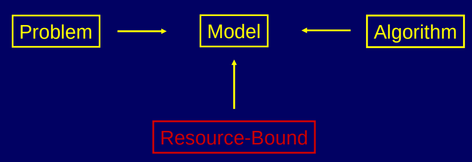
Resource Bounds



Resource Bounds

consist of

a bounded resource

e.g. time or space of a Turing Machine

the bound itself in terms of a function which bounds the resource depending on the problem size

e.g. f(n)=n

Resource Bounds

Fundamental Resources

(formulated as classes)

DTIME(f)	a DTM decides L within f(n) steps
DSPACE(f)	a DTM decides L using f(n) cells
NTIME(f)	a NTM decides L within f(n) steps
NSPACE(f)	a NTM decides L using f(n) cells

Resource Bounds

Constants do not matter

$TIME(f) = TIME(\varepsilon f + n), \varepsilon > 0$

$SPACE(f) = SPACE(\varepsilon f), \varepsilon > 0$

Deterministic or Nondeterministic, it does not matter

Constants do not matter

Linear Speedup (Proof I)

$TIME(f) = TIME(\varepsilon f + n), \varepsilon > 0$

Let $M =< K, \Sigma, \delta, s >$ be a TM which uses t tapes

Then let $\overline{M} =< \overline{K}, \overline{\Sigma}, \overline{\delta}, \overline{s} >$ be a TM which uses $t+1$ tapes and choose k > 6, set $\overline{\Sigma} = \Sigma^k$

$\overline{M}$ copies the input to its additional tape and compresses the input

Constants do not matter

Linear Speedup (Proof II)

$\overline{M}$ then simulates M by using the additional tape as input tape

$\overline{M}$ moves to the right, two times left and once right

$\overline{M}$ knows all symbols M would have read within k steps

$\overline{M}$ simulates the next k steps of M on the compressed representation (2 steps)

$\overline{M}$ requires 6 steps to simulate k steps of M ●

Constants do not matter

Linear Compression

$$SPACE(f) = SPACE(\varepsilon f), \varepsilon > 0$$

Same simulation as for linear speedup

$\overline{M}$ requires $(1/k) f + 2$ cells to simulate M

Resource Bounds

Proper Complexity Function

The functions used as bounds have to satisfy some conditions to avoid anomalies.

These functions are called "Proper Complexity Functions"

Proper Complexity Function

Definition

Let f be a function $N \to N$ with

$f(n+1) \geq f(n)$

there is a DTM M which ouputs $1^{f(n)}$ on input x ($|x| = n$) and runs within $DTIME(n + f(n))$ and $DSPACE(f(n))$

then f is a proper complexity function

Proper Complexity Function

Examples

$f(n) = c$ $\quad$ $f(n) = n!$

$f(n) = \log(n)$ $\quad$ $f(n) = \sqrt{n}$

$f(n) = n$

$\left.\begin{array}{l} f(n) + g(n) \\ f(n)g(n) \\ f(n)^{g(n)} \end{array}\right\} f(n)$ and $g(n)$ proper

Important proper complexity functions

Proper Complexity Functions

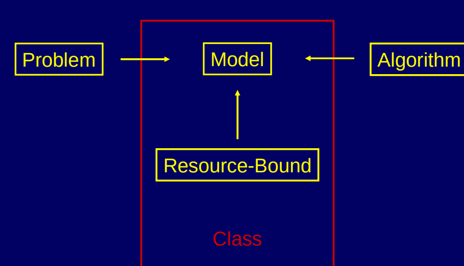
The Gap Theorem

One of the above mentioned anomalies:

Let g be a recursive function $N \to N$ with $g(n+1) > g(n)$. Then there is a recursive function $f : N \to N$ with $DTIME(f(n)) = DTIME(g(f(n)))$.

Original prove in terms of Blum-Complexity, thus the same holds for DSPACE.

Fundamental Complexity Classes



Fundamental Complexity Classes

Definitions

$L = DSPACE(\log n)$

$NL = NSPACE(\log n)$

$P = \bigcup_{c=1}^{\infty} DTIME(n^c)$

$NP = \bigcup_{c=1}^{\infty} NTIME(n^c)$

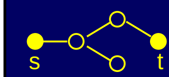
$PSPACE = \bigcup_{c=1}^{\infty} DSPACE(n^c)$

$NPSPACE = \bigcup_{c=1}^{\infty} NSPACE(n^c)$

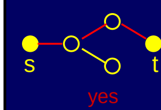
$EXP = \bigcup_{c=1}^{\infty} DTIME(2^{n^c})$

$NEXP = \bigcup_{c=1}^{\infty} NTIME(2^{n^c})$

Example: Reachability



Is there a path from s to t?



Reachability $= \langle I, f \rangle$

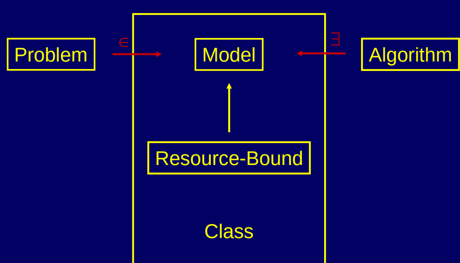
$I = \{ \langle G, s, t \rangle \mid s, t \in V(G) \}$

$f(\langle G, s, t \rangle) = \begin{cases} 1 : \exists path(G, s, t) \\ 0 : otherwise \end{cases}$

In which class is Reachability?

Example: Reachability

In which class is Reachability?



Example: Reachability

In which class is Reachability?

What is the complexity of Dijkstra?

$REACHABILITY \in P$

What about NTMs?

$REACHABILITY \in NL$

Example: Reachability

Reachability in NL (Proof)

$I = \langle G, s, t \rangle$ with $G = \langle V, E \rangle$ given.

1. $steps := 0$; $current := s$;
2. if($current = t$) return true;
3. if($steps > |V|$) return false;
4. $steps := steps + 1$;
5. $current$ chose from $\{ v \in V \mid \langle current, v \rangle \in E \}$
6. goto 2

$steps, current, |V|$, are integers $\leq |V|$

Thus $REACHABILITY \in NSPACE(3\log(\sqrt{n})) = NSPACE(\log(n))$

Relating Complexity Classes

We defined $L, NL, P, NP, PSPACE, NPSPACE, EXP$, and $NEXP$.

Which subset-relations hold between these Complexity Classes?

Relating Complexity Classes

Relationships by Definition

$L \subseteq NL$

$P \subseteq NP$

$PSPACE \subseteq NSPACE$

$EXP \subseteq NEXP$

Determinism
vs.
Nondeterminism

$L \subseteq PSPACE$

$NL \subseteq NPSPACE$

$P \subseteq EXP$

$NP \subseteq NEXP$

Exponentially
Higher Bound

Relating Complexity Classes

Hierarchy Theorems

$DTIME(f(n)) \subset DTIME(f(2n+1)^2)$
$f(n) \geq n$

$DSPACE(f(n)) \subset DSPACE(f(2n+1)\log f(n))$

f proper

The same holds for nondeterministic computation

(the bound can be lowered significantly)

Hierarchy Theorems

Time Hierarchy: Proof (I)

Let $B_f^{DTIME} = \{<M,x>| M(x)=1 \text{ within } DTIME(f(|x|))\}$

$B_f^{DTIME} \in DTIME(s[f](n))$ (Bounded Simulation)

Set $D_f^{DTIME} = \{M | <M,M> \notin B_f^{DTIME}\}$

Let N be an arbitrary Machine in $DTIME(f(n))$

$N(N)=1 \Leftrightarrow <N,N> \in B_f^{DTIME}$

$N(N)=1 \Leftrightarrow N \notin D_f^{DTIME}$
$N(N)=1 \Leftrightarrow N \in L(N)$
$\Big\} L(N) \neq D_f^{DTIME}$

$D_f^{DTIME} \notin DTIME(f(n))$ $D_f^{DTIME} \in DTIME(s[f](2n+1))$

Hierarchy Theorems

Time Hierarchy: Proof (II)

Let $B_f^{DTIME} = \{<M,x>| M(x)=1 \text{ within } DTIME(f(|x|))\}$

$B_f^{DTIME} \in DTIME(s[f](n))$ (Bounded Simulation)

$D_f^{DTIME} \notin DTIME(f(n))$ $D_f^{DTIME} \in DTIME(s[f](2n+1))$

$DTIME(f(n)) \subset DTIME(s[f](2n+1))$

There are several bounded simulation results. It is important to us that $s[f](n)$ is bounded by a polynomial in f. E.g., $s[f](n) = f^3(n)$, for $f(n) \geq n$

$DTIME(f(n)) \subset DTIME(f^3(2n+1))$

Hierarchy Theorems

Reusing the Proof

$D_f^{RES} \notin RES(f(n))$ $D_f^{RES} \in RES(s[f](2n+1))$

The last proof was generic – every bounded simulation can be substituted.

$B_f^{DSPACE} \in DSPACE(f(n)\log f(n))$

$DSPACE(f(n)) \subset DSPACE(f(2n+1)\log f(n))$

Hierarchy Theorems

Exponentially Higher Bounds

We do the DTIME-case:

$DTIME(f(n)) \subset DTIME(f(2n+1)^2)$
$f(n) \geq n$

f proper

$DTIME(p(n)) \subseteq DTIME(2^n) \subset DTIME((2^{2n+1})^3) \subseteq DTIME(2^{n^2})$

$P \subset EXP$

Relating Complexity Classes

Relationships

$L \subseteq NL$

$P \subseteq NP$

$PSPACE \subseteq NSPACE$

$EXP \subseteq NEXP$

Determinism
vs.
Nondeterminism

$L \subset PSPACE$

$NL \subset NPSPACE$

$P \subset EXP$

$NP \subset NEXP$

Exponentially
Higher Bound

Relating Complexity Classes

Further Relationships

$NTIME(f(n)) \subseteq DSPACE(f(n))$

$NSPACE(f(n)) \subseteq DTIME(c^{\log n + f(n)})$

$NSPACE(f(n)) \subseteq DSPACE(f^2(n))$
$f(n) \geq \log n$

f proper

Relating Complexity Classes

NTIME vs. DSPACE (Proof I)

$NTIME(f(n)) \subseteq DSPACE(f(n))$

Let M be an NTM running in time $f(n)$.
In each step, M can make a single nondeterministic decision.
However, M can only chose out of c_M continuations in a step.
Thus, $\overline{M}$ enumerates all possible choices, taking space $c_M f(n)$.
This string is then used by $\overline{M}$ as a lookup-table
whenever M is taking a nondet. choice.

Relating Complexity Classes

NTIME vs. DSPACE (Proof II)

Thus, $\overline{M}$ enumerates all possible choices, taking space $c_M f(n)$.
This string is then used by $\overline{M}$ as a lookup-table
whenever M is taking a nondet. choice.

For each enumerated choice-string, $\overline{M}$ simulates M.
If M aceepts in one of these simulations, $\overline{M}$ accepts, too.
Otherwise, $\overline{M}$ rejects.

$\overline{M}$ requires $c_M f(n) + f(n)$ space, i.e. $\overline{M} \in DSPACE(f(n))$. ●

Relating Complexity Classes

NTIME vs. DSPACE

$NTIME(f(n)) \subseteq DSPACE(f(n))$

$NP \subseteq PSPACE$

Relating Complexity Classes

NSPACE vs. DTIME (Proof I)

$NSPACE(f(n)) \subseteq DTIME(c^{\log n + f(n)})$

Let M be an NTM running in space $f(n)$.
A configuration of M has the following parts:
the state $k \in K_M$ of M
the cursor position $1 \leq i \leq n+1$ of M
the contents $< s_1, \ldots, s_l >$ of the tapes of M : $s_i \in \Sigma^{f(n)}$

Thus, there are $|K_M|(n+1)|\Sigma|^{lf(n)}$ different configs.
Using C_M we find at most $C_M^{\log n + f(n)}$ configs.

Relating Complexity Classes

NSPACE vs. DTIME (Proof II)

Using C_M we find at most $C_M^{\log n + f(n)}$ configs.

Now we define $G_x^M = < V, E >$ with $V = \{\text{configs. of } M\}$ and $< u, v > \in E$ iff there is a direct transition from u to v on input x.

Define $s \in V$ to be the initial config of M and
$t \in V$ to be the accepting config of M (normalization).

$< G_x^M, s, t >$ is a $REACH$ instance with $C_M^{\log n + f(n)}$ nodes.
$< G_x^M, s, t > \in REACH$ iff $M(x) = 1$

Relating Complexity Classes

NSPACE vs. DTIME (Proof III)

$< G_x^M, s, t >$ is a $REACH$ instance with $C_M^{\log n + f(n)}$ nodes.
$< G_x^M, s, t > \in REACH$ iff $M(x) = 1$

$REACH \in P$. Thus we can decide $< G_x^M, s, t > \in REACH$ in $DTIME((C_M^{\log n + f(n)})^k)$ for some constant k.

$DTIME((C_M^{\log n + f(n)})^k) = DTIME(c^{\log n + f(n)})$ •

Relating Complexity Classes

NSPACE vs. DTIME
A Note on the Proof

$< G_x^M, s, t >$ is a $REACH$ instance with $C_M^{\log n + f(n)}$ nodes.
$< G_x^M, s, t > \in REACH$ iff $M(x) = 1$

The method of representing a space-bounded computation by a graph G_x^M is called the "Reachability-Method".

Effectively, this is a generic reduction!
$REACH$ is $NL-hard$.

Relating Complexity Classes

NSPACE vs. DTIME

$$NSPACE(f(n)) \subseteq DTIME(c^{\log n + f(n)})$$

$NL \subseteq P$
$NPSPACE \subseteq EXP$

Relating Complexity Classes

NSPACE vs. DSPACE (Proof I)

$NSPACE(f(n)) \subseteq DSPACE(f^2(n))$
$f(n) \geq \log n$

$< G_x^M, s, t >$ is a $REACH$ instance with $C_M^{\log n + f(n)}$ nodes.
$< G_x^M, s, t > \in REACH$ iff $M(x) = 1$

since $f(n) \geq \log n$

$< G_x^M, s, t >$ is a $REACH$ instance with $C^{f(n)}$ nodes.
$< G_x^M, s, t > \in REACH$ iff $M(x) = 1$

Relating Complexity Classes

NSPACE vs. DSPACE (Proof II)

$< G_x^M, s, t >$ is a $REACH$ instance with $C^{f(n)}$ nodes.
$< G_x^M, s, t > \in REACH$ iff $M(x) = 1$

We cannot compute the graph – it is exponential!
So how to access it?

We can compute the configuartions s and t.

Having two nodes u and v, we check $< u, v > \in E$ by simulating M on u with input string x.

Relating Complexity Classes

NSPACE vs. DSPACE (Proof III)

$PATH(G,i,j,d)$
  if $<i,j> \in E$ then return true;
  if $d = 0$ then return false;
  for($z = 1; z \leq |V|; ++z$)
    if $PATH(G,i,z,d-1)$ and $PATH(G,z,j,d-1)$ then
      return true;
  return false;

$PATH(G,i,j,d)$ is true iff $\exists$ a path from i to j of length $\leq 2^d$

$PATH(G,s,t,\lceil \log|V| \rceil)$ iff $<G,s,t> \in REACH$

Relating Complexity Classes

NSPACE vs. DSPACE (Proof IV)

$PATH(G,i,j,d)$
  if $<i,j> \in E$ then return true;
  if $d = 0$ then return false;
  for($z = 1; z \leq |V|; ++z$)
    if $PATH(G,i,z,d-1)$ and $PATH(G,z,j,d-1)$ then
      return true;
  return false;

Recursive depth of at most d

Each "stack-frame" has size $3\log|V|$

$PATH(G,s,t,\lceil \log|V| \rceil)$ requires $3\log^2|V|$ space

Relating Complexity Classes

NSPACE vs. DSPACE (Proof V)

$<G_x^M, s, t>$ is a $REACH$ instance with $C_M^{f(n)}$ nodes.

$<G_x^M, s, t> \in REACH$ iff $M(x) = 1$

$PATH(G,s,t,\lceil \log|V| \rceil)$ iff $<G,s,t> \in REACH$

$PATH(G,s,t,\lceil \log|V| \rceil)$ requires $3\log^2|V|$ space

Taken together: $M(x) = 1$ can be decided in

$DSPACE(3\log^2(C_M^{f(n)})) = DSPACE(f^2(n))$ ●

Relating Complexity Classes

NSPACE vs. DSPACE

$NSPACE(f(n)) \subseteq DSPACE(f^2(n))$

$f(n) \geq \log n$

$NPSPACE = PSPACE$

Relating Complexity Classes

Relationships

$L \subseteq NL$

$P \subseteq NP$

$PSPACE \subseteq NSPACE$

$EXP \subseteq NEXP$

$NL \subseteq P$

$NP \subseteq PSPACE$

$NPSPACE \subseteq PSPACE$

$NPSPACE \subseteq EXP$

Determinism
vs.
Nondeterminism

Relating Complexity Classes

Relationships

$L \subseteq NL$

$P \subseteq NP$

$PSPACE \subseteq NSPACE$

$EXP \subseteq NEXP$

$NL \subseteq P$

$NP \subseteq PSPACE$

$NPSPACE \subseteq PSPACE$

$NPSPACE \subseteq EXP$

$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP \subseteq NEXP$

Relating Complexity Classes

Further Relationships

$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP \subseteq NEXP$

$NL \subset PSPACE$ $\quad P \subset EXP$ $\quad NP \subset NEXP$

Thus there must be proper set inclusions – however, the question which ones are proper is an open question.

Complement Problems

Let L be a language.
Then $\overline{L} = \{x \in \Sigma^* \mid x \notin L\}$ is the associated complement language.

Thus, formally L and $\overline{L}$ add up to Σ^*.
However, often one defines $\overline{CircuitSAT}$ as the set of circuits which are not satisfiable.

In consequence $CircuitSAT \cup \overline{CircuitSAT}$ is the set of strings which encode circuits.

Complement Classes

Let C be a class of decision problems.
Then $coC = \{\overline{L} \mid L \in C\}$.

Deterministic classes are closed under complementation:
$L = coL$, $P = coP$, $PSPACE = coPSPACE$, $EXP = coEXP$.

Complement Classes

Nondeterministic Co-Classes

How can we handle complement problems in the context of nondeterminism?

A problem is, say, in NP iff there is an NTM running in poly-time, which accepts every positive instance at the end of AT LEAST ONE path.

Consequently a problem is in $coNP$ iff there is an NTM running in poly-time, which accepts every positive instance at the end of EACH path.

Complement Classes

Example: CIRSAT

$CIRSAT$ can be solved with an NP-algorithm M:
- M guesses an assignemt A for the input circuit C
- M accepts iff A satisfies C.

Thus M evaluates $\exists A : C(A) = 1$.

$CIRSAT(COMPLEMENT)$ can be solved with a $coNP$-algorithm M:
- M guesses an assignemt A for the input circuit C
- M accepts iff A does not satisfy C.

Thus M evaluates $\forall A : C(A) = 0$

Complement Classes

Nondeterministic Co-Classes

The $NTIME$-case is open, i.e., whether $NP = coNP$, or $NEXP = coNEXP$ is unknown.

We already know: $NPSPACE = coNPSAPCE$, since $PSPACE = NPSPACE$. Is there more?

$NSPACE(f(n)) = coNSPACE(f(n))$
$f(n) \geq \log n$, proper

Immerman-Szelepscenyi Theorem

NSPACE vs. coNSPACE

Reachability Method Again

Again, we will use the reachability method:

That is, given an NTM M respecting the space bound f and an input string x, we define the configuration graph G_x^M.

$< G_x^M, s, t >$ is a $REACH$ instance with $C_M^{\log f(n)+f(n)}$ nodes.

$< G_x^M, s, t > \in REACH$ iff $M(x) = 1$

NSPACE vs. coNSPACE

Reusing: REACH is in NL

bool guesspath($G;v,k$)
1. $steps := 1$; $current := s$;
2. if($current = v$) return true;
3. if($steps > k$) return false;
4. $steps := steps + 1$;
5. $current$ chose from $\{u \in V \mid < current, u > \in E\}$
6. goto 2

guesspath$(G; v, k) = \begin{cases} \text{true} : \exists path(s,v) \text{ in } G \text{ of length} \leq k \\ \text{false} : \text{no such path exists, or wrong choices} \end{cases}$

guesspath$(G; v, k)$ takes $O(\log |V|)$ space

NSPACE vs. coNSPACE

Counting the Number of Reachable Nodes

Let $S(k) \subseteq V$ be the set of nodes which can be reached from s by a path of length $\leq k$.
$S(0) = \{s\}$.

Within $\log |V|$, we cannot compute $S(k)$ but we can compute $|S(k)|$.

This is still a bit complicated:
We will compute $|S(k+1)|$ based on $|S(k)|$.

NSPACE vs. coNSPACE

Functions & Nondeterminism

We say that we can compute a function with a non-deterministic machine, iff all accepting paths lead to the same result.

- we must prove that each accepting path leads to the correct result

- we have to prove that there is at least one accepting path

NSPACE vs. coNSPACE

CheckPath

bool checkpath(G;v,k,last)
1. $count := 0$; $result :=$ false;
2. for $u := 1$ to $|V|$ do
3. if guesspath(G;u,k-1) then
4. $count := count + 1$;
5. if $u = v$ or $< u, v > \in E$ then $result :=$ true;
6. if $count < last$ then reject; else return $result$;

checkpath$(G;v,k,|S(k\text{-}1)|) \Leftrightarrow v \in S(k) \quad k > 0$

checkpath$(G; v, k, |S(k-1)|)$ takes $O(\log |V|)$ space
(guesspath, $count$, and u require only $O(\log |V|)$)

NSPACE vs. coNSPACE

CheckPath (Correctness I)

bool checkpath(G;v,k,last)
1. $count := 0$; $result :=$ false;
2. for $u := 1$ to $|V|$ do
3. if guesspath(G;u,k-1) then
4. $count := count + 1$;
5. if $u = v$ or $< u, v > \in E$ then $result :=$ true;
6. if $count < last$ then reject; else return $result$;

$count := count + 1 \Rightarrow u$ is reachable from s by path of length $< k$
$count = last = |S(k-1)| \Rightarrow$ all nodes in $S(k-1)$ have been found, otherwise line 6 rejects

NSPACE vs. coNSPACE
CheckPath (Correctness II)

bool checkpath(G;v,k,last)
1. $count := 0$; $result :=$ false;
2. for $u := 1$ to $|V|$ do
3. if guesspath(G;u,k-1) then
4. $count := count + 1$;
5. if $u = v$ or $<u,v> \in E$ then $result :=$ true;
6. if $count < last$ then reject; else return $result$;

$count = last = |S(k-1)| \Rightarrow$ all nodes in $S(k-1)$ have been found, otherwise line 6 rejects.

but then line 5 correctly determines whether $v \in S(k)$

NSPACE vs. coNSPACE
Unreachable

bool unreachable(G)
1. $last := 1$;
2. for $k := 1$ to $|V|-2$ do
3. $current := 0$;
4. for $v := 1$ to $|V|$ do
5. if checkpath($G;v,k,last$) then $current := current + 1$;
6. $last := current$;
7. return not checkpath($G;t,|V|-1,last$);

unreachable(G) $\Leftrightarrow \neg\exists path(G,s,t)$

unreachable(G) takes $O(\log|V|)$ space
(checkpath, $last$, k, v take $O(\log|V|)$ space)

NSPACE vs. coNSPACE
Unreachable (Correctness)

bool unreachable(G)
1. $last := 1$;
2. for $k := 1$ to $|V|-2$ do
3. $current := 0$;
4. for $v := 1$ to $|V|$ do
5. if checkpath($G;v,k,last$) then $current := current + 1$;
6. $last := current$;
7. return not checkpath($G;t,|V|-1,last$);

for lines 2-5, $last = |S(k-1)|$ can be proved by induction, starting with $|S(0)| = 1$, and using $current = |S(k)|$ in line 6

therefore line 7 returns the correctly $\neg\exists$path(G,s,t)

Relating Complexity Classes
Co-Classes

$NL = coNL$ $\quad P \subseteq coNP \subseteq PSPACE$

It is a central open question whether
$NP = coNP$ or
$NEXP = coNEXP$ holds.

Also unknown: Does $NP \cap coNP = P$ hold?
If yes, RSA is breakable.

Relating Complexity Classes
Summary

$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP \subseteq NEXP$

$NL \subset PSPACE$ $\quad P \subset EXP$ $\quad NP \subset NEXP$

$NL = coNL$ $\quad P \subseteq coNP \subseteq PSPACE$

Relating Complexity Classes
Techniques

Diagonalization
$DTIME(f) \subset DTIME(f^3)$
$DSPACE(f) \subset DSPACE(f \log f)$

Reachability Method
$NSPACE(f) \subseteq DTIME(c^{\log f + f})$
$NSPACE(f) \subseteq DSPACE(f^2), \quad f \geq \log n$
$NSPACE(f) = coNSPACE(f), \quad f \geq \log n$

Counting
$NSPACE(f) = coNSPACE(f), \quad f \geq \log n$

f proper