Time/Space Hierarchy, Polynomial-time Hierarchy

Intractability

- Recall
$$P \subseteq NP \subseteq PSPACE = NSPACE.$$
- Yet we have not proved any intractable problem.
  - A problem is intractable if it cannot be solved in polynomial time.
- The most difficult problem appears to be $TQBF \in PSPACE$.
- But we do not know if $P \stackrel{?}{=} PSPACE$.

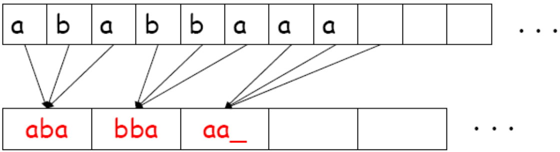
Linear Speedup

Theorem 1

Suppose TM M decides language L in time $f(n)$. Then for any $\epsilon > 0$, there exists TM M' that decides L in time $\epsilon \cdot f(n) + n + 2$.

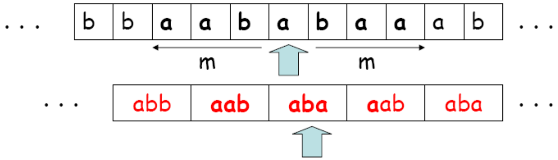
Proof Idea:

- compress input onto fresh tape:



Linear Speedup (cont'd)

- simulate M, m steps at a time



  – 4 (L,R,R,L) steps to read relevant symbols, "remember" in state

  – 2 (L,R or R,L) to make M's changes

- accounting:
  - part 1 (copying): $n + 2$ steps
  - part 2 (simulation): $6(f(n)/m)$
  - set $m = 6/\epsilon$
  - total: $\epsilon \cdot f(n) + n + 2$

Space Construcibility

Definition 2

$f : \mathbb{N} \to \mathbb{N}$ with $f(n)$ at least $O(\lg n)$ is called space constructible if the function that maps $\mathtt{1}^n$ to the binary representation of $f(n)$ is computable in space $O(f(n))$.

- That is, f is space constructible if there is an $O(f(n))$ space TM that always halts with the binary representation of $f(n)$ on input $\mathtt{1}^n$.
  - As usual, the $O(f(n))$ space TM has two tapes when $f(n) \in o(n)$.

Example 3

$\lg n$ is space constructible.

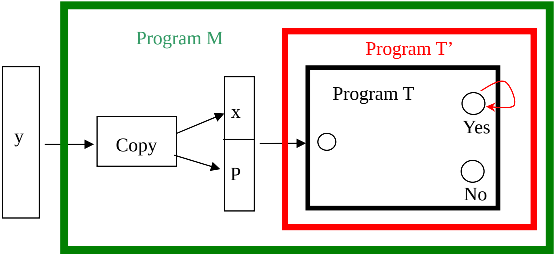
Proof.

On input $\mathtt{1}^n$, the TM counts the number of $\mathtt{1}$'s in binary representation on its work tape. $\lg n$ is the number of bits in the binary representation of n. The TM then computes $\lg n$ in binary representation. □

Space Constructibility

- Intuitively, a space $O(f(n))$ TM should be more powerful than a space $O(g(n))$ TM when $g(n) \in o(f(n))$.
- We would like to prove it by diagonalization.
- However, the difference may be very hard to compute.
  - Thus diagonalization fails.
- Space constructibility allows us to avoid the problem.

Recall the Diagonalization method for proving the halting problem



- Halt: T enters "Yes" $\Rightarrow$ Not Halt
- Not Halt: T enters "No" $\Rightarrow$ Halt

Space Hierarchy Theorem

Theorem 4

For any space constructible function $f : \mathbb{N} \to \mathbb{N}$, there is a language A decidable in $O(f(n))$ space but not in $o(f(n))$ space.

Proof.

Consider language $L = \{\langle M \rangle \mathtt{10}^* \mid M \text{ rejects } \langle M \rangle \mathtt{10}^* \text{ using } \leq f(n) \text{ space }\}$.

Consider $D =$ "On input w:

1. Compute $f(|w|)$ by space constructibility and mark off this much tape. If D ever attempts to use more space, reject.
2. If w is not of the form $\langle M \rangle \mathtt{10}^*$ for some TM M, reject.
3. Simulate M on w. If the simulation takes more than $2^{f(n)}$ M-steps, reject.
4. If M accepts, reject; if M rejects, accept."

□

Proof (cont'd).

In Step 3, D simulates M in D's tape alphabet. The simulation hence introduces a constant factor of overhead (independent of $|w|$). That is, if M runs in $g(n)$ space, D runs in $dg(n)$ space for some constant d. Clearly, D is an $O(f(n))$ space TM. We next argue that L cannot be decided in $o(f(n))$.

Suppose a TM M' decides L in space $g(n)$ for some $g(n) \in o(f(n))$. Since $g(n) \in o(f(n))$, there is an n_0 that $dg(n) < f(n)$ for every $n \geq n_0$. Consider $\langle M' \rangle \texttt{1}\texttt{0}^{n_0}$. Since $dg(n_0) < f(n_0)$, M' accepts $\langle M' \rangle \texttt{1}\texttt{0}^{n_0}$ if and only if M' rejects $\langle M' \rangle \texttt{1}\texttt{0}^{n_0}$. $\square$

Space Hierarchy Theorem

Corollary 5

Let $f_1, f_2 : \mathbb{N} \to \mathbb{N}$ with $f_1(n) \in o(f_2(n))$ and f_2 space constructible. $SPACE(f_1(n)) \subsetneq SPACE(f_2(n))$.

- We can show n^c is space constructible for any $c \in \mathbb{Q}^{\geq 0}$.
- Observe that for any $\epsilon_1, \epsilon_2 \in \mathbb{R}^{\geq 0}$ with $\epsilon_1 < \epsilon_2$, there are $c_1, c_2 \in \mathbb{Q}^{\geq 0}$ that $0 \leq \epsilon_1 < c_1 < c_2 < \epsilon_2$. Therefore

Corollary 6

For any $\epsilon_1, \epsilon_2 \in \mathbb{R}$ with $0 \leq \epsilon_1 < \epsilon_2$, $SPACE(n^{\epsilon_1}) \subsetneq SPACE(n^{\epsilon_2})$.

Corollary 7

$NL \subsetneq PSPACE$.

Proof.

By Savitch's theorem, $NL \subseteq SPACE(\lg^2 n)$. By space hierarchy theorem, $SPACE(\lg^2 n) \subsetneq SPACE(n)$. □

- Recall that $TQBF$ is $PSPACE$-complete. Hence $TQBF \notin NL$.

Corollary 8

$PSPACE \subsetneq EXPSPACE = \cup_k SPACE(2^{n^k})$.

- So far, we know

$$NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq EXPSPACE.$$

Time Constructibility

Definition 9

$t : \mathbb{N} \to \mathbb{N}$ with $t(n)$ at least $O(n \lg n)$ is called time constructible if the function that maps $\mathtt{1}^n$ to the binary representation of $t(n)$ is computable in time $O(t(n))$.

- That is, $t(n)$ is time constructible if there is an $O(t(n))$ time TM that always halts with the binary representation of $t(n)$ on input $\mathtt{1}^n$.

Example 10

$n\sqrt{n}$ is time constructible.

Proof.

On input $\mathtt{1}^n$, a TM counts the number of $\mathtt{1}$'s in binary representation. This takes time $O(n \lg n)$. $\lfloor n\sqrt{n} \rfloor$ in binary representatino can be computed in $O(n \lg n)$ time since the input is now of length $O(\lg n)$. $\square$

Time Hierarchy Theorem

Theorem 11

For any time constructible function $t : \mathbb{N} \to \mathbb{N}$*, there is a language* A *decidable in* $O(t(n))$ *time but not in* $o(t(n)/\lg t(n))$ *time.*

Proof.

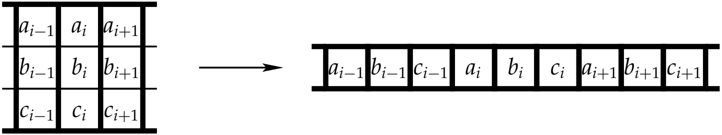
Consider

D = "On input w:

1. Compute $t(|w|)$ by time constructibility and store $\lceil t(n)/\lg t(n) \rceil$ in a binary counter. If this counter ever reaches 0, reject.
2. If w is not of the form $\langle M \rangle \mathtt{1}\mathtt{0}^*$ for some TM M, reject.
3. Simulate M on w and decrement the binary counter at each M-step.
4. If M accepts, reject; if M rejects; accept."

D simulates M with 3 tracks. Track 1 mimics M's tape; track 2 contains the current M state and the transition function of M; and track 3 contains the binary counter. Whenever M moves its tape head, D shifts the content on track 2 and 3 close to M's tape head. Since the length of the content on track 2 is independent of $|w|$, D's simulation needs a constant factor d time overhead.

Time Hierarchy Theorem



Proof.

The binary counter on track 3 need be decremented on every M-step. The length of the binary counter is $\lg(t(n)/\lg t(n)) \in O(\lg t(n))$. Hence decrementing the counter needs $\lg t(n)$ time overhead. D simulates M by at most $\lceil t(n)/\lg t(n) \rceil$ M-steps. Counting time overhead, D runs in time $O(t(n))$.

Suppose a TM M decides $A = L(D)$ in time $g(n)$ with $g(n) \in o(t(n)/\lg t(n))$. Not counting the time for updating the binary counter, D simulates M in time $dg(n)$. Since $g(n) \in o(t(n)/\lg t(n))$, there is an n_0 that $dg(n) \leq t(n)/\lg t(n)$ for all $n \geq n_0$. Consider the input $\langle M \rangle \mathtt{1}\mathtt{0}^{n_0}$. The initial binary counter is no less than $t(n_0)/\lg t(n_0) \geq dg(n_0)$. Thus D can simulate M on $\langle M \rangle \mathtt{1}\mathtt{0}^{n_0}$ with $g(n_0)$ M-steps. But D accepts $\langle M \rangle \mathtt{1}\mathtt{0}^{n_0}$ if and only if M rejects $\langle M \rangle \mathtt{1}\mathtt{0}^{n_0}$. $L(M) \neq L(D)$. $\square$

Applications of Time Hierarchy Theorem

Corollary 12

For $t_1, t_2 : \mathbb{N} \to \mathbb{N}$ *with* $t_1(n) \in o(t_2(n)/\lg t_2(n))$ *and* t_2 *time constructible.* $TIME(t_1(n)) \subsetneq TIME(t_2(n))$.

Corollary 13

For any $\epsilon_1, \epsilon_2 \in \mathbb{R}$ *with* $0 \leq \epsilon_1 < \epsilon_2$, $TIME(n^{\epsilon_1}) \subsetneq TIME(n^{\epsilon_2})$.

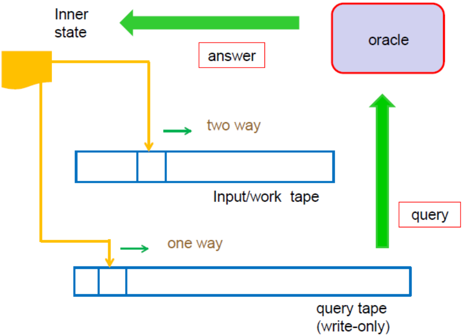
Corollary 14

$P \subsetneq EXPTIME$.

Oracle Turing Machines

Definition 15

An oracle for a language A answers whether $w \in A$ for any string w. An oracle Turing machine M^A is a Turing machine that can query an oracle A. When M^A write a string w on a special oracle tape, it is informed whether $w \in A$ in a single step.



Oracle Computations

- Let M be an oracle Turing machine (OTM)
- Let x be any string in Σ^*
- Let B be an oracle (which is now a language).
  1. M starts with input x.
  2. Whenever M writes a query word y on its query tape and enters a query state q_{query}, y is automatically sent to oracle B.
  3. The oracle B returns its answer (YES/NO) by changing M's inner state from q_{query} to either q_{yes} or q_{no}, depending on whether $y \in B$ or $y \notin B$, respectively.
  4. M resumes its computation, starting with q_{yes} or q_{no}.

Definition 16

$L(M^B) = \{x \in \Sigma^* \mid M \text{ accepts } x \text{ with oracle } B\}$.

Oracle Turing Machines

Definition 17

For two languages A and B, we say that A is Turing reducible to B (written as $A \leq_T B$) if there is an OTM M such that

1. $A = L(M^B)$; that is, for every input x, $x \in A \Leftrightarrow M^B$ accepts x via making queries to the oracle B

Definition 18

Language A is polynomial-time Turing reducible to language B (written as $A \leq_T^p B$ if there is an OTM M such that

1. $A = L(M^B)$; that is, for every input x, $x \in A \Leftrightarrow M^B$ accepts x via making queries to the oracle B
2. M runs in polynomial time.

Polynomial-time Oracle Turing Machines

Definition 19

$P^A = \{L : L \text{ is decided by a polynomial time OTM with oracle } A\}$
$NP^A = \{L : L \text{ is decided by a polynomial time ONTM with oracle } A\}$

Example 20

$NP \subseteq P^{SAT}$ and $coNP \subseteq P^{SAT}$.

Proof.

For any $A \in NP$, use the polynomial reduction of A to SAT. $\square$

Oracle Turing Machines

- Two Boolean formulae ϕ and ψ over $x_1, \ldots, x_l$ are equivalent if they have the same value on any assignments to $x_1, \ldots, x_l$.
- A formula is minimal if it is not equivalent to a smaller formula.
- Consider

$$NONMINFORMULA = \{\langle\phi\rangle : \phi \text{ is not a minimal Boolean formula}\}.$$

Example 21

$NONMINFORMULA \in NP^{SAT}$.

Proof.

"On input $\langle\phi\rangle$:

1. Nondeterministically select a smaller formula ψ.
2. Ask $\langle\phi \ XOR \ \psi\rangle \in SAT$.
3. If yes, accept; otherwise, reject."

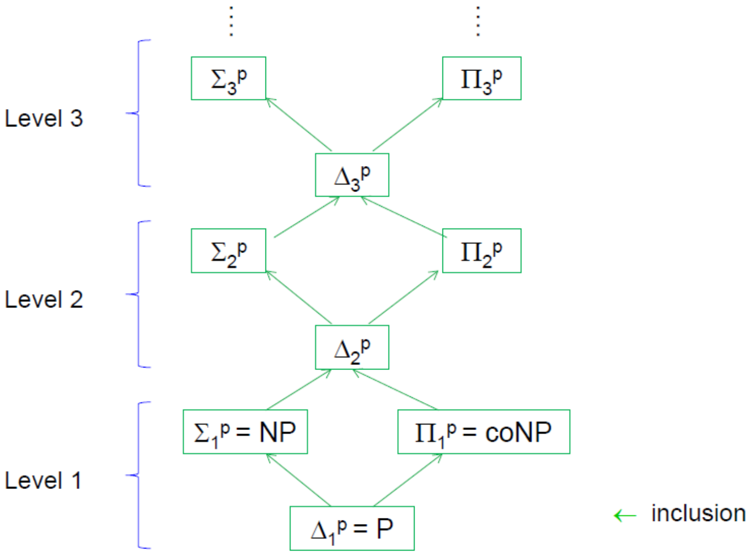
$\square$

Polynomial-time Hierarchy

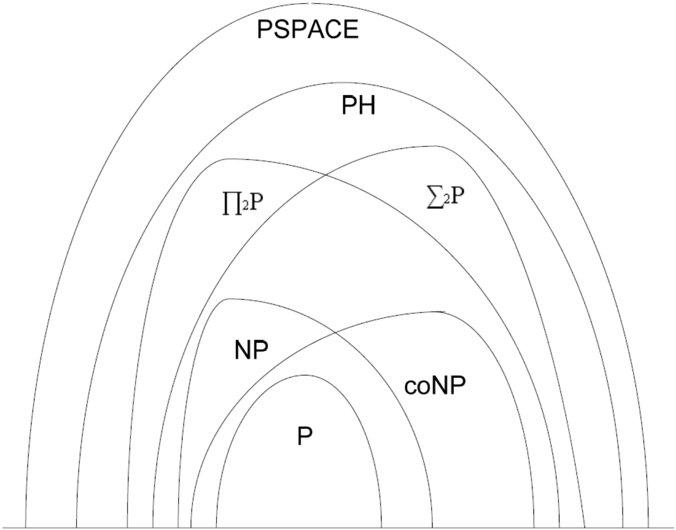
Meyer and Stockmeyer (1972, 1973) introduced a notion of the polynomial-time hierarchy over NP.
The polynomial hierarchy consists of the following complexity classes: for every index $k \geq 1$,

1. $\Delta_1^P = P$
2. $\Sigma_1^P = NP, \;\; \Pi_1^P = co\text{-}NP$
3. $\Delta_{k+1}^P = P^{\Sigma_k^P}$
4. $\Sigma_{k+1}^P = NP^{\Sigma_k^P}, \;\; \Pi_{k+1}^P = co\text{-}\Sigma_{k+1}^P$

Polynomial-time Hierarchy



Polynomial-time Hierarchy



Polynomial-time Hierarchy

We define the complexity class PH as follows:

$$PH = \bigcup_{k \geq 1} (\Sigma_k^P \cup \Pi_k^P)$$

- $NP \subseteq PH \subseteq PSPACE$
- If $P = NP$, then $P = PH$.
- $P^{PH} = NP^{PH} = PH$.

Another Characterization of Polynomial-time Hierarchy

We have already seen, that deciding whether a formula is satisfiable

- $\exists x_1 \cdots x_n (x_1 \vee \bar{x_2} \vee x_8) \wedge \cdots \wedge (\bar{x_6} \vee x_3)$
  - only existential quantifier – NP-complete
- $\exists x_1 \forall x_2 \exists x_3 ... (x_1 \vee \bar{x_2} \vee x_8) \wedge \cdots \wedge (\bar{x_6} \vee x_3)$
  - existential & universal quantifiers – PSPACE-complete

Definition 22

Consider language classes reducible to deciding the satisfiability of

$$\Sigma_i SAT : \exists x_1 \forall x_2 \exists x_3 ... R(x_1, x_2, x_3 ...)$$

$$\Pi_i SAT : \forall x_1 \exists x_2 \forall x_3 ... R(x_1, x_2, x_3 ...)$$

with i alternating quantifiers and $R(...)$ is a polynomial-time predicate.

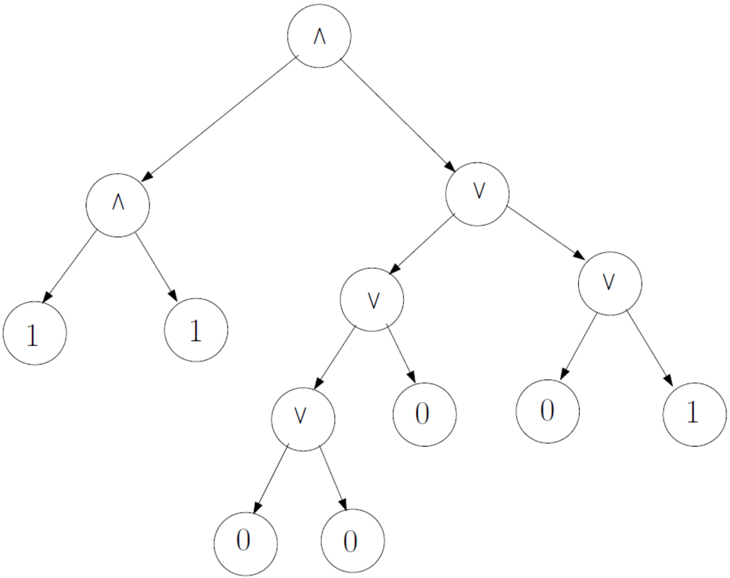
$\Sigma_i SAT$ and $\Pi_i SAT$ above define exactly the i-level of the polynomial-time hierarchy using polynomial-time oracle TMs.

Alternating Turing Machines

- An alternating Turing machine (ATM) $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ is a Turing machine with a non-deterministic transition function $\delta : Q \times \Gamma \to 2^{Q\times\Gamma\times\{L,R\}}$ whose set of states, in addition to accepting/rejecting states, is partitioned into existential ($\exists$ or $\vee$) and universal ($\forall$ or $\wedge$) states.
- A configuration C of an ATM M can reach acceptance if either of the following is true:
  - C is existential and some branch can reach acceptance.
  - C is universal and all branches can reach acceptance.

  M accepts a word w if the start configuration on w is accepting.

Alternating Turing Machines



Alternating Polynomial-time Hierarchy

Definition 23

Consider language classes

- $A\Sigma_i^p$: the language accepted by polynomial time ATM using at most i alternations with the initial state an $\exists$-state,
- $A\Pi_i^p$: the language accepted by polynomial time ATM using at most i alternations with the initial state an $\forall$-state,

It turns out that $A\Sigma_i$ and $A\Pi_i$ above again define exactly the i-level of the polynomial-time hierarchy using polynomial-time oracle TMs.

More on Alternating Complexity Classes

We define

- APTime $= \bigcup_{d \geq 1} ATime(n^d)$
- AExpTime $= \bigcup_{d \geq 1} ATime(2^{n^d})$
- ALogSpace $= \bigcup_{d \geq 1} ASpace(\log n)$
- APSpace $= \bigcup_{d \geq 1} ASpace(n^d)$
- AExpSpace $= \bigcup_{d \geq 1} ASpace(2^{n^d})$

Theorem 24

$$
\begin{array}{ccccccccc}
\mathsf{L} & \subseteq & \mathsf{PTime} & \subseteq & \mathsf{PSpace} & \subseteq & \mathsf{ExpTime} & \subseteq & \mathsf{ExpSpace} \\
 & & \parallel & & \parallel & & \parallel & & \parallel \\
 & & \mathsf{ALogSpace} & \subseteq & \mathsf{APTime} & \subseteq & \mathsf{APSpace} & \subseteq & \mathsf{AExpTime}
\end{array}
$$

Limits of the Diagonalization Method

- We have seen many applications of the diagonalization methd.
  - Particularly, the proofs of space and time hierarchy theorems.
- Can we use the diagonalization method to show $P \stackrel{?}{=} NP$?
  - Say, to construct an NTM that accepts $\langle M\rangle 10^n$ if and only if the polynomial time TM M rejects $\langle M\rangle 10^n$.
- We give a strong evdience to explain why it may not work.
- The diagonalization method basically simulates a TM M by a TM D. If M and D are given an oracle A, D^A can simulate M^A as well.
- Hence if the diagonalization method can prove $P \stackrel{?}{=} NP$, it can also prove $P^A \stackrel{?}{=} NP^A$ for any oracle A.
- We will now give two oracles A and B such that $P^A \neq NP^A$ and $P^B = NP^B$.
- The diagonalization method does not suffice to prove $P \stackrel{?}{=} NP$.

Limits of the Diagonalization Method

Theorem 25

There are oracles A and B such that $P^A \neq NP^A$ and $P^B = NP^B$.

Proof.

Let B be $TQBF$. Then $NP^{TQBF} \subseteq NPSPACE \subseteq PSPACE \subseteq P^{TQBF}$.
For any oracle C, define

$$L_C = \{1^n : \exists x \in C\ [\ |x| = n\ ]\}.$$

Clearly, $L_C \in NP^C$ for any C. We construct a language A such that $L_A \notin P^A$.

$\exists A, NP^A \not\subseteq P^A$

Proof.

- Let $M_1^?, M_2^?, \ldots$ be an enumeration of oracle DTMs that run in polynomial time. Assume for simplicity that $M_i^?$ has running time n^i. Since oracle machines query their oracle as a black box, can plug in any oracle.
- We will build an oracle A so that none of these machines can decide L_A.
- Inductive construction. We start with nothing, and at each stage we declare a finite set of strings to be in the language of A or out of it.
- Goal: At stage i, make sure that $L(M_i^A)$ and L_A disagree on some string.

$\square$

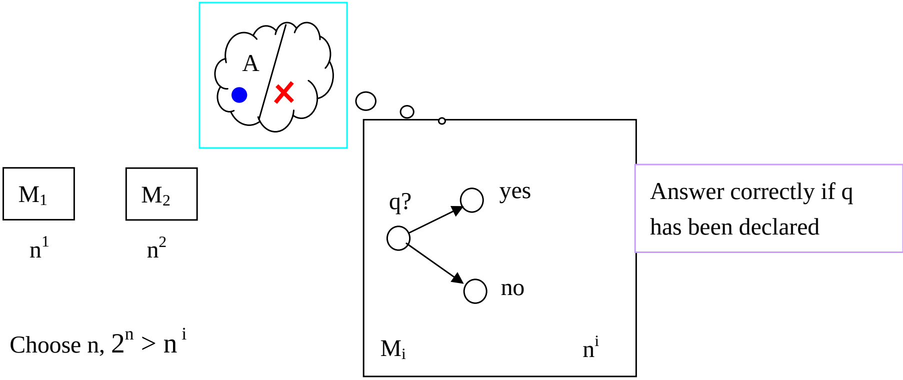
Proof.

- **Stage** i
  - Let M_i^A have running time n^i . Choose n larger than any string declared for A, such that $2^n > n^i$.
  - We are going to run M_i^A on 1^n. When M_i^A queries A with q, we
    - Answer correctly if q has been declared,
    - and answer NO otherwise.
  - If M_i^A accepts 1^n, we declare all strings of length n to be NO-strings. Then A has no YES-string of length n, and $1^n \notin L_A$. .
  - If M_i^A rejects 1^n, we find a string of length n that M_i^A did not query. This exists, since $2^n > n^i$. Declare this string to be YES.
- Finally, declare all undeclared strings of length up to n arbitrarily.

Hence M_i accepts $\mathtt{1}^n$ if and only if $\mathtt{1}^n \notin L_A$. M_i does not decide L_A.

□

$\exists A, NP^A \not\subseteq P^A$



- M_i accept 1^n, declare all strings of length n to be NO-strings
- M_i rejects 1^n, we find a string w length n not queried by M_i and adds w to A

Diagonalization - Cantor's Argument

Recall Cantor's Argument for showing 2^S of a countable set $S = \{s_1, s_2, ...\}$ is not countable

Proof.

Suppose for a contradiction that 2^S is countable.

- Then the sets in 2^S can be enumerated in a list $S_1, S_2, S_3, ... \subseteq S$
- Let us write this list as boolean matrix with rows representing the sets $S_1, S_2, S_3, ...$ columns representing a (countably infinite) enumeration of S, and boolean entries encoding the $\in$ relationship.
- For a contradiction, define a set S_d by diagonalization to differ from all other S_i in the enumeration:

	s_1	s_2	s_3	$\dots$
S_1	$\times$			$\dots$
S_2			$\times$	$\dots$
S_3	$\times$	$\times$		$\dots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
S_d		$\times$	$\times$	$\dots$

Diagonalization - The Halting Problem

Proof.

Suppose for a contradiction that Halting is decidable.

- Then set of all Turing machines can be enumerated in a list $M_1, M_2, M_3, \ldots$
- We are interested in their halting on inputs of the form $\langle M_i \rangle$ for some TM M
- We can write it as a boolean matrix with rows representing the TMs $M_1, M_2, M_3, \ldots$ columns representing an enumeration of strings $\langle M_i \rangle$, and boolean entries encoding if TM halts.
- Using a decider for the halting problem, we can define a TM M_d by diagonalization to differ from all other M_i in the enumeration:

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\ldots$
M_1	$\times$			$\ldots$
M_2			$\times$	$\ldots$
M_3	$\times$	$\times$		$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$
M_d		$\times$	$\times$	$\ldots$

Generalizing Diagonalization

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

Definition 26

Given a class K of Turing machines (e.g., 2-tape deterministic TMs), R is a resource (e.g., time or space) defined for all machines in K if $R_M(w) \in N \cup \{\infty\}$ for all $M \in K$ and all words w.
Then, any function $f : N \to N$ gives rises to a class of languages:

$$R(f) =$$

$$\{\mathrm{L} \mid \text{ there is } M \in K \text{ with } L(M) = L \text{ and } R_M(w) \le f(|w|) \text{ for all } w \in \Sigma^*\}$$

Generalizing Diagonalization

Consider resources R_1 and R_2 for two classes of Turing machines K_1 and K_2, and two functions $f_1, f_2 : N \to N$.

Definition 27

We say that $R_1(f_1)$ allows diagonalization over $R_2(f_2)$ if there exists a Turing machine $D \in K_1$ such that

- $L(D) \in R_1(f_1)$, and
- for each $M \in K_2$ that is R_2-bounded by f_2, there exists a w such that $\langle M, w \rangle \in L(D)$ if and only if $\langle M, w \rangle \notin L(M)$.

Example 28

Let R_1 and R_2 be *DSPACE*. $f_1 = O(f(n))$ and $f_2 = O(g(n))$ with $g(n) = o(f(n))$ in the space hierarchy theorem. $L(D) = \{\langle M \rangle \texttt{10}^* \mid M$ rejects $\langle M \rangle \texttt{10}^*$ using $\le f(n)$ space $\}$.

Generalizing Diagonalization

Theorem 29

If $R_1(f_1)$ allows diagonalization over $R_2(f_2)$, then $R_1(f_1) \setminus R_2(f_2) \neq \emptyset$.

Proof.

Let D be as in the Definition. We show $L(D) \notin R_2(f_2)$.

1. Suppose for a contradiction that there $M \in K_2$ that is R_2-bounded by f_2 with $L(D) = L(M)$.
2. We obtain a contradiction, since, by Definition, there is a word w such that

$$\langle M, w \rangle \in L(D) = L(M) \Leftrightarrow \langle M, w \rangle \notin L(M)$$

□