FOUNDATIONS OF COMPLEXITY THEORY

Lecture 9: Space Complexity

David Carral
Knowledge-Based Systems

TU Dresden, November 20, 2020

Review: Space Complexity Classes

Recall our earlier definitions of space complexities:

Definition 9.1: Let $f : \mathbb{N} \to \mathbb{R}^+$ be a function.

(1) 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) 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.

Space Complexity Classes

Some important space complexity classes:

$\text{L} = \text{LogSpace} = \text{DSpace}(\log n)$	logarithmic space
$\text{PSpace} = \bigcup_{d\geq 1} \text{DSpace}(n^d)$	polynomial space
$\text{ExpSpace} = \bigcup_{d\geq 1} \text{DSpace}(2^{n^d})$	exponential space
$\text{NL} = \text{NLogSpace} = \text{NSpace}(\log n)$	nondet. logarithmic space
$\text{NPSpace} = \bigcup_{d\geq 1} \text{NSpace}(n^d)$	nondet. polynomial space
$\text{NExpSpace} = \bigcup_{d\geq 1} \text{NSpace}(2^{n^d})$	nondet. exponential space

The Power of Space

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

Example 9.2: 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 9.3: 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: NP $\subseteq$ PSpace and coNP $\subseteq$ 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/nondeterminsitic) Turing machine $\mathcal{M}$:

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

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 $\mathbf{L} \subseteq \Sigma^*$:

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.

Time vs. Space

Theorem 9.6: 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. □

Theorem 9.7: 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. **Proof:** Build the configuration graph (time $2^{O(f(n))}$) and find a path from the start to an accepting stop configuration (time $2^{O(f(n))}$). □

Number of Possible Configurations

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

Recall: A configuration of $\mathcal{M}$ 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 $\mathcal{M}$ and $n := |w|$.

- Then also $p_1 \leq n$.
- If $\mathcal{M}$ 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 $f(n) \geq \log n$).

Configuration Graphs

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

- Vertices: configurations that $\mathcal{M}$ can reach (on input w)
- Edges: there is an edge from C_1 to C_2 if $C_1 \vdash_{\mathcal{M}} C_2$ (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 $(2^{O(f(n))})^2 = 2^{O(f(n))}$ edges

A computation of $\mathcal{M}$ on input w corresponds to a path in the configuration graph from the start configuration to a stop configuration.

Hence, to test if $\mathcal{M}$ 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 $f : \mathbb{N} \to \mathbb{R}^+$:

$$\mathsf{DTime}(f) \subseteq \mathsf{DSpace}(f) \quad \text{and} \quad \mathsf{NTime}(f) \subseteq \mathsf{NSpace}(f)$$

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

Theorem 9.7: For all functions $f : \mathbb{N} \to \mathbb{R}^+$ with $f(n) \geq \log n$:

$$\mathsf{DSpace}(f) \subseteq \mathsf{DTime}(2^{O(f)}) \quad \text{and} \quad \mathsf{NSpace}(f) \subseteq \mathsf{DTime}(2^{O(f)})$$

Proof: Based on configuration graphs and a bound on the number of possible configurations. **Proof:** Build the configuration graph (time $2^{O(f(n))}$) and find a path from the start to an accepting stop configuration (time $2^{O(f(n))}$). □

Basic Space/Time Relationships

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

$$\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSpace} \subseteq \mathsf{NPSpace} \subseteq \mathsf{ExpTime} \subseteq \mathsf{NExpTime}$$

We also noted $\mathsf{P} \subseteq \mathsf{coNP} \subseteq \mathsf{PSpace}$.

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 $\mathsf{P} \subsetneq \mathsf{NP}$

How about nondeterminism in space-bounded TMs?

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

$$\mathsf{NSpace}(f(n)) \subseteq \mathsf{DSpace}(f^2(n)).$$

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 $f : \mathbb{N} \to \mathbb{R}^+$ with $f(n) \geq \log n$:

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

Corollary 9.9: PSpace = NPSpace.

Proof: PSpace $\subseteq$ NPSpace is clear. The converse follows since the square of a polynomial is still a polynomial. □

Similarly for "bigger" classes, e.g., ExpSpace = NExpSpace.

Corollary 9.10: NL $\subseteq$ DSpace($O(\log^2 n)$).

Note that $\log^2(n) \notin O(\log n)$, so we do not obtain NL = 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 $f(n)$ configurations (remember we have $f^2(n)$ 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 slighly 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

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

$\leadsto$ Space-efficient: we can reuse the same space for both steps

An Algorithm for Yieldability

```
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
}
```

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

Space Requirement for the Algorithm

```
01 CanYield(C1,C2,k) {
02   if k = 1 :
03     return (C1 = C2) or (C1 ⊢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 }
```

- 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 $\mathcal{M}$ that runs in NSpace($f(n)$); input word w of length n

Algorithm:

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

Space requirements:
CanYield runs in space

$$O(f(n) \cdot \log k) = O\left(f(n) \cdot \log 2^{df(n)}\right) = O(f(n) \cdot df(n)) = O(f^2(n))$$

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 ℓ and probe its value

(1) Start with $\ell = 1$

(2) Check if $\mathcal{M}$ can reach any configuration with more than ℓ tape cells (iterate over all configurations of size $\ell + 1$; use CanYield on each)

(3) If yes, increase ℓ by 1; goto (2)

(4) Run algorithm as before, with $f(n)$ replaced by ℓ

Therefore: we don't need to know f at all. This finishes the proof. □

Summary: Relationships of Space and Time

Summing up, we get the following relations:

L ⊆ NL ⊆ P ⊆ NP ⊆ PSpace = NPSpace ⊆ ExpTime ⊆ NExpTime

We also noted P ⊆ coNP ⊆ PSpace.

Open questions:

- Is Savitch's Theorem tight?
- Are there any interesting problems in these space classes?
- We have PSpace = NPSpace = coNPSpace.
  But what about L, NL, and coNL?

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