TECHNISCHE UNIVERSITÄT DRESDEN

COMPLEXITY THEORY

Lecture 12: Hierarchy Theorems

Markus Krötzsch

Knowledge-Based Systems

TU Dresden, 20th Nov 2018

Review

Relationships of Complexity Classes

$$\begin{array}{ccccccccccccc}
\mathsf{L} & \subseteq & \mathsf{NL} & \subseteq & \mathsf{P} & \subseteq & \mathsf{NP} & \subseteq & \mathsf{PSpace} & = & \mathsf{NPSpace} & \subseteq & \mathsf{Exp} \\
\rotatebox{90}{$=$} & & \rotatebox{90}{$=$} & & \rotatebox{90}{$=$} & & ? & & \rotatebox{90}{$=$} & & \rotatebox{90}{$=$} & & \rotatebox{90}{$=$} \\
\mathsf{coL} & \subseteq & \mathsf{coNL} & \subseteq & \mathsf{coP} & \subseteq & \mathsf{coNP} & \subseteq & \mathsf{coPSpace} & = & \mathsf{coNPSpace} & \subseteq & \mathsf{coExp}
\end{array}$$

Obvious question:

Are any of these $\subseteq$ strict $\subsetneq$?

Relationships of Complexity Classes

Relating different complexity classes is a central goal of complexity theory

Complexity classes differ by:

- Underlying machine model (e.g., DTM vs. NTM)
- Restricted resource (e.g., time or space)
- Resource bound (e.g., polynomial or exponential)

Relationships of Complexity Classes

Relating different complexity classes is a central goal of complexity theory

Complexity classes differ by:

- Underlying machine model (e.g., DTM vs. NTM)
- Restricted resource (e.g., time or space)
- Resource bound (e.g., polynomial or exponential)

Some intuitions:

- Nondeterminism seems to add some more power
- Space seems to be more powerful than time
- More resources seem to add more power

Relationships of Complexity Classes

Relating different complexity classes is a central goal of complexity theory

Complexity classes differ by:

- Underlying machine model (e.g., DTM vs. NTM)
- Restricted resource (e.g., time or space)
- Resource bound (e.g., polynomial or exponential)

Some intuitions:

- Nondeterminism seems to add some more power
- Space seems to be more powerful than time
- More resources seem to add more power

However: it is often difficult to confirm these intuitive ideas formally
(and many of them remain unproven)

Something we do know

At least one thing is known to be (mostly) true:

Intuition: If we allow a given Turing machine to use strictly more time or space, then it can really solve strictly more problems.

- This is not always true, but it is true for “reasonable” ways of defining resource bounds, such as polynomial or exponential
- We will formalise and prove it later today
- The proof method we will use is called diagonalisation

Diagonalisation

Review: Cantor's Argument

Diagonalisation is the basis of a well known argument to show that the powerset $2^{\mathcal{S}}$ of a countable set $\mathcal{S}$ is not countable

Proof: Suppose for a contradiction that $2^{\mathcal{S}}$ is countable.

- Then the sets in $2^{\mathcal{S}}$ can be enumerated in a list $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \ldots \subseteq \mathcal{S}$
- Let us write this list as boolean matrix with rows representing the sets $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \ldots$, columns representing a (countably infinite) enumeration of $\mathcal{S}$, and boolean entries encoding the $\in$ relationship.
- For a contradiction, define a set $\mathcal{S}_d$ by diagonalisation to differ from all other $\mathcal{S}_i$ in the enumeration:

	s_1	s_2	s_3	$\ldots$
$\mathcal{S}_1$	×			$\ldots$
$\mathcal{S}_2$			×	$\ldots$
$\mathcal{S}_3$	×	×		$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
$\mathcal{S}_d$		×	×	$\ldots$

□

Review: The Halting Problem

We have used a similar argument to show undecidability of the Halting problem:

Proof: Suppose for a contradiction that Halting is decidable.

- Then set of all Turing machines can be enumerated in a list $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \ldots$
- We are interested in their halting on inputs of the form $\langle\mathcal{M}\rangle$ for some TM $\mathcal{M}$
- We can write it as a boolean matrix with rows representing the TMs $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \ldots$, columns representing a (countably infinite) enumeration of strings $\langle\mathcal{M}\rangle$, and boolean entries encoding if TM halts.
- Using a decider for the halting problem, we can define a TM $\mathcal{M}_d$ by diagonalisation to differ from all other $\mathcal{M}_i$ in the enumeration:

	$\langle\mathcal{M}_1\rangle$	$\langle\mathcal{M}_2\rangle$	$\langle\mathcal{M}_3\rangle$	$\ldots$
$\mathcal{M}_1$	×			$\ldots$
$\mathcal{M}_2$			×	$\ldots$
$\mathcal{M}_3$	×	×		$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
$\mathcal{M}_d$		×	×	$\ldots$

□

Generalising Diagonalisation

To generalise diagonalisation as a method for complexity classes, we consider arbitrary resources (time, space, . . .):

Definition 12.1: Given a class $\mathcal{K}$ of Turing machines (e.g., 2-tape deterministic TMs), R is a resource (e.g., time or space) defined for all machines in $\mathcal{K}$ if $\mathrm{R}_{\mathcal{M}}(w) \in \mathbb{N} \cup \{\infty\}$ for all $\mathcal{M} \in \mathcal{K}$ and all words w.

Then, any function $f : \mathbb{N} \to \mathbb{N}$ gives rises to a class of languages:

$$\mathrm{R}(f) = \{\mathbf{L} \mid \text{there is } \mathcal{M} \in \mathcal{K} \text{ with } \mathbf{L}(\mathcal{M}) = \mathbf{L} \text{ and } \mathrm{R}_{\mathcal{M}}(w) \leq f(|w|) \text{ for all } w \in \Sigma^*\}.$$

Generalising Diagonalisation

To generalise diagonalisation as a method for complexity classes, we consider arbitrary resources (time, space, . . .):

Definition 12.1: Given a class $\mathcal{K}$ of Turing machines (e.g., 2-tape deterministic TMs), R is a resource (e.g., time or space) defined for all machines in $\mathcal{K}$ if $\mathrm{R}_{\mathcal{M}}(w) \in \mathbb{N} \cup \{\infty\}$ for all $\mathcal{M} \in \mathcal{K}$ and all words w.

Then, any function $f : \mathbb{N} \to \mathbb{N}$ gives rises to a class of languages:

$$\mathrm{R}(f) = \{\mathbf{L} \mid \text{there is } \mathcal{M} \in \mathcal{K} \text{ with } \mathbf{L}(\mathcal{M}) = \mathbf{L} \text{ and } \mathrm{R}_{\mathcal{M}}(w) \leq f(|w|) \text{ for all } w \in \Sigma^*\}.$$

Example 12.2: We will use, e.g., the following resources:

- DTime time used by a deterministic 1-tape TM
- DTime_k time used by a deterministic k-tape TM
- DTime_* time used by a deterministic TM with any number of tapes

Diagonalisation of Resources

Consider resources R_1 and R_2 for two classes of Turing machines $\mathcal{K}_1$ and $\mathcal{K}_2$, and two functions $f_1, f_2 : \mathbb{N} \to \mathbb{N}$.

Definition 12.3: We say that $\mathrm{R}_1(f_1)$ allows diagonalisation over $\mathrm{R}_2(f_2)$ if there exists a Turing machine $\mathcal{D} \in \mathcal{K}_1$ such that

- $\mathbf{L}(\mathcal{D}) \in \mathrm{R}_1(f_1)$, and
- for each $\mathcal{M} \in \mathcal{K}_2$ that is R_2-bounded by f_2, there exists a w such that $\langle\mathcal{M}, w\rangle \in \mathbf{L}(\mathcal{D})$ if and only if $\langle\mathcal{M}, w\rangle \notin \mathbf{L}(\mathcal{M})$.

Diagonalisation of Resources

Consider resources R_1 and R_2 for two classes of Turing machines $\mathcal{K}_1$ and $\mathcal{K}_2$, and two functions $f_1, f_2 : \mathbb{N} \to \mathbb{N}$.

Definition 12.3: We say that $\mathrm{R}_1(f_1)$ allows diagonalisation over $\mathrm{R}_2(f_2)$ if there exists a Turing machine $\mathcal{D} \in \mathcal{K}_1$ such that

- $\mathbf{L}(\mathcal{D}) \in \mathrm{R}_1(f_1)$, and
- for each $\mathcal{M} \in \mathcal{K}_2$ that is R_2-bounded by f_2, there exists a w such that $\langle \mathcal{M}, w\rangle \in \mathbf{L}(\mathcal{D})$ if and only if $\langle \mathcal{M}, w\rangle \notin \mathbf{L}(\mathcal{M})$.

Theorem 12.4: If $\mathrm{R}_1(f_1)$ allows diagonalisation over $\mathrm{R}_2(f_2)$, then $\mathrm{R}_1(f_1) \setminus \mathrm{R}_2(f_2) \neq \emptyset$.

Diagonalisation of Resources

Consider resources R_1 and R_2 for two classes of Turing machines $\mathcal{K}_1$ and $\mathcal{K}_2$, and two functions $f_1, f_2 : \mathbb{N} \to \mathbb{N}$.

Definition 12.3: We say that $\mathrm{R}_1(f_1)$ allows diagonalisation over $\mathrm{R}_2(f_2)$ if there exists a Turing machine $\mathcal{D} \in \mathcal{K}_1$ such that

- $\mathbf{L}(\mathcal{D}) \in \mathrm{R}_1(f_1)$, and
- for each $\mathcal{M} \in \mathcal{K}_2$ that is R_2-bounded by f_2, there exists a w such that $\langle \mathcal{M}, w \rangle \in \mathbf{L}(\mathcal{D})$ if and only if $\langle \mathcal{M}, w \rangle \notin \mathbf{L}(\mathcal{M})$.

Theorem 12.4: If $\mathrm{R}_1(f_1)$ allows diagonalisation over $\mathrm{R}_2(f_2)$, then $\mathrm{R}_1(f_1) \setminus \mathrm{R}_2(f_2) \neq \emptyset$.

Proof: Let $\mathcal{D}$ be as in Definition 12.3. We show $\mathbf{L}(\mathcal{D}) \notin \mathrm{R}_2(f_2)$.

(1) Suppose for a contradiction that there is $\mathcal{M} \in \mathcal{K}_2$ that is R_2-bounded by f_2 with $\mathbf{L}(\mathcal{D}) = \mathbf{L}(\mathcal{M})$.

Diagonalisation of Resources

Consider resources R_1 and R_2 for two classes of Turing machines $\mathcal{K}_1$ and $\mathcal{K}_2$, and two functions $f_1, f_2 : \mathbb{N} \to \mathbb{N}$.

Definition 12.3: We say that $\mathrm{R}_1(f_1)$ allows diagonalisation over $\mathrm{R}_2(f_2)$ if there exists a Turing machine $\mathcal{D} \in \mathcal{K}_1$ such that

- $\mathbf{L}(\mathcal{D}) \in \mathrm{R}_1(f_1)$, and
- for each $\mathcal{M} \in \mathcal{K}_2$ that is R_2-bounded by f_2, there exists a w such that $\langle \mathcal{M}, w\rangle \in \mathbf{L}(\mathcal{D})$ if and only if $\langle \mathcal{M}, w\rangle \notin \mathbf{L}(\mathcal{M})$.

Theorem 12.4: If $\mathrm{R}_1(f_1)$ allows diagonalisation over $\mathrm{R}_2(f_2)$, then $\mathrm{R}_1(f_1) \setminus \mathrm{R}_2(f_2) \neq \emptyset$.

Proof: Let $\mathcal{D}$ be as in Definition 12.3. We show $\mathbf{L}(\mathcal{D}) \notin \mathrm{R}_2(f_2)$.

(1) Suppose for a contradiction that there is $\mathcal{M} \in \mathcal{K}_2$ that is R_2-bounded by f_2 with $\mathbf{L}(\mathcal{D}) = \mathbf{L}(\mathcal{M})$.

(2) We obtain a contradiction, since, by Definition 12.3, there is a word w such that

$$\langle \mathcal{M}, w\rangle \in \mathbf{L}(\mathcal{D}) = \mathbf{L}(\mathcal{M}) \text{ if and only if } \langle \mathcal{M}, w\rangle \notin \mathbf{L}(\mathcal{M})$$

□

Hierarchy Theorems

Reasonable bounds

What kind of functions should we consider as resource bounds?

- Functions $f : \mathbb{N} \to \mathbb{N}$ can be very weird
- The intuition "higher bound $\Rightarrow$ more power" turns out to be wrong in general

Reasonable bounds

What kind of functions should we consider as resource bounds?

- Functions $f : \mathbb{N} \to \mathbb{N}$ can be very weird
- The intuition "higher bound ⇒ more power" turns out to be wrong in general

However, our intuition can be confirmed for "reasonable" functions:

Definition 12.5: A function $t : \mathbb{N} \to \mathbb{N}$ is time-constructible if $t(n) \geq n$ for all n and there exists a TM that computes $t(n)$ in unary in time $O(t(n))$.

A function $s : \mathbb{N} \to \mathbb{N}$ is space-constructible if $s(n) \geq \log n$ and there exists a TM that computes $s(n)$ in unary in space $O(s(n))$.

Note 1: We do consider arbitrary deterministic multi-tape TMs here.

Note 2: A TM that computes $f(n)$ "in unary" takes n as input and writes a symbol (say `x`) $f(n)$ times before terminating

Time and space constructible functions

There are alternative definitions of time and space constructibility in the literature, but the general intuition is similar:

- Time-constructible: Computing f does not require significantly more time than the resulting value of f
- Space-constructible: Computing f does not require significantly more space than the resulting value of f

Time and space constructible functions

There are alternative definitions of time and space constructibility in the literature, but the general intuition is similar:

- Time-constructible: Computing f does not require significantly more time than the resulting value of f
- Space-constructible: Computing f does not require significantly more space than the resulting value of f

All functions commonly used to bound time or space satisfy these criteria:

Theorem 12.6: If f and g are time-constructible (space-constructible), then so are $f + g$, $f \cdot g$, 2^f, and f^g. Moreover, the following common functions have these properties:

- n^d $(d \geq 1)$, b^n $(b \geq 2)$, and $n!$ are time-constructible
- $\log n$, n^d $(d \geq 1)$, b^n $(b \geq 2)$, and $n!$ are space-constructible

Using Constructibility to Halt

We had required time-bounded nondeterministic TMs to halt on all computation paths, even if not accepting. — Is this really necessary?

Using Constructibility to Halt

We had required time-bounded nondeterministic TMs to halt on all computation paths, even if not accepting. — Is this really necessary?

Theorem 12.7: Given a time-constructible function f and an NTM $\mathcal{M}$, one can construct an $O(f)$-time bounded NTM $\mathcal{M}'$ that accepts exactly those words $|w|$ that $\mathcal{M}$ accepts in $f(|w|)$ steps.

Consequences: (1) we can enforce timely termination on unsuccessful paths; (2) if we have at least polynomial time, this can also be achieved with only one tape.

Using Constructibility to Halt

We had required time-bounded nondeterministic TMs to halt on all computation paths, even if not accepting. — Is this really necessary?

Theorem 12.7: Given a time-constructible function f and an NTM $\mathcal{M}$, one can construct an $O(f)$-time bounded NTM $\mathcal{M}'$ that accepts exactly those words $|w|$ that $\mathcal{M}$ accepts in $f(|w|)$ steps.

Consequences: (1) we can enforce timely termination on unsuccessful paths; (2) if we have at least polynomial time, this can also be achieved with only one tape.

Proof: On input w, $\mathcal{M}'$ operates as follows:

(1) Compute $f(|w|)$ on a separate tape (creating $f(|w|)$ symbols). This can be done in $O(f(|w|))$ time.

(2) Perform the same transitions as $\mathcal{M}$ (on dedicated tapes) while "counting down" the $f(|w|)$ symbols in each step

(3) Terminate if either $\mathcal{M}$ terminates (in this case return its result) or if the countdown reaches 0 (in this case reject). □

Doing more by using more tapes

We first show a preliminary result: “more time + more tapes = more power”

Theorem 12.8: Let $f, g : \mathbb{N} \to \mathbb{N}$ such that f is time-constructible, and $g \in o(f)$. Then, for all $k \in \mathbb{N}$, we have

$$\mathsf{DTime}_k(g) \subsetneq \mathsf{DTime}_*(f)$$

Doing more by using more tapes

We first show a preliminary result: "more time + more tapes = more power"

Theorem 12.8: Let $f, g : \mathbb{N} \to \mathbb{N}$ such that f is time-constructible, and $g \in o(f)$. Then, for all $k \in \mathbb{N}$, we have

$$\mathsf{DTime}_k(g) \subsetneq \mathsf{DTime}_*(f)$$

Proof: Clearly, $\mathsf{DTime}_k(g) \subseteq \mathsf{DTime}_k(f) \subseteq \mathsf{DTime}_*(f)$. We get $\mathsf{DTime}_*(f) \neq \mathsf{DTime}_k(g)$ by showing that that $\mathsf{DTime}_*(f)$ allows diagonalisation over $\mathsf{DTime}_k(g)$.

We define a multi-tape TM $\mathcal{D}$ for inputs of the form $\langle \mathcal{M}, w \rangle$ (other cases do not matter):

- Compute $f(|\langle \mathcal{M}, w \rangle|)$ in unary on a separate "countdown" tape
- Simulate $\mathcal{M}$ on $\langle \mathcal{M}, w \rangle$, using an appropriate number of tapes (see Theorem 3.8).
- Time-bound the simulation by $f(|\langle \mathcal{M}, w \rangle|)$ using the countdown tape as in Theorem 12.7
- If $\mathcal{M}$ rejects (in this time bound), then accept;
  otherwise, if $\mathcal{M}$ accepts or fails to stop (in the bounded time), reject

The countdown ensures that $\mathcal{D}$ runs in $O(f)$ time, i.e. $\mathbf{L}(\mathcal{D}) \in \mathsf{DTime}_*(f)$.

Doing more by using more tapes (2)

We first show a preliminary result: "more time + more tapes = more power"

Theorem 12.8: Let $f, g : \mathbb{N} \to \mathbb{N}$ such that f is time-constructible, and $g \in o(f)$. Then, for all $k \in \mathbb{N}$, we have

$$\mathsf{DTime}_k(g) \subsetneq \mathsf{DTime}_*(f)$$

Proof (continued): To invoke Theorem 12.4, we still have to show that, for every k-tape TM $\mathcal{M}$ that is g-time bounded, there is a word w such that

$$\langle \mathcal{M}, w \rangle \in \mathbf{L}(\mathcal{D}) \text{ if and only if } \langle \mathcal{M}, w \rangle \notin \mathbf{L}(\mathcal{M})$$

Doing more by using more tapes (2)

We first show a preliminary result: "more time + more tapes = more power"

Theorem 12.8: Let $f, g : \mathbb{N} \to \mathbb{N}$ such that f is time-constructible, and $g \in o(f)$. Then, for all $k \in \mathbb{N}$, we have

$$\mathsf{DTime}_k(g) \subsetneq \mathsf{DTime}_*(f)$$

Proof (continued): To invoke Theorem 12.4, we still have to show that, for every k-tape TM $\mathcal{M}$ that is g-time bounded, there is a word w such that

$$\langle \mathcal{M}, w\rangle \in \mathbf{L}(\mathcal{D}) \text{ if and only if } \langle \mathcal{M}, w\rangle \notin \mathbf{L}(\mathcal{M})$$

For this, we need to show that there is a word w for which $\mathcal{D}$'s simulation of $\mathcal{M}$ will terminate on time:

- For all $\mathcal{M}$, there is a constant number $c_{\mathcal{M}}$ of steps that $\mathcal{D}$ will at most need to simulate one step of $\mathcal{M}$ (this depends on the size of $\mathcal{M}$)
- Since $g \in o(f)$ there is a number n_0 such that $f(n) \geq c_{\mathcal{M}} \cdot g(n)$ for all $n \geq n_0$.
- Therefore, for all (infinitely many) words w with $|\langle \mathcal{M}, w\rangle| \geq n_0$, $\mathcal{D}$'s simulation of $\mathcal{M}$ will terminate. □

A Time Hierarchy Theorem

We can now show that (sufficiently) more time always allows us to solve strictly more problems, even if we are allowed to use any number of tapes (i.e., the advantage of having more time cannot be compensated by adding more tapes):

Time Hierarchy Theorem (weaker version) 12.9: If $f, g : \mathbb{N} \to \mathbb{N}$ are such that f is time-constructible, and $g^2 \in o(f)$, then

$$\text{DTime}_*(g) \subsetneq \text{DTime}_*(f)$$

Proof: Since $\text{DTime}_k(g) \subseteq \text{DTime}_*(f)$ for all k, it is clear that $\text{DTime}_*(g) \subseteq \text{DTime}_*(f)$.

A Time Hierarchy Theorem

We can now show that (sufficiently) more time always allows us to solve strictly more problems, even if we are allowed to use any number of tapes (i.e., the advantage of having more time cannot be compensated by adding more tapes):

Time Hierarchy Theorem (weaker version) 12.9: If $f, g : \mathbb{N} \to \mathbb{N}$ are such that f is time-constructible, and $g^2 \in o(f)$, then

$$\mathsf{DTime}_*(g) \subsetneq \mathsf{DTime}_*(f)$$

Proof: Since $\mathsf{DTime}_k(g) \subseteq \mathsf{DTime}_*(f)$ for all k, it is clear that $\mathsf{DTime}_*(g) \subseteq \mathsf{DTime}_*(f)$.

But Theorem 12.8 does not show $\mathsf{DTime}_*(g) \neq \mathsf{DTime}_*(f)$: it could be that every problem in $\mathsf{DTime}_k(f)$ is also in $\mathsf{DTime}_{k'}(g)$ for a big enough k'.

A Time Hierarchy Theorem

We can now show that (sufficiently) more time always allows us to solve strictly more problems, even if we are allowed to use any number of tapes (i.e., the advantage of having more time cannot be compensated by adding more tapes):

Time Hierarchy Theorem (weaker version) 12.9: If $f, g : \mathbb{N} \to \mathbb{N}$ are such that f is time-constructible, and $g^2 \in o(f)$, then

$$\mathrm{DTime}_*(g) \subsetneq \mathrm{DTime}_*(f)$$

Proof: Since $\mathrm{DTime}_k(g) \subseteq \mathrm{DTime}_*(f)$ for all k, it is clear that $\mathrm{DTime}_*(g) \subseteq \mathrm{DTime}_*(f)$.

But Theorem 12.8 does not show $\mathrm{DTime}_*(g) \neq \mathrm{DTime}_*(f)$: it could be that every problem in $\mathrm{DTime}_k(f)$ is also in $\mathrm{DTime}_{k'}(g)$ for a big enough k'.

- Multi-tape TMs can be transformed into single tape TMs with quadratic time overhead (Theorem 5.10), hence $\mathrm{DTime}_*(g) \subseteq \mathrm{DTime}_1(g^2)$.
- By Theorem 12.8, $\mathrm{DTime}_1(g^2) \subsetneq \mathrm{DTime}_*(f)$ since $g^2 \in o(f)$
- Hence $\mathrm{DTime}_*(g) \subsetneq \mathrm{DTime}_*(f)$ □

Exponential jumps make a difference

Corollary 12.10: $\mathsf{P} \subsetneq \mathsf{ExpTime}$.

Exponential jumps make a difference

Corollary 12.10: P $\subsetneq$ ExpTime.

Proof:

- For every polynomial p, we have $p(n) \in o(2^n)$, so P $\subseteq$ DTime$(2^n) \subsetneq$ ExpTime

Exponential jumps make a difference

Corollary 12.10: P $\subsetneq$ ExpTime.

Proof:

- For every polynomial p, we have $p(n) \in o(2^n)$, so P $\subseteq$ DTime$(2^n) \subseteq$ ExpTime

  Note: of course, we also have $p^2 \in o(2^n)$, but this only shows that DTime$(p) \subsetneq$ ExpTime holds for specific polynomials, rather than P $\subsetneq$ ExpTime for the union over all polynomials.

- For proper inclusion, note $(2^n)^2 = 2^{2n} \in o(2^{n^2})$, so DTime$(2^n) \subsetneq$ DTime(2^{n^2})

Exponential jumps make a difference

Corollary 12.10: P $\subsetneq$ ExpTime.

Proof:

- For every polynomial p, we have $p(n) \in o(2^n)$, so P $\subseteq$ DTime(2^n) $\subseteq$ ExpTime

  Note: of course, we also have $p^2 \in o(2^n)$, but this only shows that DTime(p) $\subsetneq$ ExpTime holds for specific polynomials, rather than P $\subsetneq$ ExpTime for the union over all polynomials.

- For proper inclusion, note $(2^n)^2 = 2^{2n} \in o(2^{n^2})$, so DTime$(2^n)$ $\subsetneq$ DTime(2^{n^2})
- In summary:

$$\mathsf{P} \subseteq \mathsf{DTime}(2^n) \subsetneq \mathsf{DTime}(2^{n^2}) \subseteq \mathsf{ExpTime}$$

□

Exponential jumps make a difference

Corollary 12.10: P $\subsetneq$ ExpTime.

Proof:

- For every polynomial p, we have $p(n) \in o(2^n)$, so P $\subseteq$ DTime(2^n) $\subseteq$ ExpTime

  Note: of course, we also have $p^2 \in o(2^n)$, but this only shows that DTime(p) $\subsetneq$ ExpTime holds for specific polynomials, rather than P $\subsetneq$ ExpTime for the union over all polynomials.

- For proper inclusion, note $(2^n)^2 = 2^{2n} \in o(2^{n^2})$, so DTime$(2^n)$ $\subsetneq$ DTime(2^{n^2})
- In summary:

$$\mathsf{P} \subseteq \mathsf{DTime}(2^n) \subsetneq \mathsf{DTime}(2^{n^2}) \subseteq \mathsf{ExpTime}$$

□

Note: Simliar results hold for any exponential time gap, e.g., ExpTime $\subsetneq$ 2ExpTime.

Tighter Bounds

We have shown our Time Hierarchy Theorem using the fact that 1-tape DTMs can simulate k-tape DTMs with quadratic overhead.

Better results are known:

Theorem 12.11 (Hennie and Stearns, 1966): For any f with $f(n) \geq n$, we have $\mathsf{DTime}_*(f) \subseteq \mathsf{DTime}_2(f \cdot \log f)$.

(without proof; see, e.g., Hopcroft & Ullman, p. 292ff for details)

Tighter Bounds

We have shown our Time Hierarchy Theorem using the fact that 1-tape DTMs can simulate k-tape DTMs with quadratic overhead.

Better results are known:

Theorem 12.11 (Hennie and Stearns, 1966): For any f with $f(n) \geq n$, we have $\mathsf{DTime}_*(f) \subseteq \mathsf{DTime}_2(f \cdot \log f)$.

(without proof; see, e.g., Hopcroft & Ullman, p. 292ff for details)

Our first proof of the Time Hierarchy Theorem can use this 2-tape encoding to get the following result:

Time Hierarchy Theorem 12.12: If $f, g : \mathbb{N} \to \mathbb{N}$ are such that f is time-constructible, and $g \cdot \log g \in o(f)$, then

$$\mathsf{DTime}_*(g) \subsetneq \mathsf{DTime}_*(f)$$

This improvement was discovered soon after the first Time Hierarchy Theorem was found by Hartmanis and Stearns (1965).

Polynomial Time revisited

The stronger version of the Time Hierarchy Theorem can even separate different degrees of polynomials:

Corollary 12.13: For all $d \geq 1$, $\mathsf{DTime}_*(n^d) \subsetneq \mathsf{DTime}_*(n^{d+1})$.

Polynomial Time revisited

The stronger version of the Time Hierarchy Theorem can even separate different degrees of polynomials:

Corollary 12.13: For all $d \geq 1$, $\text{DTime}_*(n^d) \subsetneq \text{DTime}_*(n^{d+1})$.

Proof: Polynomial functions are time-constructible and we have:

$$n^d \in O(n^d \cdot \log n^d) = O(n^d \cdot \log n) = o(n^{d+1})$$

Hence the Time Hierarchy Theorem applies. □

Polynomial Time revisited

The stronger version of the Time Hierarchy Theorem can even separate different degrees of polynomials:

Corollary 12.13: For all $d \geq 1$, $\mathsf{DTime}_*(n^d) \subsetneq \mathsf{DTime}_*(n^{d+1})$.

Proof: Polynomial functions are time-constructible and we have:

$$n^d \in O(n^d \cdot \log n^d) = O(n^d \cdot \log n) = o(n^{d+1})$$

Hence the Time Hierarchy Theorem applies. □

One can view this as an argument against Cobham's Thesis ("P = practically tractable") since it shows that P has problems that require arbitrarily high degrees of polynomials, and are therefore most likely not practically tractable.

Nondeterminism

The results so far are for deterministic TMs only. One can find analogous results for NTMs . . .

- as expected, following our intuition that more time enables more computations
- surprisingly, since our earlier proof that simply flips the answer for diagonalisation does not work for NTMs

Nondeterminism

The results so far are for deterministic TMs only. One can find analogous results for NTMs . . .

- as expected, following our intuition that more time enables more computations
- surprisingly, since our earlier proof that simply flips the answer for diagonalisation does not work for NTMs

Nondeterministic Time Hierarchy Theorem (Cook, 1972) 12.14: If $f, g : \mathbb{N} \to \mathbb{N}$ are such that f is time-constructible, and $g(n+1) \in o(f(n))$, then

$$\mathsf{NTime}_*(g) \subsetneq \mathsf{NTime}_*(f)$$

(without proof; see, e.g., Arora & Barak, Section 3.2 for a sketch)

One can therefore get similar separation results as for deterministic time, e.g., NP $\subsetneq$ NExpTime.

Summary and Outlook

The time hierarchy theorems tell us that more time leads to more power:

$$\begin{tikzcd}
\mathrm{L} & \subseteq & \mathrm{NL} & \subseteq & \mathrm{P} \arrow[rrrrrr, dash, bend right, "\neq"'] & \subseteq & \mathrm{NP} \arrow[rrrrrr, dash, bend left, "\neq"] & \subseteq & \mathrm{PSpace} & \subseteq & \mathrm{ExpTime} & \subseteq & \mathrm{NExpTime} & \subseteq & \mathrm{ExpSpace}
\end{tikzcd}$$

Summary and Outlook

The time hierarchy theorems tell us that more time leads to more power:

$$\begin{tikzcd}
\mathsf{L} \arrow[r, phantom, "\subseteq"] & \mathsf{NL} \arrow[r, phantom, "\subseteq"] & \mathsf{P} \arrow[r, phantom, "\subseteq"] \arrow[rrr, dash, bend right, "\neq"'] & \mathsf{NP} \arrow[r, phantom, "\subseteq"] \arrow[rrr, dash, bend left, "\neq"] & \mathsf{PSpace} \arrow[r, phantom, "\subseteq"] & \mathsf{ExpTime} \arrow[r, phantom, "\subseteq"] & \mathsf{NExpTime} \arrow[r, phantom, "\subseteq"] & \mathsf{ExpSpace}
\end{tikzcd}$$

However, they don't help us in comparing different resources and machine types (P vs. NP, or PSpace vs. ExpTime)

Summary and Outlook

The time hierarchy theorems tell us that more time leads to more power:

$$\begin{tikzcd}
\mathrm{L} \arrow[r, phantom, "\subseteq"] & \mathrm{NL} \arrow[r, phantom, "\subseteq"] & \mathrm{P} \arrow[r, phantom, "\subseteq"] \arrow[rrr, dash, bend right, "\neq"'] & \mathrm{NP} \arrow[r, phantom, "\subseteq"] \arrow[rrr, dash, bend left, "\neq"] & \mathrm{PSpace} \arrow[r, phantom, "\subseteq"] & \mathrm{ExpTime} \arrow[r, phantom, "\subseteq"] & \mathrm{NExpTime} \arrow[r, phantom, "\subseteq"] & \mathrm{ExpSpace}
\end{tikzcd}$$

However, they don’t help us in comparing different resources and machine types (P vs. NP, or PSpace vs. ExpTime)

The diagram shows that in sequences such as P ⊆ NP ⊆ PSpace ⊆ ExpTime, one of the inclusions must be proper – but we don’t know which (expectation: all!)

Summary and Outlook

The time hierarchy theorems tell us that more time leads to more power:

$$\begin{tikzcd}
\mathsf{L} \arrow[r, phantom, "\subseteq"] & \mathsf{NL} \arrow[r, phantom, "\subseteq"] & \mathsf{P} \arrow[r, phantom, "\subseteq"] \arrow[rrr, dash, bend right, "\neq"'] & \mathsf{NP} \arrow[r, phantom, "\subseteq"] \arrow[rrr, dash, bend left, "\neq"] & \mathsf{PSpace} \arrow[r, phantom, "\subseteq"] & \mathsf{ExpTime} \arrow[r, phantom, "\subseteq"] & \mathsf{NExpTime} \arrow[r, phantom, "\subseteq"] & \mathsf{ExpSpace}
\end{tikzcd}$$

However, they don't help us in comparing different resources and machine types (P vs. NP, or PSpace vs. ExpTime)

The diagram shows that in sequences such as P ⊆ NP ⊆ PSpace ⊆ ExpTime, one of the inclusions must be proper – but we don't know which (expectation: all!)

What's next?

- The space hierarchy theorem
- Do we need time and space constructibility? What could possibly go wrong . . . ?
- The limits of diagonalisation, proved by diagonalisation