Recall the given relation between classes:

\[ L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP \subseteq NEXP \subseteq EXPSPACE \subseteq \ldots \]

These classes are defined for two reasons: First to study computation under resources such as time and space, but also just as importantly to help us understand the computational complexity of concrete computational problems that we are interested in. It turns out that all the classes above have natural complete problems.

Recalling the notion of NP-completeness, this is based on polynomial time reduction. This is however not usable for "lesser" classes such as NL and P, so a weaker notion of reductions is needed.

**Definition 1.** The language \( L_f \) associated with a function \( f : \Sigma^* \rightarrow \Delta^* \) is

\[ L_f = \{ <x, \text{bin}(j), y> | x \in \Sigma^*, j \in \mathbb{N}, y \in \Delta, f(x)_j = y \} \]

\( L_f \) will here work as a substitute for the function \( f \).

**Definition 2.** We say

- \( f \) is P-computable if: \( |f(x)| = n^{O(1)} \) and \( L_f \in P \).
- \( f \) is L-computable if: \( |f(x)| = n^{O(1)} \) and \( L_f \in L \).

**Definition 3.** Let \( A \subseteq \Sigma^* \) and \( B \in \Delta^* \) be languages.

We say \( A \leq_p^m B \) (p stands for polynomial time, m stands for “many-one”, i.e. many instances may be mapped to one instance) if there is a P-computable \( f \) so that \( \forall x \in \Sigma^* \, , \, x \in A \iff f(x) \in B \)

We say \( A \leq_{m}^{log} B \) if there is an L-computable \( f \) so that \( \forall x \in \Sigma^* \, , \, x \in A \iff f(x) \in B \)

Let \( C \) be a class of languages.

**Definition 4.** \( L \) is \( \leq_{m}^{log} \)-hard for \( C \) if for all \( L' \in C : L' \leq_{m}^{log} L \).

**Definition 5.** \( L \) is \( \leq_{m}^{log} \)-complete for \( C \) if \( L \) is \( \leq_{m}^{log} \)-hard for \( C \) , \( L \in C \).

**Lemma 6.** (Transitivity) If \( A \leq_{m}^{log} B \) and \( B \leq_{m}^{log} C \) then \( A \leq_{m}^{log} C \)

**Proof.** Let the reductions be given by L-computable functions \( f \) and \( g \), i.e. we have \( L_f \in L \) and \( L_g \in L \). Define \( h(x) = g(f(x)) \) , and let \( M_f, M_g \) be logspace machines for \( L_f, L_g \). We must show that \( h \) is L-computable.

Note that since we have limited space available, we can’t just compute and put \( f(x) \) on a tape, as an intermediate result. We will now construct a logspace machine \( M_h \) for \( L_h \).
On input \(< x, \text{bin}(j), y >\), \(M_h\) will simulate \(M_g\) with a \textit{fictitious} input tape containing \(< f(x), \text{bin}(j), y >\). We will maintain the position of the tapehead on the fictitious tape by a counter, stored on the work tape.

Whenever \(M_g\) requires the current symbol on the fictitious input tape, \(M_h\) will run the machine \(M_f\) to compute the symbol. \(M_f\) will be able to read \(x\) from the input tape of \(M_h\).

Space analysis on input \(x\):

- for \(M_g\): \(O(\log(|f(x)|)) = O(\log(|x|^{O(1)})) = O(\log(|x|)).\)
- for counter: \(O(\log(|f(x)|)) = O(\log(|x|^{O(1)})) = O(\log(|x|)).\)
- for \(M_f\): \(O(\log(|x|)).\)

Thus the space used by \(M_h\) is \(O(\log(|x|)).\)

We know that NP is closed under reductions; so is all the other classes we have considered.

Lemma 7. \textit{Closure under reductions}

If \(B \in \mathcal{C}\) and \(A \preceq_m^{\log} B\) then \(A \in \mathcal{C}\), for \(\mathcal{C} = L, NL, P, ...\),

Example of log-complete problem in NL:

Definition 8. STCON

Instance: Directed graph \(G = (V, E)\), \(s, t \in V\)

Question: Is there a path from \(s\) to \(t\)?

Theorem 9. STCON is complete for NL

Proof.  
- \(\text{STCON} \in \text{NL}\):

  For this we have the following algorithm:
  1. \(v := s\)
  2. for \(i = 0\) to \(n - 1\)
  3. guess \(u \in V\)
  4. if \((v, u) \notin E\), reject
  5. if \(u = t\), accept
  6. \(v = u\)

  Space usage: \(O(\log n)\), since we only need to store node index \(v\), counter \(i\), etc.

- \(\text{NL-hardness}\): Let \(L \in \text{NL}\), and let \(M\) be a nondeterministic \(O(\log n)\) space bounded machine for \(L\).

  We will assume that \(M\) has a \textit{unique} accepting configuration. This we can easily do by modifying \(M\), to clearing the work tape and place tape heads at cell 1, before accepting.

  Reduction: On input \(x\), output:

  \(G\): configuration graph of \(M\) on input \(x\)

  \(s\): initial configuration
t: accept configuration

Since $M$ is $O(\log n)$ space bounded, each configuration is encoded by a string of length $O(\log n)$. We can thus output the configuration graph, by looping over check pairs of string encoding configurations, and check if $M$ can go from one configuration to the next in 1 step. This can be done in $O(\log n)$ space.

\[\]

**Theorem 10.** (Immerman-Szelepcsényi) Let $S(n) \geq \log(n)$. Then

\[
\text{co-NSPACE}(S(n)) = \text{NSPACE}(S(n))
\]

(recall that $\text{co-}C = \{L|L \in C\}$)

Proof. We will prove that $\text{STCON} \in \text{NL}$. Then the statement of the theorem follows by running the NL algorithm on the configuration graph of a $O(S(n))$ space bounded nondeterministic machine.

We first note that simply switching accept and reject in the algorithm from the proof of STCON is in NL does not work: this would not compute the complement of STCON.

Crucial idea:

Let $V_i$ = set of vertices reachable from $s$ in $\leq i$ steps, and let $C_i = |V_i|$. If we are given $C_i$, then we can nondeterministically enumerate the vertices of $V_i$ and afterwards we can know for sure if we enumerated exactly the vertices in $V_i$ as follows:

1. $\text{count} := 0$
2. for $u \in V$
3.  nondeterministically guess a path from $s$ to $u$ of length $\leq i$, vertex by vertex.
   (done with earlier shown algorithm)
4.  if found
5.    $\text{count} := \text{count} + 1$
6.  if $\text{count} = C_i$, correct enumeration!

Algorithm for $\text{STCON}$:

1. $C_{\text{cur}} := 1$  //$C_0 = 1$
2. for $i = 1$ to $n - 1$  //compute $C_i$, given $C_{i-1}$
3.   $C_{\text{prev}} := C_{\text{cur}}$, $C_{\text{cur}} = 0$
4.   for all $v \in V$  //test if $v \in V_i$
5.    $\text{count} := 0$, $\text{include} := \text{false}$
6.   for all $u \in V$  //enumerate $V_{i-1}$
7.    nondeterministically guess a path from $s$ to $u$ of length $\leq i - 1$, vertex by vertex
8.    if (found)
9.       $\text{count} := \text{count} + 1$
10.      if $((u, v) \in E$ or $u = v)$
11.     $\text{include} := \text{true}$
12.    if $\text{count} = C_{\text{prev}}$ → reject  // Enumeration of $V_{i-1}$ failed
13.   if (include) $C_{\text{cur}} := C_{\text{cur}} + 1$
// At this point we have computed $C_{n-1}$ correctly.
14. count := 0
15. for all $u \in V$ // Enumerate $V_{n-1}$ to see if $t \in V_{n-1}$
16. nondeterministically guess a path from $s$ to $u$ of length $\leq n - 1$ vertex by vertex
17. if( found )
18. count := count + 1
19. if( $u = t$ ) $\rightarrow$ reject
20. if( count $= C_{cur} \rightarrow$ accept ) // If enumeration was successful we can tell for sure if $t \not\in V_{n-1}$

**Boolean Circuits**

- Inputs: $x_1, \ldots, x_n \in \{0, 1\}$
- Circuit: Directed acyclic graph. Nodes correspond to *gates*, edges to *wires*.
- indegree 0 nodes: labeled by variables
- indegree 1 nodes: labeled by 1
- indegree 2 nodes: labeled by $\lor$, $\land$
- specified output node

$C$ computes a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$.

**Circuit value problem, CVP:**

- Instance: Circuit $C$ on $n$ inputs and concrete input $x \in \{0, 1\}^n$.
- Question: Is $C(x) = 1$?

**Circuit-SAT:**

- Instance: Circuit $C$.
- Question: Does there exist $x \in 0,1^n$ such that $C(x) = 1$?

**Theorem 11.** $CVP$ is $\leq_{m}^{log} \text{-complete for } P$

*Proof.* To show that $CVP \in P$, given a circuit together with an input, we simple evaluate the circuit bottom up.

To show $\leq_{m}^{log}$-hardness, let $L \in P$, and let $M$ be $p(n)$ time bounded *single-tape* TM accepting $L$. Build circuit by tableau method.
In each coordinate \((i, t)\) we encode the content of the tape in cell \(i\), whether the tape head is scanning the cell, and if so, the state of the machine.

Now given this information we can compute the contents of the cell just by looking at the 3 previous cells below, in the neighbor positions (see figure).

\[ \textbf{Theorem 12.} \quad \text{Circuit-SAT is } \leq_{m}^{\log} \text{complete for NP} \]

\[ \text{Proof.} \quad \text{This can again by proven by the tableau method. We have additional inputs to describe the nondeterministic choices. These are fed to each coordinate. The reduction will on input } x \text{ construct this circuit, and then hardwire } x, \text{ leaving the inputs for the nondeterministic choices be the only inputs left.} \]