# RELATIONS BETWEEN COMPLEXITY CLASSES

- Basic requirements for complexity classes
- Complexity classes
- Hierarchy theorems
- Reachability method
- Class inclusions
- Simulating nondeterministic space
- Closure under complement

(C. Papadimitriou: *Computational complexity*, Chapter 7)





## 1. Basic Requirements for Complexity Classes

A complexity class is specified by

- model of computation (multi-string TMs)
- mode of computation (deterministic, nondeterministic,...)
- resource (time, space, ...)
- bound (function $f$)

A *complexity class* is the set of all *languages* decided by some multi-string Turing machine $M$ operating in the appropriate mode, and such that, for any input $x$, $M$ expends at most $f(|x|)$ units of the specified resource.





### Reasonable bound functions

**Definition.** A function $f : \mathbf{N} \to \mathbf{N}$ is a *proper complexity function* if $f$ is nondecreasing and there is a $k$-string TM $M_f$ with input and output such that on any input $x$,

1. $M_f(x) = \sqcap^{f(|x|)}$ where $\sqcap$ is a *quasi-blank* symbol,
2. $M_f$ halts after $O(|x| + f(|x|))$ steps, and
3. $M_f$ uses $O(f(|x|))$ space besides its input.

- Examples of proper complexity functions $f(n)$:

$$c,\ n,\ \lceil \log n \rceil,\ \log^2 n,\ n \log n,\ n^2,\ n^3 + 3n,\ 2^n,\ \sqrt{n},\ n!,\ \ldots$$

- If $f$ and $g$ are proper, so are, e.g., $f + g$, $f \cdot g$, $2^g$.
- Only proper complexity functions will be used as bounds.





### Precise Turing machines

**Definition.** Let $M$ be a deterministic/nondeterministic multi-string Turing machine (with or without input and output).

Machine $M$ is *precise* if there are functions $f$ and $g$ such that for every $n \geq 0$, for every input $x$ of length $n$, and for every computation of $M$,

1. $M$ halts after precisely $f(|x|)$ steps and
2. all of its strings (except those reserved for input and output whenever present) are at halting of length precisely $g(|x|)$.

(Precise bounds will be convenient in various simulation results).

## Simulating TMs with precise TMs

**Proposition.** Let $M$ be a deterministic or nondeterministic TM deciding a language $L$ within time/space $f(n)$ where $f$ is proper.

Then there is a precise TM $M'$ which decides $L$ in time/space $\mathrm{O}(f(n))$.

Proof sketch.

The simulating machine $M'$

1. computes a yardstick/alarm clock $\sqcap^{f(|x|)}$ using $M_f$ and
2. simulates $M$ for exactly $f(|x|)$ steps *or*
   simulates $M$ using exactly $f(|x|)$ units of space.





## 2. Complexity Classes

- Given a proper complexity function $f$, we obtain following classes:

  $\mathbf{TIME}(f)$ (deterministic time)

  $\mathbf{NTIME}(f)$ (nondeterministic time)

  $\mathbf{SPACE}(f)$ (deterministic space)

  $\mathbf{NSPACE}(f)$ (nondeterministic space)

- The bound $f$ can be a family of functions parameterized by a non-negative integer $k$; meaning the union of all individual classes.

  The most important are: $\mathbf{TIME}(n^k) = \bigcup_{j>0} \mathbf{TIME}(n^j)$

  $\mathbf{NTIME}(n^k) = \bigcup_{j>0} \mathbf{NTIME}(n^j)$





## Variety of complexity classes

$$
\begin{aligned}
\mathbf{P} &= \mathbf{TIME}(n^k) \\
\mathbf{NP} &= \mathbf{NTIME}(n^k) \\
\mathbf{PSPACE} &= \mathbf{SPACE}(n^k) \\
\mathbf{NPSPACE} &= \mathbf{NSPACE}(n^k) \\
\mathbf{EXP} &= \mathbf{TIME}(2^{n^k}) \\
\mathbf{L} &= \mathbf{SPACE}(\log(n)) \\
\mathbf{NL} &= \mathbf{NSPACE}(\log(n))
\end{aligned}
$$

The relationships of these classes will be studied in the sequel.





## Complements of decision problems

- Given an alphabet $\Sigma$ and a language $L \subseteq \Sigma^*$, the *complement* of $L$

  $$\overline{L} = \Sigma^* - L.$$

- For a decision problem A, the answer for the complement "A COMPLEMENT" is "yes" iff the answer for A is "no".

**Example.** SAT COMPLEMENT: given a Boolean expression $\phi$ in CNF, is $\phi$ unsatisfiable?

**Example.** REACHABILITY COMPLEMENT: given a graph $(V,E)$ and nodes $v,u \in V$, is it the case that there is no path from $v$ to $u$?

## Closure under Complement

- For any complexity class $C$, $\mathbf{co}C$ denotes the class

$$\{\overline{L} \mid L \in C\}.$$

- All deterministic time and space complexity classes are closed under complement. Hence, e.g., $\mathbf{P} = \mathbf{coP}$.

  Proof. Exchange "yes" and "no" states of the deciding machine.

- The same holds for nondeterministic *space* complexity classes (to be shown in the sequel).

- An important open question: are nondeterministic *time* complexity classes closed under complement? For instance, $\mathbf{NP} = \mathbf{coNP}$?





# 3. Hierarchy Theorems

- We derive a quantitative hierarchy result:

  with sufficiently greater time allocation, Turing machines are able to perform more complex computational tasks.

- For a proper complexity function $f(n) \geq n$, define

$$H_f = \{M;x \mid M \text{ accepts input } x \text{ after at most } f(|x|) \text{ steps}\}.$$

- Thus $H_f$ is the time-bounded version of $H$, i.e. the language of the HALTING problem.





## Upper bound for $H_f$

**Lemma.** $H_f \in \mathbf{TIME}((f(n))^3)$.

Proof sketch.

A 4-string machine $U_f$ deciding $H_f$ in time $f(n)^3$ is based on

(i) the universal Turing machine $U$,

(ii) the single-string simulator of a multi-string machine,

(iii) the linear speedup machine, and

(iv) the machine $M_f$ computing the yardstick of length $f(n)$ where $n$ is the length of the input $x$.





Proof—cont'd.

The machine $U_f$ operates as follows:

1. $M_f$ computes the alarm clock $\sqcap^{f(|x|)}$ for $M$ (string 4).
2. The description of $M$ is copied on string 3 and string 2 initialized to encode the initial state $s$ and string 1 the input $\triangleright x$.
3. Then $U_f$ simulates $M$ and advances the alarm clock. If $U_f$ finds out that $M$ accepts input $x$ within $f(|x|)$ steps, then $U_f$ accepts, but if the alarm clock expires, then $U_f$ rejects.

Observations:

- Since $M$ is simulated using a single string, each simulation step takes $\mathrm{O}(f(n)^2)$ time.
- The total running time is $\mathrm{O}(f(n)^3)$ for $f(|x|)$ steps.

## Lower bound for $H_f$

**Lemma.** $H_f \notin \mathbf{TIME}(f(\lfloor \frac{n}{2} \rfloor))$

Proof sketch.

- Suppose there is a TM $M_{H_f}$ that decides $H_f$ in time $f(\lfloor \frac{n}{2} \rfloor)$.
- Consider $D_f(M)$: **if** $M_{H_f}(M;M) =$ "yes" **then** "no" **else** "yes".

  Thus $D_f$ on input $M$ runs in time $f(\lfloor \frac{2|M|+1}{2} \rfloor) = f(|M|)$.
- If $D_f(D_f) =$ "yes", then $M_{H_f}(D_f, D_f) =$ "no", hence, $D_f;D_f \notin H_f$ and $D_f$ fails to accept input $D_f$ within $f(|D_f|)$ steps, i.e. $D_f(D_f) =$ "no", a contradiction.
- Hence, $D_f(D_f) \neq$ "yes". Then $D_f(D_f) =$ "no" and $M_{H_f}(D_f, D_f) =$ "yes". Therefore, $D_f;D_f \in H_f$, and $D_f$ accepts input $D_f$ within $f(|D_f|)$ steps, i.e., $D_f(D_f) =$ "yes", a contradiction again.





## The time hierarchy theorem

**Theorem.** If $f(n) \geq n$ is a proper complexity function, then the class $\mathbf{TIME}(f(n))$ is strictly contained within $\mathbf{TIME}((f(2n+1))^3)$.

- $\mathbf{TIME}(f(n)) \subseteq \mathbf{TIME}((f(2n+1))^3)$ as $f$ is nondecreasing.
- By the first lemma: $H_{f(2n+1)} \in \mathbf{TIME}((f(2n+1))^3)$.
- By the second lemma:

  $H_{f(2n+1)} \notin \mathbf{TIME}(f(\lfloor \frac{2n+1}{2} \rfloor)) = \mathbf{TIME}(f(n))$.

**Corollary.** $\mathbf{P}$ is a *proper* subset of $\mathbf{EXP}$.

- Since $n^k = \mathrm{O}(2^n)$, we have $\mathbf{P} \subseteq \mathbf{TIME}(2^n) \subseteq \mathbf{EXP}$.
- It follows by the time hierarchy theorem that

  $\mathbf{TIME}(2^n) \subset \mathbf{TIME}((2^{2n+1})^3) \subseteq \mathbf{TIME}(2^{n^2}) \subseteq \mathbf{EXP}$.





## The space hierarchy theorem

**Theorem.** If $f(n) \geq n$ is a proper complexity function, then the class $\mathbf{SPACE}(f(n))$ is a *proper* subset of $\mathbf{SPACE}(f(n)\log f(n))$.

However, counter-intuitive results are obtained if non-proper complexity functions are allowed.

**Theorem. (The Gap Theorem).**

There is a recursive function $f$ from the nonnegative integers to the nonnegative integers such that $\mathbf{TIME}(f(n)) = \mathbf{TIME}(2^{f(n)})$.

Proof sketch.

The bound $f$ can be defined so that no TM $M$ computing on input $x$ with $|x| = n$ halts after number of steps between $f(n)$ and $2^{f(n)}$.





# 4. Reachability Method

**Theorem.** Let $f(n)$ be a proper complexity function. Then

(a) $\mathbf{SPACE}(f(n)) \subseteq \mathbf{NSPACE}(f(n))$ and
$\mathbf{TIME}(f(n)) \subseteq \mathbf{NTIME}(f(n))$.

(b) $\mathbf{NTIME}(f(n)) \subseteq \mathbf{SPACE}(f(n))$.

(c) $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{TIME}(c^{\log n + f(n)})$.

Proofs.

(a) A TM is a NTM, too.

(b) Simulation of all choices within space $f(n)$ (see below).

(c) Proof by reachability method (see below).

## Proof of $\mathbf{NTIME}(f(n)) \subseteq \mathbf{SPACE}(f(n))$

- Let $L \in \mathbf{NTIME}(f(n))$. Hence there is a precise nondeterministic Turing machine $N$ that decides $L$ in time $f(n)$.
- Let $d$ be the degree on nondeterminism (maximal number of possible moves for any state-symbol pair in $\Delta$).
- Any computation of $N$ is a $f(n)$-long sequence of nondeterministic choices (represented by integers $0, 1, \ldots, d-1$).
- The simulating deterministic machine $M$ considers all such sequences of choices and simulates $N$ on each.





Proof—cont'd.

- With sequence $(c_1, c_2, \ldots, c_{f(n)})$ $M$ simulates the actions that $N$ would have taken had $N$ taken choice $c_i$ at step $i$.
- If a sequence leads $N$ to halting with "yes", then $M$ does, too.

  Otherwise it considers the next sequence. If all sequences are exhausted without accepting, then $M$ rejects.
- There is an exponential number of simulations to be tried but they can be carried out in *space* $f(n)$ by carrying them out one-by-one, always erasing the previous simulation to reuse space.
- As $f(n)$ is proper, the first sequence $0^{f(n)}$ can be generated in space $f(n)$.





## Proof of $\mathbf{NSPACE}(f(n)) \subseteq \mathbf{TIME}(c^{\log n + f(n)})$

The *reachability method* is used to prove the claim.

- Consider a $k$-string *nondeterministic* TM $M$ with input and output which decides a language $L$ within space $f(n)$.
- We develop a deterministic method for simulating the nondeterministic computation of $M$ on input $x$ within time $c^{\log n + f(n)}$ where $n = |x|$ and $c$ is a constant depending on $M$.
- The *configuration graph* $G(M,x)$ of $M$ is used:

  nodes are all possible configurations of $M$ and there is an edge between two nodes (configurations) $C_1$ and $C_2$ iff $C_1 \xrightarrow{M} C_2$.
- Now $x \in L$ iff there is a path from $C_0 = (s, \triangleright, x, \triangleright, \varepsilon, \ldots, \triangleright, \varepsilon)$ to some configuration of the form $C = (\text{"yes"}, \ldots)$ in $G(M,x)$.





Proof—cont'd.

- A configuration $(q, w_1, u_1, \ldots, w_k, u_k)$ is a complete "snapshot" of a computation.
- Since $M$ is a machine with input and output *deciding* $L$:

  – the output string can be neglected,

  – for the input string, only the cursor position can change, and

  – for all other $k-2$ strings, the length is at most $f(n)$.
- A configuration can be represented as $(q, i, w_2, u_2, \ldots, w_{k-1}, u_{k-1})$ where $1 \leq i \leq n$ gives the cursor position on the input string.
- How many possible configurations does $M$ have? At most

  $$|K|(n+1)(|\Sigma|^{f(n)})^{2(k-2)} \leq |K|2n(|\Sigma|^{2(k-2)})^{f(n)} \leq nc_1^{f(n)} \leq c_1^{\log n + f(n)}$$

  for some constant $c_1$ depending on $M$.

Proof—cont'd.

- Hence, deciding whether $x \in L$ holds can be done by solving a reachability problem for a graph with at most $c_1^{\log n + f(n)}$ nodes.
- The problem can be solved, say, with a quadratic algorithm in time $c_2 c_1^{2(\log n + f(n))} \leq c^{\log n + f(n)}$ with $c = c_2 c_1^2$.
- The graph $G(M,x)$ needs not to be represented explicitly (e.g., as an adjacency matrix) for the reachability algorithm.
- The existence of an edge from $C$ to $C'$ can be determined on the fly by examining $C$, $C'$, and the input $x$.





## 5. Class Inclusions

**Corollary.** $\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$.

Proof.

1. $\mathbf{L} = \mathbf{SPACE}(\log n) \subseteq \mathbf{NSPACE}(\log n) = \mathbf{NL}$ follows by (a).
2. $\mathbf{NL} = \mathbf{NSPACE}(\log n) \subseteq \mathbf{TIME}(c^{\log n + \log n}) = \mathbf{TIME}(n^{2\log c}) \subseteq \mathbf{P}$ follows by (c).
3. By (a) $\mathbf{TIME}(n^k) \subseteq \mathbf{NTIME}(n^k)$ which implies $\mathbf{P} \subseteq \mathbf{NP}$.
4. By (b) $\mathbf{NTIME}(n^k) \subseteq \mathbf{SPACE}(n^k)$ which implies $\mathbf{NP} \subseteq \mathbf{PSPACE}$.
5. By (a) and (c) $\mathbf{SPACE}(n^k) \subseteq \mathbf{NSPACE}(n^k) \subseteq \mathbf{TIME}(c^{\log n + n^k}) \subseteq \mathbf{TIME}(2^{n^{k+c'}}) \subseteq \mathbf{EXP}$.





## Which inclusions are proper?

**Corollary.** The class $\mathbf{L}$ is a proper subset of $\mathbf{PSPACE}$.

Proof. The space hierarchy theorem tells us $\mathbf{L} = \mathbf{SPACE}(\log(n)) \subset \mathbf{SPACE}(\log(n)\log(\log(n))) \subseteq \mathbf{SPACE}(n^2) \subseteq \mathbf{PSPACE}$. □

It is believed that *all* inclusions of the complexity classes in $\mathbf{L} \subseteq \mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$ are proper.

However, we only know that

- at least one of the inclusions between $\mathbf{L}$ and $\mathbf{PSPACE}$ is proper (but don't know which) and
- at least one of the inclusions between $\mathbf{P}$ and $\mathbf{EXP}$ is proper (but don't know which).





## 6. Simulating Nondeterministic Space

- The question is how efficiently can we simulate nondeterministic space by deterministic space?
- It follows by the previous theorem that

$$\mathbf{NSPACE}(f(n)) \subseteq \mathbf{TIME}(c^{\log n + f(n)}) \subseteq \mathbf{SPACE}(c^{\log n + f(n)}).$$

  But can we do better than this?
- *Yes*, in fact. Nondeterministic space can be simulated with quadratic deterministic space (using a theorem that follows).

## Savitch's theorem

**Theorem.** REACHABILITY $\in \mathbf{SPACE}(\log^2 n)$.

Proof sketch.

- Given a graph $G$ and nodes $x,y$ and $i \geq 0$, define $PATH(x,y,i)$: there is a path from $x$ to $y$ of length at most $2^i$.
- If $G$ has $n$ nodes, any simple path is at most $n$ long and we can solve reachability in $G$ if we can compute whether $PATH(x,y,\lceil \log n \rceil)$ holds for any given nodes $x,y$ of $G$.
- This can be done using *middle-first search*.





Proof—cont'd.

- **function** $path(x,y,i)$ /* middle-first search */
  **if** $i = 0$ **then**
  **if** $x = y$ or there is an edge $(x,y)$ in $G$ **then** return "yes"
  **else for** all nodes $z$ **do**
  **if** $path(x,z,i-1)$ and $path(z,y,i-1)$ **then** return "yes";
  return "no"
- Proof that $path(x,y,i)$ correctly determines $PATH(x,y,i)$:
  If $i = 0$, then clearly $path$ correctly determines $PATH(x,y,0)$.
  For $i > 0$, $path(x,y,i)$ returns "yes" iff there is a node $z$ with $path(x,z,i-1)$ and $path(z,y,i-1)$ holding. By the inductive hypothesis there are paths from $x$ to $z$ and from $z$ to $y$ both at most $2^{i-1}$ long. Then there is a path from $x$ to $y$ at most $2^i$ long.





Proof—cont'd.

- The algorithm is started with $path(x,y,\lceil \log n \rceil)$.
- $\mathrm{O}(\log^2 n)$ space bound can be achieved by handling recursion using a stack containing a triple $(x,y,i)$ for each active recursive call.
  For each node $z$ put $(x,z,i-1)$ into the stack and call $path(x,z,i-1)$. If this fails, erase $(x,z,i-1)$ and put $(x,z',i-1)$ for the next $z'$ otherwise erase $(x,z,i-1)$ and put $(z,y,i-1)$.
- As there are at most $\log n$ recursive calls active with each taking at most $3\log n$ space, the $\mathrm{O}(\log^2 n)$ space bound is achieved.





**Corollary.** For any proper complexity function $f(n) \geq \log n$,

$$\mathbf{NSPACE}(f(n)) \subseteq \mathbf{SPACE}((f(n))^2).$$

Proof.

- To simulate an $f(n)$-space bounded NTM $M$ on input $x$, run the previous algorithm on the configuration graph $G(M,x)$.
- The edges of the graph $G(M,x)$ are determined on the fly by examining the input $x$.
- The configuration graph has at most $c_1^{\log n + f(n)} \leq c^{f(n)}$ nodes.
- By Savitch's theorem, the algorithm needs at most $(\log c^{f(n)})^2 = f(n)^2 \log^2 c = \mathrm{O}(f(n)^2)$ space.

**Corollary.** $\mathbf{PSPACE} = \mathbf{NPSPACE}$.

☞ Nondeterminism is less powerful with respect to space than time.

# 7. Closure under Complement

- A key result about reachability will be established:

  the number of nodes reachable from a node $x$ can be computed in nondeterministic $\log n$ space!

- The complement (the number of nodes not reachable from $x$) can be handled in nondeterministic $\log n$ space, too!

  (This quantity can be obtained by a simple subtraction.)

- It is open (and doubtful) whether nondeterministic *time* complexity classes are closed under complement.





## Functions computed by NTMs

When does a NTM $M$ compute a function $F$ from strings to strings?

- On input $x$, each computation of $M$ either

  – outputs the correct answer $F(x)$ or

  – enters the rejecting "no" state.

- At least one computation must end up with $F(x)$ which must be unique for all such computations.

- Such a machine observes a space bound $f(n)$ iff for any input $x$, at halting all strings (except the ones reserved for input and output) are of length at most $f(|x|)$.





## Immerman-Szelepscényi theorem

**Theorem.** Given a graph $G$ and a node $x$, the number of nodes reachable from $x$ in $G$ can be computed by a NTM within space $\log n$.

Proof.

- Let us define $S(k)$ as the set of nodes in $G$ which are reachable from $x$ via paths of length $k$ or less.

- The strategy is to compute values $|S(1)|, |S(2)|, \ldots, |S(n-1)|$ iteratively and recursively, i.e. $|S(i)|$ is computed from $|S(i-1)|$.

- Given that the number of nodes in $G$ is $n$, the number of nodes reachable from $x$ in $G$ is $|S(n-1)|$.

- Let $G(v,u)$ mean that $v = u$ or there is an arc from $v$ to $u$ in $G$.





Proof—cont'd.

The nondeterministic algorithm:

$|S(0)| := 1;$
**for** $k := 1, 2, \ldots, n-1$ **do**
  $l := 0;$
  **for** each node $u := 1, 2, \ldots, n$ **do**
    check whether $u \in S(k)$ and set *reply* accordingly;
    /* See below how this is implemented */
    **if** $reply = true$ **then** $l := l + 1;$
  **end for**;
  $|S(k)| := l$
**end for**

Proof–cont'd.

/* Check whether $u \in S(k)$ and set *reply* */
$m := 0$; $reply := false$;
**for** each node $v := 1, 2, \ldots, n$ **do**
  /* check whether $v \in S(k-1)$ */
  $w_0 := x$; $path := true$
  **for** $p := 1, 2, \ldots, k-1$ **do**
    guess a node $w_p$; **if** not $G(w_{p-1}, w_p)$ **then** $path := false$
  **end for**
  **if** $path = true$ and $w_{k-1} = v$ **then**
    $m := m+1$; /* $v \in S(k-1)$ holds */
    **if** $G(v,u)$ **then** $reply := true$
  **end if**
**end for**
**if** $m < |S(k-1)|$ **then** "give up" (end in "no" state)





Proof—cont'd.

- Note that only $\log n$-space is needed as there are only nine variables: $|S(k)|, k, l, u, m, v, p, w_p, w_{p-1}$
  which each (an integer) can be stored in $\log n$ space.
- The algorithm computes correctly $|S(k)|$ (by induction on $k$):
  – If $k = 0$, then $|S(k)| = 1$ as given by the algorithm.
  – For $k > 0$, consider a computation that does not "give up". We need to show that counter $l$ is incremented iff $u \in S(k)$.
  If counter $l$ is incremented, then $reply = true$ implying that $u \in S(k)$, i.e. there is a path $(x=)w_0, \ldots, w_{k-1}(=v), u$.
  If $u \in S(k)$, then there is some $v \in S(k-1)$ such that $G(v,u)$. But as the computation does not "give up", $m = |S(k-1)|$ (which is the correct value by induction) and therefore all $v \in S(k-1)$ are verified as such and, thus, *reply* is set to *true*.
  – Moreover, clearly there is at least one accepting computation where paths to the members of $S(k-1)$ are correctly guessed.





## Closure under Complement

**Corollary.** If $f(n) \geq \log n$ is a proper complexity function, then $\mathbf{NSPACE}(f(n)) = \mathbf{coNSPACE}(f(n))$.

Proof sketch.

- Suppose $L \in \mathbf{NSPACE}(f(n))$ is decided by an $f(n)$-space bounded NTM $M$. We build an $f(n)$-space bounded NTM $\overline{M}$ deciding $\overline{L}$.
- On input $x$, $\overline{M}$ runs the previous algorithm on the configuration graph $G(M,x)$ associated with $M$ and $x$.
- $\overline{M}$ rejects if it finds an accepting configuration in any $S(k)$.
- Since $G(M,x)$ has at most $n_g = c^{f(n)}$ nodes, then $\overline{M}$ can accept if $|S(n_g - 1)|$ is computed without an accepting configuration.
- Due to bound $n_g$, $\overline{M}$ needs at most $\log c^{f(n)} = O(f(n))$ space.





## Learning Objectives

- The definitions and background of major complexity classes: **P**, **NP**, **PSPACE**, **NPSPACE**, **EXP**, **L**, and **NL**.
- The knowledge of basic relationships between complexity classes (inclusions and proper inclusions).
- Savitch's theorem and Immerman-Szelepscényi theorem.