been examined. The successor-layer characterization then shows that the source test and the absence of an incoming edge are precisely nonmembership in the next layer. □

Lemma 8 (Correctness of one counting step)**.** *Starting from the correct layer size, valid classifications produce exactly the size of the next layer. Such classifications exist.*

Proof. Correct classification identifies the vertices marked true with the next reachable layer, so their cardinalities agree. For existence, classify each vertex by its actual membership in that layer and apply the classification equivalence. □

Lemma 9 (Correctness of the count sequence)**.** *The initial layer consists of the source. Inductive counting certifies exactly the successive layer sizes, including the final size after* N *steps.*

Proof. The initial layer is the singleton source, so its size is one. Induction using correctness of a counting step identifies every subsequent count. Conversely, the actual layer sizes satisfy the initial condition and each required step. □

Lemma 10 (Correctness of nonreachability certificates)**.** *A valid sequence of counts and a final census omitting the target exist exactly when the target is unreachable from the source.*

Proof. In a graph with N vertices, every reachable target has a walk of length less than N. Thus the final layer is the set of all reachable vertices. Correctness of the trace fixes the final count, and an exact final census equals this layer. Such a census omits the target exactly when the target is unreachable. □

3 The counting machine and complementation

Definition 11 (Counting on the configuration graph)**.** For every terminal accepting configuration, certify that it cannot be reached from the initial configuration. The implementation must generate the witnesses sequentially; this predicate alone makes no space claim.

Lemma 12 (Correctness of rejection by counting)**.** *On a space-bounded computation, counting certifies rejection precisely when there is no accepting computation.*

Proof. For runs obeying the space bound, acceptance is equivalent to reachability between the corresponding bounded configurations. Apply nonreachability correctness to every terminal accepting configuration and negate the existential acceptance condition. □

Lemma 13 (Logarithmic space for inductive counting)**.** *For a machine using logarithmic space, a nondeterministic machine implements its configuration counting predicate in logarithmic space. Vertices and bounded paths are generated sequentially. Every branch halts. More precisely, if $c > 0$ and M uses at most $c\,\ell(|w|)$ work cells on every input, there are $d > 0$ and a machine D such that, for every w, all branches of D halt, D accepts exactly the counting-rejection predicate for M at bound $c\,\ell(|w|)$, and D uses at most $d\,\ell(|w|)$ work cells. Here ℓ is the logarithmic bound in the unchanged class definitions.*

Proof. The program first counts the input length in binary and chooses a logarithmic register width. It enumerates possible accepting configurations and uses inductive counting to certify nonreachability. Vertex names, depths, and counts occupy a fixed collection of bounded binary registers; path and census witnesses are generated sequentially. The proved loop invariants establish soundness, existence of an accepting branch for every valid certificate, and termination of all branches. The configuration-graph query is implemented by the checked encoding and stack routines. A verified compiler translates each register operation into the finite Turing-machine model. The bounds cover the input counter, all scratch registers, and compiler microsteps, giving the stated logarithmic work-space bound. □

Lemma 14 (Complementing a logarithmic space language)**.** *The complement of a language in* NL *also belongs to* NL*.*

Proof. Choose the logarithmic-space machine deciding the given language. Its space bound supplies the hypothesis of the counting-machine construction. Rejection correctness identifies the new machine's language with the complement; its termination and space bound give membership in NL. □

Theorem 15 (The Immerman–Szelepcsényi theorem)**.** *Nondeterministic logarithmic space is closed under complement:* NL = coNL*.*

Proof. By definition, a language belongs to coNL exactly when its complement belongs to NL. Apply complement closure in both directions and simplify the double complement. □