Recall our earlier definition of space complexities:

**Definition 10.1**

Let \( f : \mathbb{N} \rightarrow \mathbb{R}^+ \) be a function.

- \( \text{DSpace}(f(n)) \) is the class of all languages \( \mathcal{L} \) for which there is an \( O(f(n)) \)-space bounded Turing machine deciding \( \mathcal{L} \).
- \( \text{NSpace}(f(n)) \) is the class of all languages \( \mathcal{L} \) for which there is an \( O(f(n)) \)-space bounded nondeterministic Turing machine deciding \( \mathcal{L} \).

Being \( O(f(n)) \)-space bounded requires a (nondeterministic) TM

- to halt on every input and
- to use \( \leq f(|w|) \) tape cells on every computation path.
## Space Complexity Classes

Some important space complexity classes:

\[
\begin{align*}
L &= \text{LogSpace} = \text{DSpace}(\log n) \\
PSPACE &= \bigcup_{d \geq 1} \text{DSpace}(n^d) \\
\text{EXPSPACE} &= \bigcup_{d \geq 1} \text{DSpace}(2^{n^d}) \\
\text{NL} &= \text{NLogSpace} = \text{NSpace}(\log n) \\
\text{NPSPACE} &= \bigcup_{d \geq 1} \text{NSpace}(n^d) \\
\text{NEXPSPACE} &= \bigcup_{d \geq 1} \text{NSpace}(2^{n^d})
\end{align*}
\]

logarithmic space

polynomial space

exponential space

nondet. logarithmic space

nondet. polynomial space

nondet. exponential space

### The Power Of Space

Space seems to be more powerful than time because space can be reused.

**Example 10.2**

SAT can be solved in linear space:

Just iterate over all possible truth assignments (each linear in size) and check if one satisfies the formula.

**Example 10.3**

TAUTOLGY can be solved in linear space:

Just iterate over all possible truth assignments (each linear in size) and check if all satisfy the formula.

More generally: \( \text{NP} \subseteq \text{PSpace} \) and \( \text{coNP} \subseteq \text{PSpace} \)

### Linear Compression

**Theorem 10.4**

For every function \( f : \mathbb{N} \to \mathbb{R}^+ \), for all \( c \in \mathbb{N} \), and for every \( f \)-space bounded (deterministic/nondeterministic) Turing machine \( M \):

there is a \( \max\{1, \frac{f(n)}{c}\} \)-space bounded (deterministic/nondeterministic) Turing machine \( M' \) that accepts the same language as \( M \).

**Proof idea.**

Similar to (but much simpler than) linear speed-up.

This justifies using \( O \)-notation for defining space classes.

### Tape Reduction

**Theorem 10.5**

For every function \( f : \mathbb{N} \to \mathbb{R}^+ \) all \( k \geq 1 \) and \( L \subseteq \Sigma^* \):

If \( L \) can be decided by an \( f \)-space bounded \( k \)-tape Turing-machine, it can also be decided by an \( f \)-space bounded \( 1 \)-tape Turing-machine

**Proof idea.**

Combine tapes with a similar reduction as for time. Compress space to avoid linear increase.

Recall that we still use a separate read-only input tape to define some space complexities, such as \( \text{LogSpace} \).
Time vs. Space

Theorem 10.6
For all functions \( f : \mathbb{N} \to \mathbb{R}^+ \):
\[
\text{DTIME}(f) \subseteq \text{DSpace}(f) \quad \text{and} \quad \text{NTIME}(f) \subseteq \text{NSPACE}(f)
\]

Proof.
Visiting a cell takes at least one time step. \( \square \)

Theorem 10.7
For all functions \( f : \mathbb{N} \to \mathbb{R}^+ \) with \( f(n) \geq \log n \):
\[
\text{DSpace}(f) \subseteq \text{DTIME}(2^{O(f)}) \quad \text{and} \quad \text{NSPACE}(f) \subseteq \text{DTIME}(2^{O(f)})
\]

Proof.
Based on configuration graphs and a bound on the number of possible configurations.

Configuration Graphs

The possible computations of a TM \( M \) (on input \( w \)) form a directed graph:

- Vertices: configurations that \( M \) can reach (on input \( w \))
- Edges: there is an edge from \( C_1 \) to \( C_2 \) if \( C_1 \to_M C_2 \) (\( C_2 \) reachable from \( C_1 \) in a single step)

This yields the configuration graph

- Could be infinite in general.
- For \( f(n) \)-space bounded 2-tape TMs, there can be at most \( 2^{O(f(n))} \) vertices and \( 2 \cdot (2^{O(f(n))})^2 = 2^{O(f(n))} \) edges

A computation of \( M \) on input \( w \) corresponds to a path in the configuration graph from the start configuration to a stop configuration.

Hence, to test if \( M \) accepts input \( w \),

- construct the configuration graph and
- find a path from the start to an accepting stop configuration.

Number of Possible Configurations

Let \( M := (Q, \Sigma, \Gamma, q_0, \delta, q_{\text{start}}) \) be a 2-tape Turing machine (1 read-only input tape + 1 work tape)

Recall: A configuration of \( M \) is a quadruple \((q, p_1, p_2, x)\) where

- \( q \in Q \) is the current state,
- \( p_1 \in \mathbb{N} \) is the head position on tape \( i \), and
- \( x \in \Gamma^* \) is the tape content.

Let \( w \in \Sigma^* \) be an input to \( M \) and \( n := |w| \). Then also \( p_2 \leq n \).

If \( M \) is \( f(n) \)-space bounded we can assume \( p_2 \leq f(n) \) and \( |x| \leq f(n) \)

Hence, there are at most
\[
|Q| \cdot n \cdot f(n) \cdot |\Gamma|^{f(n)} = n \cdot 2^{O(f(n))} = 2^{O(f(n))}
\]
different configurations on inputs of length \( n \)

(the last equality requires \( f(n) \geq \log n \)).
Basic Space/Time Relationships

Applying the results of the previous slides, we get the following relations:

\[ L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq \text{NPSpace} \subseteq \text{ExpTime} \subseteq \text{NExpTime} \]

We also noted \( P \subseteq \text{coNP} \subseteq \text{PSpace} \).

Open questions:
- What is the relationship between space classes and their co-classes?
- What is the relationship between deterministic and non-deterministic space classes?

Nondeterminism in Space

Most experts think that nondeterministic TMs can solve strictly more problems when given the same amount of time than a deterministic TM:

Most believe that \( P \subset \text{NP} \).

How about nondeterminism in space-bounded TMs?

**Theorem 10.8 (Savitch’s Theorem, 1970)**

For any function \( f : \mathbb{N} \rightarrow \mathbb{R}^+ \) with \( f(n) \geq \log n \):

\[ \text{NSpace}(f(n)) \subseteq \text{DSpace}(f^2(n)). \]

That is: nondeterminism adds almost no power to space-bounded TMs!

Consequences of Savitch’s Theorem

**Savitch’s Theorem**: For any function \( f : \mathbb{N} \rightarrow \mathbb{R}^+ \) with \( f(n) \geq \log n \):

\[ \text{NSpace}(f(n)) \subseteq \text{DSpace}(f^2(n)). \]

**Corollary 10.9**

\( \text{PSPACE} = \text{NPSpace} \).

**Proof**.

\( \text{PSPACE} \subseteq \text{NPSpace} \) is clear. The converse follows since the square of a polynomial is still a polynomial.

**Corollary 10.10**

\( \text{NL} \subseteq \text{DSpace}(O(\log^2 n)) \).

Note that \( \log^2 n \notin O(\log n) \), so we do not obtain \( \text{NL} = \text{L} \) from this.

**Proving Savitch’s Theorem**

Simulating nondeterminism with more space:
- Use configuration graph of nondeterministic space-bounded TM
- Check if an accepting configuration can be reached
- Store only one computation path at a time (depth-first search)

This still requires exponential space. We want quadratic space!

What to do?

**Things we can do**:
- Store one configuration:
  - one configuration requires \( \log n + O(f(n)) \) space
  - if \( f(n) \geq \log n \), then this is \( O(f(n)) \) space
- Store \( \log n \) configurations (remember we have \( \log^2 n \) space)
- Iterate over all configurations (one by one)
Proving Savitch’s Theorem: Key Idea

To find out if we can reach an accepting configuration, we solve a slightly more general question:

**YIELDABILITY**

*Input:* TM configurations \( C_1 \) and \( C_2 \), integer \( k \)

*Problem:* Can TM get from \( C_1 \) to \( C_2 \) in at most \( k \) steps?

Approach: check if there is an intermediate configuration \( C' \) such that

1. \( C_1 \) can reach \( C' \) in \( k/2 \) steps and
2. \( C' \) can reach \( C_2 \) in \( k/2 \) steps

\( \rightarrow \) Deterministic: we can try all \( C' \) (iteration)

\( \rightarrow \) Space-efficient: we can reuse the same space for both steps

An Algorithm for Yieldability

```
01 CanYield(C1,C2,k) {
02     if k = 1 :
03         return (C1 = C2) or (C1 \( \vdash _M \) C2)
04     else if k > 1 :
05         for each configuration C of \( M \) for input size \( n \) :
06             if CanYield(C1,C,k/2) and
07                 CanYield(C,C2,k/2) :
08                 return true
09         // eventually, if no success:
10         return false
11 }
```

- We only call \( \text{CanYield} \) only with \( k \) a power of 2, so \( k/2 \in \mathbb{N} \)

Space Requirement for the Algorithm

01 \( \text{CanYield}(C_1, C_2, k) \) {
02     if \( k = 1 \) :
03         return \((C_1 = C_2) \) or \((C_1 \vdash _M C_2)\)
04     else if \( k > 1 \) :
05         for each configuration \( C \) of \( M \) for input size \( n \) :
06             if \( \text{CanYield}(C_1, C, k/2) \) and
07                 \( \text{CanYield}(C, C_2, k/2) \) :
08                 return true
09         // eventually, if no success:
10         return \text{false}
11 }

- During iteration (line 05), we store one \( C \) in \( O(f(n)) \)
- Calls in lines 06 and 07 can reuse the same space
- Maximum depth of recursive call stack: \( \log_2 k \)

Overall space usage: \( O(f(n) \cdot \log k) \)

Simulating Nondeterministic Space-Bounded TMs

*Input:* TM \( M \) that runs in \( \text{NSpace}(f(n)) \); input word \( w \) of length \( n \)

*Algorithm:*

- Modify \( M \) to have a unique accepting configuration \( C_{\text{accept}} \) when accepting, erase tape and move head to the very left
- Select \( d \) such that \( 2^{df(n)} \geq |Q| \cdot n \cdot f(n) \cdot |\Gamma|^f(n) \)
- Return \( \text{CanYield}(C_{\text{start}}, C_{\text{accept}}, k) \) with \( k = 2^{df(n)} \)

*Space requirements:*

\( \text{CanYield} \) runs in

\[
O(f(n) \cdot \log k) = O(f(n) \cdot \log 2^{df(n)}) = O(f(n) \cdot df(n)) = O(f^2(n))
\]
Did We Really Do It?

“Select \( d \) such that \( 2^{df(n)} \geq |Q| \cdot n \cdot f(n) \cdot |\Gamma|^f(n) \)

How does the algorithm actually do this?

- \( f(n) \) was not part of the input!
- Even if we knew \( f \), it might not be easy to compute!

Solution: replace \( f(n) \) by a parameter \( \ell \) and probe its value

1. Start with \( \ell = 1 \)
2. Check if \( M \) can reach any configuration with more than \( \ell \) tape cells
   (iterate over all configurations of size \( \ell + 1 \); use \text{CAYIELD} on each)
3. If yes, increase \( \ell \) by 1; goto (2)
4. Run algorithm as before, with \( f(n) \) replaced by \( \ell \)

Therefore: we don’t need to know \( f \) at all. This finishes the proof.

Open questions:
- Is Savitch’s Theorem tight?
- Are there any interesting problems in these space classes?
- We have \( \text{PSpace} = \text{NPSpace} = \text{coNPSpace} \).
  But what about \( L, NL, \text{and coNL} \)?

\( \rightsquigarrow \) the first: nobody knows; the others: see upcoming lectures

Relationships of Space and Time

Summing up, we get the following relations:

\[
L \subseteq NL \subseteq P \subseteq NP \subseteq \text{PSpace} = \text{NPSpace} \subseteq \text{ExpTime} \subseteq \text{NExpTime}
\]

We also noted \( P \subseteq \text{coNP} \subseteq \text{PSpace} \).

\( \Box \)