COMP4141 Theory of Computation

Complexity Hierarchy

Ron van der Meyden

CSE, UNSW

Revision: 2016/05/11

We have introduced several complexity classes, and have shown results of the form $\mathcal{C} \subseteq \mathcal{C}'$, but have generally had to say "we don't know if this containment is strict."

In general, separation results seem to be hard to prove.

But we do know *some*.

Reminder: big Oh

Let $f, g : \mathbb{N} \longrightarrow \mathbb{N}$.

Definition

$f = O(g)$ if there exists constants $c, n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ we have $f(n) < c \cdot g(n)$.

Little oh

Definition

$f = o(g)$ if

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$$

or, equivalently, for all real constants $c > 0$ there exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$ we have $f(n) < c \cdot g(n)$.

Examples:

- $\frac{1}{n} = o(1)$
- $\sqrt{n} = o(n)$
- $n^k = o(n^{k+1})$
- $n \cdot \log n = o(n^2)$

Space Constructibility

To state a space hierarchy theorem, we need the following technicality:

Definition

A function $f : \mathbb{N} \longrightarrow \mathbb{N}$ with $f(n) \geq \log n$ is *space constructible* if there exists a TM M that runs in space $O(f(n))$ that on input 1^n computes a binary representation of $f(n)$.

Space Hierarchy Theorem

Theorem

Let $f : \mathbb{N} \longrightarrow \mathbb{N}$ *be space constructible. Then there exists a language* A *that is decidable in space* $O(f(n))$ *but not in space* $o(f(n))$*.*

Corollary

if $f_1, f_2 : \mathbb{N} \longrightarrow \mathbb{N}$ *with* $f_1 = o(f_2)$ *and* f_2 *is space constructible, then* $\mathbf{SPACE}(f_1) \subsetneq \mathbf{SPACE}(f_2)$

Examples:

- for $0 \leq \epsilon_1 < \epsilon_2$, $\mathbf{SPACE}(n^{\epsilon_1}) \subsetneq \mathbf{SPACE}(n^{\epsilon_2})$
- $\mathbf{SPACE}((\log n)^2) \subsetneq \mathbf{SPACE}(n)$
- Since $\mathbf{NL} \subseteq \mathbf{SPACE}((\log n)^2)$ by Savitch, we derive $\mathbf{NL} \subsetneq \mathbf{SPACE}(n)$ and $\mathbf{NL} \subsetneq \mathbf{PSPACE}$
- $\mathbf{SPACE}(n^{\log n}) \subsetneq \mathbf{SPACE}(2^n)$
- Since $\mathbf{SPACE}(n^k) \subseteq \mathbf{SPACE}(n^{\log n})$ for all k, we derive $\mathbf{PSPACE} \subsetneq \mathbf{EXPSPACE} = \bigcup_k \mathbf{SPACE}(2^{n^k})$

Proof of Space Hierarchy Theorem I

Intuition: by Diagonalization. We construct a language $A \in \textbf{SPACE}(f(n))$ that differs from the language accepted by any $o(f(n))$ space machine M on at least one string w_M.

In particular, we would like to pick $w_M = \langle M \rangle$ and have $\langle M \rangle \notin A$ iff M accepts $\langle M \rangle$.

Problem:

(1) Simulating an arbitrary M that runs in space g within a fixed TM D requires space $c_M \cdot g(n)$ for some constant c_M that depends on M, since M may have more tape symbols than D.

(2) $g = o(f)$ implies that $g(n) < \frac{1}{c_M} \cdot f(n)$, which resolves (1), but only for $n \geq n_0$, whereas we may have $|\langle M \rangle| < n_0$.

Solution: Simulate M for *multiple* inputs of the form $w = \langle M \rangle \# 1^k$.

Proof of Space Hierarchy Theorem II

Let A be the language decided by the following machine D that uses space $O(f(n))$:

$D =$
"On input w with $|w| = n$:

1. If w is of the form $\langle M \rangle \# 1^k$ for some TM M, continue, else *reject*
2. Compute $f(n)$ and mark out space $f(n)$. If later steps leave this area, *reject*
3. Simulate M on input w while counting computation steps:
   - if the count exceeds $2^{f(n)}$, reject
   - if M accepts, reject
   - if M rejects, accept"

Proof of Space Hierarchy Theorem III

(**Comment:** It is *undecidable* whether a machine runs in space $o(f(n))$. But A doesn't care about machines that take more than this, we just need to make sure we have covered at least the space $o(f(n))$ ones.)

Suppose M runs in space $g = o(f)$. We show by contradiction that M does not decide A. Suppose it does.

Let c_M be the tape symbol simulation cost factor for D to simulate M.

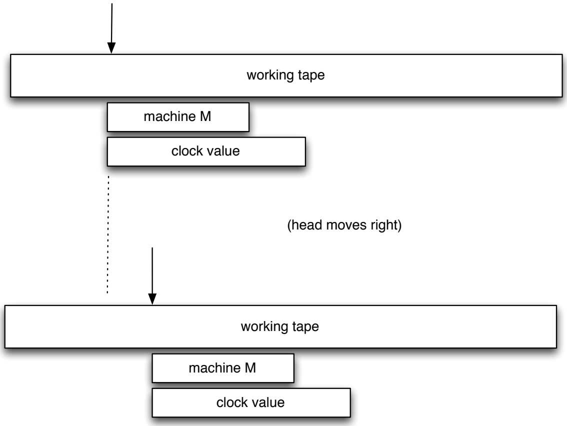
For some n_0 we have $n \geq n_0$ implies $c_M \cdot g(n) < f(n)$, so on input $w = \langle M \rangle \# 1^{n_0}$, the simulation of M on w runs in space $f(n)$ and terminates.

But then the decision of $D(w)$ and $M(w)$ are the opposite. So M does not decide $A = L(D)$. Contradiction.

Can we get a similar separation result for time complexity classes?

Complication: simulating *one-tape* TM's for a time-bound t requires moving the head backwards and forwards between representations of the working tape, machine being simulated, and clock bits. This costs time!

Key idea: encode three tracks of information in each tape symbol of the simulating machine: working tape symbol, machine, clock. At each simulation step, move the machine and clock representations so that they stay near the position of the head on the working tape.



Now the overhead of simulation is a time factor of $O(|\langle M\rangle| + \log t)$.

Time Hierarchy Theorem

Definition

A function $t : \mathbb{N} \longrightarrow \mathbb{N}$ with $t(n) \geq n \log n$ is *time constructible* if the function that maps string 1^n to the binary representation of $t(n)$ is computable in time $O(t(n))$.

Theorem (Time Hierarchy Theorem)

For any time constructible function $t : \mathbb{N} \longrightarrow \mathbb{N}$ there exists a language A that is decidable in time $O(t(n))$ but not decidable in time $o(\frac{t(n)}{\log t(n)})$.

Proof.

Similar to proof of space hierachy theorem, but using the $|\langle M \rangle| + \log t$ simulation trick. The $|\langle M \rangle|$ is constant where it is needed in the proof. (See Sipser Thm 9.10) □

Corollary

If $t_1, t_2 : \mathbb{N} \longrightarrow \mathbb{N}$ *satisfy* $t_1(n) = o(t_2(n)/\log t_2(n))$ *and* t_2 *is time constructible, then* $\mathbf{TIME}(t_1(n)) \subsetneq \mathbf{TIME}(t_2(n))$*.*

Examples:

- For real numbers $1 \le \epsilon_1 < \epsilon_2$, we have $\mathbf{TIME}(n^{\epsilon_1}) \subsetneq \mathbf{TIME}(n^{\epsilon_2})$.
- $\mathbf{P} \subsetneq \mathbf{EXPTIME} = \bigcup_k \mathbf{TIME}(2^{n^k})$
- The problem of equivalence of regular expressions using the additional construct

  $$R \uparrow k = R \cdot R \cdot \ldots \cdot R \qquad \text{(concatenate } k \text{ copies of } R\text{)}$$

  is **EXPTIME**-complete, so is not in **P**. (Sipser Thm 9.15)