Review: Space Complexity Classes

Recall our earlier definitions of space complexities:

**Definition 9.1**: Let \( f : \mathbb{N} \rightarrow \mathbb{R}^+ \) be a function.

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

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.

The Power of Space

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

**Example 9.2**: \( \text{Sat} \) 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 9.3**: \( \text{Tautology} \) 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 9.4:** 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 $\max\{1, \frac{1}{f(n)}\}$-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 9.5:** 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, then 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. □

**Note:** We still use a separate read-only input tape to define some space complexities, such as LogSpace.

**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,
- $p_i \in \mathbb{N}$ 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 $p_1 \leq n$.
- 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
\[
|Q| \cdot n \cdot f(n) \cdot |\Gamma|^n = n \cdot 2^{O(f(n))} = 2^{O(f(n))}
\]
different configurations on inputs of length $n$ (the last equality requires $f(n) \geq \log n$).
**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 $C_1 \vdash M C_2$ (i.e., $C_2$ 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 $(2^{O(f(n))})^2 = 2^{O(f(n))}$ 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.

**Time vs. Space**

**Theorem 9.6**: For all functions $f : \mathbb{N} \rightarrow \mathbb{R}^+$:

\[
\text{DTime}(f) \subseteq \text{DSpace}(f) \quad \text{and} \quad \text{NTime}(f) \subseteq \text{NSpace}(f)
\]

**Proof**: Visiting a cell takes at least one time step.

**Theorem 9.7**: For all functions $f : \mathbb{N} \rightarrow \mathbb{R}^+$ 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.

**Basic Space/Time Relationships**

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

\[
L \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXPSPACE} \subseteq \text{EXPSPACE}
\]

We also noted $\text{P} \subseteq \text{coNP} \subseteq \text{PSPACE}$.

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 $\text{P} \subsetneq \text{NP}$.

How about nondeterminism in space-bounded TMs?

**Theorem 9.8 (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

Theorem 9.8 (Savitch’s Theorem, 1970): For any function \( f : \mathbb{N} \rightarrow \mathbb{R}^+ \) with \( f(n) \geq \log n \):

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

Corollary 9.9: \( \text{PSPACE} = \text{NPSPACE} \).

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

Similarly for “bigger” classes, e.g., \( \text{EXPSPACE} = \text{NEXPSPACE} \).

Corollary 9.10: \( \text{NL} \subseteq \text{DSpace}(O(\log^2 n)) \).

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

Markus Krötzsch, 12th Nov 2018

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)

Markus Krötzsch, 12th Nov 2018

An Algorithm for Yieldability

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

\[ \text{YIELDABILITY} \]

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

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

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

\[
\begin{align*}
\text{CanYield}(C_1, C_2, k) \{ \\
\text{if } k = 1 : \\
\text{return } (C_1 = C_2) \text{ or } (C_1 \vdash_M C_2) \\
\text{else if } k > 1 : \\
\text{for each configuration } C \text{ of } M \text{ for input size } n : \\
\text{if CanYield}(C_1, C, k/2) \text{ and } \\
\text{CanYield}(C, C_2, k/2) : \\
\text{return true} \\
\text{// eventually, if no success:} \\
\text{return false}
\}
\end{align*}
\]

- We only call CanYield only with \( k \) a power of 2, so \( k/2 \in \mathbb{N} \)
Space Requirement for the Algorithm

```c
int CanYield(C1, C2, k) {  
  if k = 1 :  
    return (C1 = C2) or (C1 ⊢ M C2)  
  else if k > 1 :  
    for each configuration C of M for input size n :  
      if CanYield(C1,C,k/2) and  
        CanYield(C,C2,k/2) :  
        return true  
      // eventually, if no success:  
      return false  
}
```

- During iteration (line 5), we store one \( C \) in \( O(f(n)) \)
- Calls in lines 6 and 7 can reuse the same space
- Maximum depth of recursive call stack: \( \log_2 k \)

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

Did We Really Do It?

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

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 CanYield 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. \( \square \)

Simulating Nondeterministic Space-Bounded TMs

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

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

Space requirements:
CanYield runs in space

\[ O(f(n) \cdot \log k) = O(\ell f(n) \cdot \log 2^{d(n)}) = O(\ell f(n) \cdot df(n)) = O(f^2(n)) \]

Summary: Relationships of Space and Time

Summing up, we get the following relations:

\( L \subseteq NL \subseteq P \subseteq NP \subseteq PSpace = NPSpace \subseteq ExpTime \subseteq NExpTime \)

We also noted \( P \subseteq coNP \subseteq PSpace \).

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

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