Resource Bounds

- Problem
- Model
- Algorithm
- Resource-Bound

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 (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

Constants do not matter

**Linear Speedup (Proof I)**

- $\text{TIME}(f) = \text{TIME}(gf + n), \epsilon > 0$

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

Then let $\overline{M} = \langle \overline{K}, \overline{\Sigma}, \overline{\delta}, \overline{s} \rangle$ 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

**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

$\text{SPACE}(f) = \text{SPACE}(gf), \varepsilon > 0$

Same simulation as for linear speedup

$M$ requires $(1/k)f + 2$ cells to simulate $M$

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 $N \rightarrow N$ with

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

there is a DTM $M$ which outputs $t^{(n)}$ on input $x \in \{1^n\}$ and runs within $\text{DTIME}(n + f(n))$ and $\text{DSPACE}(f(n))$

then $f$ is a proper complexity function

Proper Complexity Function

Examples

- $f(n) = c$
- $f(n) = \log(n)$
- $f(n) = n$
- $f(n) = g(n)$
- $f(n)g(n)$
- $f(n)^{g(n)}$

Important proper complexity functions

Proper Complexity Functions

The Gap Theorem

One of the above mentioned anomalies:

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

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

Fundamental Complexity Classes

Problem

Model

Resource-Bound

Algorithm

Class
Fundamental Complexity Classes

Definitions

- $L = \mathcal{DSPACE}(\log n)$
- $NL = \mathcal{NSPACE}(\log n)$
- $P = \bigcup_{c > 0} \mathcal{DTIME}(n^c)$
- $NP = \bigcup_{c > 0} \mathcal{NTIME}(n^c)$
- $PSPACE = \bigcup_{c > 0} \mathcal{DSPACE}(n^c)$
- $NPSPACE = \bigcup_{c > 0} \mathcal{NSPACE}(n^c)$
- $EXP = \bigcup_{c > 0} \mathcal{DTIME}(2^{n^c})$
- $NEXP = \bigcup_{c > 0} \mathcal{NTIME}(2^{n^c})$

Example: Reachability

In which class is Reachability?

What is the complexity of Dijkstra?

$\text{REACHABILITY} \in P$

What about NTMs?

$\text{REACHABILITY} \in NL$

Example: Reachability

Reachability in NL (Proof)

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

1. \text{steps} = 0, \text{current} = s,
2. \text{if} (\text{current} = t) \text{return true;}
3. \text{if} (\text{steps} \geq |V|) \text{return false;}
4. \text{steps} = \text{steps} + 1;
5. \text{current} \text{ chose from } \{ v \in V | v < \text{current}, v \not \in E \}
6. \text{goto 2}

\text{steps, current,} |V|, \text{are integers } \subseteq \mathbb{N}

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

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

<table>
<thead>
<tr>
<th>Class</th>
<th>Relationship</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L \subseteq NL )</td>
<td>( L \subseteq PSPACE )</td>
<td>( P \subseteq NP )</td>
</tr>
<tr>
<td>( PSPACE \subseteq NSPACE )</td>
<td>( P \subseteq EXP )</td>
<td>( EXP \subseteq NEXP )</td>
</tr>
<tr>
<td>Determinism</td>
<td>Exponentially</td>
<td>Nondeterminism</td>
</tr>
<tr>
<td>vs.</td>
<td>Higher Bound</td>
<td></td>
</tr>
</tbody>
</table>

Hierarchy Theorems

Time Hierarchy: Proof (I)

Let \( B^{DTIME}_1 = \{ M, x \rightarrow M(x) = 1 \text{ within } DTIME(f(o)) \} \)
\( B^{DTIME}_n \in DTIME(s(f(n))) \) (Bounded Simulation)
Set \( D^{DTIME}_n = \{ M \in M, M > e^{D^{DTIME}_n} \} \)
Let \( N \) be an arbitrary Machine in \( DTIME(f(n)) \)
\( N(N) = 1 \Rightarrow N \notin D^{DTIME}_n \)
\( N(N) = 1 \Rightarrow N \notin L(N) \)
\( D^{DTIME}_n \not\subseteq DTIME(f(n)) \)
\( D^{DTIME}_n \in DTIME(s(f(2n + 1))) \)

Hierarchy Theorems

Reusing the Proof

\( D^{RE} \not\subseteq RES(f(n)) \)
\( D^{RE} \subseteq RES(s(f(n))(2n + 1)) \)
The last proof was generic – every bounded simulation can be substituted.
\( B^{SPACE}_n \subseteq DSPACE(f(n) \log f(n)) \)
\( DSPACE(f(n)) \subseteq DSPACE(f(2n + 1) \log f(n)) \)

Exponentially Higher Bounds

We do the \( DTIME \)-case:

\( DTIME(f(n)) \subseteq DTIME(f(2n + 1))^2 \)
\( f(n) \geq n \)
\( f \text{ proper} \)

\( DTIME(\alpha) \subseteq DTIME(2^\alpha) \subseteq DTIME(2^{2^\alpha}) \subseteq DTIME(2^{2^\alpha}) \)
\( P \subseteq EXP \)
Relating Complexity Classes

Relationships

\[ L \subseteq NL \]
\[ P \subseteq NP \]
\[ \text{PSPACE} \subseteq \text{NSPACE} \]
\[ \text{EXP} \subseteq \text{NEXP} \]

Determinism vs. Nondeterminism

Exponentially Higher Bound

Further Relationships

\[ \text{NTIME}(f(n)) \subseteq \text{DTIME}(e^{\text{poly}(f(n))}) \]
\[ \text{NSPACE}(f(n)) \subseteq \text{DTIME}(e^{\text{poly}(f(n))}) \]

\[ f(n) \geq \log n \]

\[ f \text{ proper} \]

NTIME vs. DSPACE (Proof I)

\[ \text{NTIME}(f(n)) \subseteq \text{DSPACE}(f(n)) \]

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

NTIME vs. DSPACE (Proof II)

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

This string is then used by \( \overline{M} \) as a lookup-table whenever \( M \) is taking a nondet. choice.

NSPACE vs. DTIME (Proof I)

\[ \text{NSPACE}(f(n)) \subseteq \text{DTIME}(e^{\text{poly}(f(n))}) \]

Let \( M \) be an NTM running in space \( f(n) \).
A configuration of \( M \) has the following parts:
the state \( k \in K_w \) of \( M \)
the cursor position \( i \leq i \leq n+1 \) of \( M \)
the contents \( \in \Sigma^{(s)} \) of the tapes of \( M \)
Thus, there are \( |K_w| (n+1)^{|\Sigma^{(s)}|} \) different configs.
Using \( C_w \) we find at most \( C_w^{e^{\text{poly}(f(n))}} \) configs.
Relating Complexity Classes

**NSPACE vs. DTIME (Proof II)**

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

Now we define $G^M_u = \langle V, E \rangle$ 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).

$G^M_u, s, t$ is a REACH instance with $C_M^{\text{rev}(f(n))}$ nodes.

$G^M_u, s, t \in \text{REACH}$ iff $M(x) = 1$.

**NSPACE vs. DTIME (Proof III)**

$G^M_u, s, t$ is a REACH instance with $C_M^{\text{rev}(f(n))}$ nodes.

$G^M_u, s, t \in \text{REACH}$ iff $M(x) = 1$.

$\text{REACH} \in P$. Thus we can decide $G^M_u, s, t \in \text{REACH}$ in $\text{DTIME}(C^{\text{rev}(f(n))})$ for some constant $k$.

$\text{DTIME}(C^{\text{rev}(f(n))}) = \text{DTIME}(e^{\text{rev}(f(n))})$.

---

**NSPACE vs. DTIME**

**A Note on the Proof**

$G^M_u, s, t$ is a REACH instance with $C_M^{\text{rev}(f(n))}$ nodes.

$G^M_u, s, t \in \text{REACH}$ iff $M(x) = 1$.

The method of representing a space-bounded computation by a graph $G^M_u$ is called the "Reachability Method".

Effectively, this is a generic reduction! $\text{REACH}$ is NL-hard.

---

**NSPACE vs. DSPACE (Proof I)**

$\text{NSPACE}(f(n)) \subseteq \text{DSPACE}(f^2(n))$.

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

$G^M_u, s, t$ is a REACH instance with $C_M^{\text{rev}(f(n))}$ nodes.

$G^M_u, s, t \in \text{REACH}$ iff $M(x) = 1$.

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

$G^M_u, s, t$ is a REACH instance with $C^{f(n)}$ nodes.

$G^M_u, s, t \in \text{REACH}$ iff $M(x) = 1$.

**NSPACE vs. DSPACE (Proof II)**

$G^M_u, s, t$ is a REACH instance with $C^{f(n)}$ nodes.

$G^M_u, s, t \in \text{REACH}$ iff $M(x) = 1$.

We cannot compute the graph – it is exponential!

So how to access it?

We can compute the configurations $s$ 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)

\[ \text{PATH}(G, i, j, d) \]
if \( i, j \in E \) then return true;
if \( d = 0 \) then return false;
for \( z = 1, z \leq |V| + |+ + z) \)
if \( \text{PATH}(G, i, z, d - 1) \) and \( \text{PATH}(G, z, j, d - 1) \) then
return true;
return false;

\[ \text{PATH}(G, i, j, d) \] is true iff \( \exists \) a path from \( i \) to \( j \) of length \( \leq 2^d \)
\[ \text{PATH}(G, s, t, \log |V|) \] iff \( G, s, t \in \text{REACH} \]

Relating Complexity Classes
NSPACE vs. DSPACE (Proof IV)

\[ \text{PATH}(G, i, j, d) \]
if \( i, j \in E \) then return true;
if \( d = 0 \) then return false;
for \( z = 1, z \leq |V| + |+ + z) \)
if \( \text{PATH}(G, i, z, d - 1) \) and \( \text{PATH}(G, z, j, d - 1) \) then
return true;
return false;

Recursive depth at most \( d \)
Each "stack-frame" has size \( 3 \log |V| \)
\[ \text{PATH}(G, s, t, \log |V|) \] requires \( 3 \log^2 |V| \) space

Relating Complexity Classes
NSPACE vs. DSPACE (Proof V)

\[ <G^{O(n)}, s, t> \] is a \( \text{REACH} \) instance with \( C^{O(n)} \) nodes.
\[ <G^{O(n)}, s, t> \in \text{REACH} \) iff \( M(x) = 1 \)
\[ \text{PATH}(G, s, t, \log |V|) \] iff \( G, s, t \in \text{REACH} \)
\[ \text{PATH}(G, s, t, \log |V|) \] requires \( 3 \log^2 |V| \) space

Taken together: \( M(x) = 1 \) can be decided in
\[ \text{DSpace}(3 \log^2 (C^{O(n)})) = \text{DSpace}(f^2(n)) \]

Relating Complexity Classes
Relationships

\[ L \subseteq NL \]
\[ NL \subseteq \text{P} \]
\[ P \subseteq \text{NP} \]
\[ \text{PSpace} \subseteq \text{NSpace} \]
\[ \text{EXP} \subseteq \text{NEXP} \]

Determinism
vs.
Nondeterminism

\[ \text{Nspace}(f(n)) \subseteq \text{DSpace}(f^2(n)) \]
\[ f(n) \geq \log n \]

\[ \text{Nspace} = \text{Pspace} \]

Relating Complexity Classes
Relationships

\[ L \subseteq NL \]
\[ NL \subseteq \text{P} \]
\[ P \subseteq \text{NP} \]
\[ \text{PSpace} \subseteq \text{NSpace} \]
\[ \text{EXP} \subseteq \text{NEXP} \]

\[ L \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSpace} \subseteq \text{EXP} \subseteq \text{NEXP} \]
Relating Complexity Classes

Further Relationships

\[ L \subseteq NL \subseteq \mathcal{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXP} \subseteq \text{NEXP} \]
\[ NL \subseteq \text{PSPACE} \quad \mathcal{P} \subseteq \text{EXP} \quad \text{NP} \subseteq \text{NEXP} \]

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 \( \bar{L} = \{ x \in \Sigma^* \mid x \notin L \} \) is the associated complement language.

Thus, formally \( L \) and \( \bar{L} \) add up to \( \Sigma^* \).
However, often one defines \( \text{CircuitSAT} \) as the set of circuits which are not satisfiable.
In consequence \( \text{CircuitSAT} \cap \text{CircuitSAT}^c \) is the set of strings which encode circuits.

Complement Classes

Nondeterministic Co-Classes

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

A problem is, say, in \( \text{NP} \) 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 \( \text{coNP} \) iff there is an NTM running in poly-time, which accepts every positive instance at the end of EACH path.

Example: CIRSAT

\( \text{CIRSAT} \) can be solved with an \( \text{NP} \)-algorithm \( M \):
\( M \) guesses an assignment \( A \) for the input circuit \( C \)
\( M \) accepts iff \( A \) satisfies \( C \).
Thus \( M \) evaluates \( \exists A: C(A) = 1 \).

\( \text{CIRSAT} (\text{COMPLEMENT}) \) can be solved with
a \( \text{coNP} \)-algorithm \( M' \):
\( M' \) guesses an assignment \( A \) for the input circuit \( C \)
\( M' \) accepts iff \( A \) does not satisfy \( C \).
Thus \( M' \) evaluates \( \forall A: C(A) = 0 \).

Immerman-Szelepsenczyi Theorem

The \( \text{NTIME} \) case is open, i.e., whether
\( \text{NP} = \text{coNP} \), or \( \text{NEXP} = \text{coNEXP} \) is unknown.

We already know: \( \text{NSPACE}(f(n)) = \text{coNSPACE}(f(n)) \)
\( f(n) \geq \log n \), proper

\( \text{Immerman-Szelepsenczyi Theorem} \)
NSPACE vs. coNSPACE

Reachability Method Again

Again, we will use the reachability method:

That is, given an NTM $M$ respecting the space bound $s$ and an input string $x$, we define the configuration graph $G^s_x$.

$G^s_x$ is a REACH instance with $C^s_M$ nodes.

$G^s_x$ is in REACH iff $M(x) = 1$

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

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)|) \iff \forall v \in S(k) \quad k > 0
checkpath(G, v, k, |S(k-1)|) takes $O(\log |V|)$ space
(guesspath, count, and v require only $O(\log |V|)$)
```
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 > ∈ E then result := true;
   6. if count < last then reject; else return result;

result := last ∈ S(k - 1) ≡ all nodes in S(k - 1) have been found, otherwise line 6 rejects.

but then line 5 correctly determines whether v ∈ S(k)

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);

unreachable(G) holds.

Relating Complexity Classes

Co-Classes

NL = coNL ⊆ PSPACE = EXP ⊆ NEXP

NL ⊆ P ⊆ NP ⊆ EXP ⊆ NEXP

NL ⊆ PSPACE ⊆ EXP ⊆ NEXP

NL = coNL ⊆ P ⊆ coNP ⊆ PSPACE

Techniques

Diagonalization
DTIME(f) ⊆ DTIME(f^2)
DTIME(f) ⊆ DSPACE(f log f)

Reachability Method
NSPACE(f) ⊆ DTIME(ex^{o(f)})
NSPACE(f) ⊆ DSPACE(f^2), f ≥ log n
NSPACE(f) = coNSPACE(f), f ≥ log n

Counting
NSPACE(f) = coNSPACE(f). f ≥ log n

If yes, RSA is breakable.

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

Also unknown: Does NP ∩ coNP = P hold?