Hierarchy Theorem

Proof
The properties of NL and coNL are similar to those of NP and coNP:

- if \( L \) is NL-complete then \( \overline{L} \) is coNL-complete
- if a coNL-complete problem belongs to NL then NL = coNL

Reachability is NL-complete.
Therefore it is enough to show that No-Reachability is in NL.

In order to do this, we have to find a non-deterministic algorithm that proves in log-space that there is a path between two specified vertices in a graph.

NL and coNL

For a language \( L \) over an alphabet \( \Sigma \), we denote \( \overline{L} \) the complement of \( L \), the language \( \Sigma^* - L \).

Definition
The class of languages \( L \) such that \( \overline{L} \) can be solved by a non-deterministic log-space Turing machine verifier is called coNL.

Theorem (Immerman)
\[ \text{NL} = \text{coNL} \]

Counting the number of reachable vertices
Given a graph \( G \) and two its vertices \( s \) and \( t \):
let \( m \) be the number of vertices of \( G \)
First, we count the number \( c \) of vertices connected to \( s \) with a path of length at most \( i \), and \( c_i = |A_i| \)
Clearly, \( c_i = A_1 \subseteq A_2 \subseteq \ldots \subseteq A_i \) and \( c_i = c \)
We compute the numbers \( c_i, c_{i-1}, \ldots, c_1 \) inductively

Checking Reachability
Given \( G, s, t \) and \( c \):

- set \( m = c \)
- for every vertex \( v \) from \( G \) non-deterministically do or not do:
  - check whether or not \( v \) is reachable from \( s \) using non-deterministic walk
  - if not then reject
  - if yes then set \( m = m - 1 \)
  - if there is the edge \((w,v)\) then set \( c_{m} = c_{m} + 1 \) and leave the loop
- if \( m = 0 \) reject
- accept
Complexity Classes

We know a number of complexity classes and how they relate to each other:

\[ L \subseteq NL \subseteq P \subseteq NP \subseteq coNP \subseteq PSPACE \]

However, we do not know if any of them are different.

The amount of time/space available determines determinism and nondeterminism and known to be extremely difficult.

Complexity classes can be distinguished using another parameter:

**Hierachy Theorem**

**Theorem**

For any space constructable function \( f : \mathbb{N} \rightarrow \mathbb{N} \), there exists a language \( L \) that is decidable in space \( O(f(n)) \), but not in space \( o(f(n)) \).

**Corollary**

If \( f(n) \) and \( g(n) \) are space constructable functions, and \( f(n) = o(g(n)) \), then \( \text{SPACE}(f) \subseteq \text{SPACE}(g) \).

**Corollary**

\( L = \text{PSPACE} \)

**Corollary**

\( NL = \text{PSPACE} \)

**Space Constructable Functions**

**Definition**

A function \( f : \mathbb{N} \rightarrow \mathbb{N} \), where \( f(n) \geq \log n \), is called space constructable, if the function that maps \( 1^n \) to the binary representation of \( f(n) \) is computable in space \( O(f(n)) \).

**Examples**

- polynomials
- \( n \log n \)
- \( 2^n \), \( 3^n \), ...
- ...

**Proof Idea**

Diagonalization Method:

- apply a Turing Machine to its own description
- revert the answer
- get a contradiction

In our case, a contradiction can be with the claim that something is computable within \( o(f(n)) \).

Let \( L = \{ M | M \text{ does not accept } \text{"M\text," in } f(n) \text{ space} \} \)

**Proof**

In order to kick in asymptotics, change the language

\( L = \{ M | M \text{ does not accept } \text{"M\text," in } f(n) \text{ space} \} \)

The following algorithm decides \( L \) in \( O(f(n)) \)

On input \( x \):

- Let \( n \) be the length of \( x \)
- Compute \( f(n) \) and mark off this much tape. If later stages ever attempt to use more space, reject
- If \( x \) is not of the form \( M10^n \) for some \( M \), reject
- Simulate \( M \) on \( x \) while counting the number of steps used in the simulation. If the count ever exceeds \( 2^{n^2} \), accept
- If \( M \) accepts, reject. If \( M \) rejects, accept
The key stage is the simulation of \( M \)

Our algorithm simulates \( M \) with some loss of efficiency, because the alphabet of \( M \) can be arbitrary.

If \( M \) works in \( g(n) \) space then our algorithm simulates \( M \) using \( bg(n) \) space, where \( b \) is a constant factor depending on \( M \).

Thus, \( bg(n) \leq f(n) \).

Clearly, this algorithm works in \( O(f(n)) \) space.

Suppose that there exists a TM \( M \) deciding \( L \) in space \( g(n) = o(f(n)) \).

We can simulate \( M \) using \( bg(n) \) space.

There is \( n_0 \) such that for all inputs \( x \) with \( |x| > n_0 \) we have \( bg(|x|) < f(|x|) \).

Consider \( M(M(1^n)) \).

Since \( |M(1^n)| > n_0 \), the simulation of \( M \) either accepts or rejects on this input in space \( f(n) \):

- If the simulation accepts then \( M(M(1^n)) \) does not accept
- If the simulation rejects then \( M(M(1^n)) \) accepts

**Time Hierarchy Theorem**

**Theorem**
For any time constructable function \( f : \mathbb{N} \rightarrow \mathbb{N} \), there exists a language \( L \) that is decidable in time \( O(f(n)) \), but not in time \( f(n) \).

**Corollary**
If \( f(n) \) and \( g(n) \) are space constructable functions, and \( g(n) = \frac{f(n)}{\log f(n)} \) then \( \text{TIME}[f] \neq \text{TIME}[g] \).

**The Class \( \text{EXPTIME} \)**

**Definition**
\[
\text{EXPTIME} = \bigcup_{c \in \mathbb{N}} \text{TIME}[2^{cn}]
\]

**Corollary**
\( P \neq \text{EXPTIME} \)
ERROR: undefined
OFFENDING COMMAND:

STACK: