(continued) We want to show that $\forall L \in \Sigma_2$ there is an equivalent formula $\exists \forall \ldots$

- let M be a TM $[\Sigma_1]$ ... that is TM $[L_u]$ accepting L
- formula: $\exists \delta$ (check that δ is consistent with input α, with oracle L_u, and with itself)

$\exists \delta$: computation of M including all oracle queries & answers

consistent with input α: decidable

with oracle L_u: for positive answers: $\exists \gamma_1 - \gamma_k$ $\psi(\gamma_1, \text{question } 1)$ ⋮ $\psi(\gamma_k, \text{question } k)$ — Σ_1-formula for L_u

itself: decidable

together: $\exists \delta \left(\exists \gamma \; \forall \varepsilon \; \varphi(\alpha, \delta, \gamma, \varepsilon) \right)$

- γ: code of $\gamma_1 - \gamma_k$
- ε: code of $\varepsilon_1 - \varepsilon_\ell$

for negative answers: $\forall \varepsilon_1 - \varepsilon_\ell \; \neg\psi(\varepsilon_1, \ldots) \; \& \; \ldots$

... but this is $\exists \langle \delta, \gamma \rangle \forall \varepsilon \; \varphi(\ldots)$ as we need.

COMPLEXITY

Df:

- For a multi-tape machine M, define run time $t_M(\alpha)$ for input α as # steps before computation $M(\alpha)$ halts.
  - $t_M : \{0,1\}^* \to \mathbb{N}^*$ (generally Σ^*), $\mathbb{N}^* = \mathbb{N} \cup \{\infty\}$ (sorry for confusing notation: this $*$ is not iteration, but: for divergent computations)

- Time complexity of machine M: $T_M : \mathbb{N} \to \mathbb{N}^*$ s.t. $T_M(n) = \max_{\alpha \in \Sigma^n} t_M(\alpha)$.
  - M always halts $\Leftrightarrow$ $T_M(n)$ finite for all n.

Let's use $\alpha \in \Sigma^*, |\alpha| \le n$ instead: this is slightly non-standard, but much more convenient. In particular, $T_M(n)$ will be non-decreasing.

Df: Asymptotic notation: for functions $f, g : \mathbb{N} \to \mathbb{R}$ define:

1. $f \in O(g) \equiv \exists c > 0 \; \forall^* n \; f(n) \le c \cdot g(n)$ ← asymptotic upper bound
   - $\forall^* n$: for all but finitely many exceptions
2. $f \in \Omega(g) \equiv \exists c > 0 \; \forall^* n \; f(n) \ge c \cdot g(n)$ ← asymp. lower bound
3. $f \in \Theta(g) \equiv f \in O(g) \; \& \; f \in \Omega(g)$ ← both at once
   ... that is, $\Theta(g) = O(g) \cap \Omega(g)$
4. $f \in o(g) \equiv \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$ ← strict upper bound
5. $f \in \omega(g) \equiv g \in o(f)$ ← strict lower bound

Examples: $f : n \mapsto 5n^3 - 7n^2 + 18$

$\in O(n^3), O(n^4), O(2^n)$ $\quad \in o(n^4)$

$\in \Omega(n^3), \Omega(n^2), \Omega(1)$ $\quad \in \omega(n^2)$

$\in \Theta(n^3)$ "drop lower-order terms & multiplicative constant"

$O(1)$ = "bounded by constant"

Dependence of complexity on # tapes

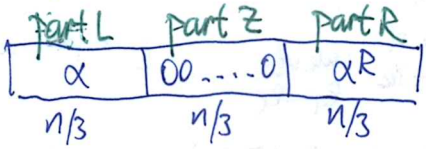
Df: $L_{PAL} = \{ \alpha \alpha^R \mid \alpha \in \{0,1\}^* \}$ "even palindromes"
(α^R = reversed)

- trivial 2-tape TM deciding L_{PAL} in $\Theta(n)$ time.
- but all machines we found with 1 tape run in $\Theta(n^2)$ time?

Thm: Every 1-tape machine deciding L_{PAL} runs in time $\Omega(n^2)$.

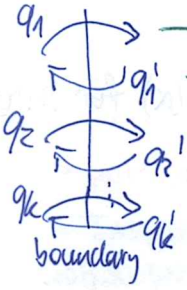
Proof: Assume there is M deciding L_{PAL} s.t. $T_M(n) \in o(n^2)$. ← that is: $\forall \varepsilon > 0 \; \forall^* n : T_M(n) < \varepsilon n^2$.

- Consider inputs of type: for $6 \mid n$ (n will be chosen later)



← answer is YES for every such string

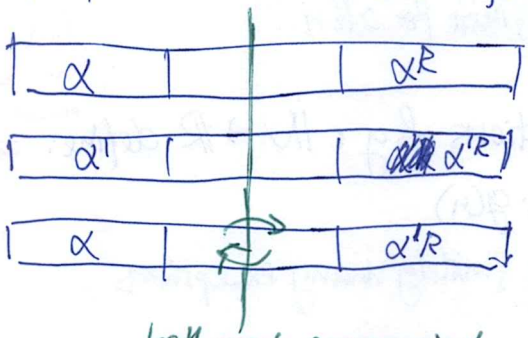
- Consider boundary between 2 zeros in part Z & how computation of M crosses it:



— first, we cross from left to right in some state q_1
— then we return in state q_1'
— WLOG computation ends in L
(we can modify M to "rewind" without affecting $o(n^2)$)

↳ crossing sequence $(q_1, q_1'), \ldots, (q_k, q_k')$

- If two inputs with α, α' have the same C.S. for the same boundary:



answers YES
answers YES
also answers YES, although the answer should be NO ↯

- Similarly if they have same C.S. for different boundaries.
(part Z can have odd length, but that implies NO anyway)

- Let's use P.H.P. (Pigeon-hole principle) to show that such α, α' exist:

c.s. of length $k = |Q|^{2k}$

c.s. of length at most $k \leq c \cdot |Q|^{2k}$ for some constant c
(via sum of geom. series)

If # c.s. < # inputs, then ∃ two inputs with the same C.S.
↳ so we want $c \cdot |Q|^{2k} < 2^{n/3} \ldots 2^{\overbrace{\log c + 2k \log|Q|}^{\leq 3k \log|Q|}} < 2^{\frac{n}{3}} \ldots k < \frac{n}{9k \log|Q|}$

- P.H.P. once again (for every machine we can find boundary with a small # crossings):
  - we have $n/3$ boundaries
  - Σ of lengths of their C.S. $\leq T_M(n) < \varepsilon n^2$
  - $\Rightarrow$ $\exists$ boundary with at most $\frac{\varepsilon n^2}{n/3} = 3\varepsilon n$ crossings
- now set ε such that we have:

$$\text{min. \# crossings} \leq 3\varepsilon n < \frac{n}{9 \log(|Q|)}$$

such ε exists & inequality satisfied for n large enough

- so there are 2 inputs with the same C.S. $\Rightarrow$ mixing produces contradiction.

Thm: For every multi-tape TM M there is 1-tape TM M' s.t. $L(M) = L(M')$ & $T_{M'}(n) \in O(T_M(n)^2)$.

Proof: Analyze reduction from previous lectures. [hint: at most $T_M(n)$ tape cells used on each tape]

Thm: For every multi-tape TM M there is 2-tape TM M' s.t. $L(M) = L(M')$ & $T_{M'}(n) \in O(T_M(n) \cdot \log T_M(n))$.

(proof omitted)

Time complexity classes

- $\mathrm{DTIME}(f) := \{ L \subseteq \{0,1\}^* \mid \exists M$ multi-tape TM deciding L in time $O(f) \}$ ($T_M \in O(f)$)
  - $f: \mathbb{N} \to \mathbb{R}$
  - sometimes just $\mathrm{TIME}(f)$ ["D" is for "deterministic"]
  - every finite Σ s.t. $|\Sigma| \geq 2$ would work
  - this will be the default
  - makes things easier, but for TMs it's not necessary because of time compression thm.
- We will require f to have these properties:
  1. non-decreasing
  2. $\forall n \; f(n) \geq n$
  3. time-constructible $\equiv \exists$ TM M_f which for input 1^n produces output $1^{f(n)}$ in time $O(f(n))$

("proper time complexity function")

- $P := \bigcup_{i \geq 1} \mathrm{DTIME}(n^i)$ ← polynomial-time decidable languages

Why we like P:
- Corresponds (roughly) to "efficiently solvable"
- independent of model of computation (RAM gives the same P as TM)
- polynomials are the smallest set of functions containing constants & identity and closed under addition, multiplication and composition.

Examples:
- reachability in graphs
- evaluation of Boolean formulas

Classes of functions

- $\mathrm{DTIMEF}(f) := \{g : \{0,1\}^* \to \{0,1\}^* \mid \exists \text{ TM } M \text{ computing } g \text{ in time } O(f)\}$
- $PF := \bigcup_{i \geq 1} \mathrm{DTIME}(n^i)$



if $|f(n)| \in poly(n)$: $L_g := \{\langle x, i \rangle \mid i\text{-th bit of } g(x) \text{ is } 1\}$ (encoded in binary)

($O(n^k)$ for some fixed k)

$L_g \in P \Leftrightarrow g \in PF$.

So studying only languages in P is WLOG.

Consider these problems:

(as usually: suitably encoded; path vs. walk ← can repeat, doesn't repeat vertices)

① HAMILTON PATH
Input: undirected graph G, vertices u, v
Question: $\exists$ path in G with endpoints u, v containing all vertices (exactly) once.

② 3-COLORING
Input: undirected graph G
Q: $\exists f: V(G) \to \{1,2,3\}$ s.t. $\forall \{u,v\} \in E(G): f(u) \neq f(v)$
(coloring of G with 3 colors)

③ INDEPENDENT SET
Input: undirected graph G, $k \in \mathbb{N}$
Q: $\exists A \subseteq V(G): |A| \geq k \;\&\; \forall u,v \in A: \{u,v\} \notin E(G)$

④ CLIQUE
Input: G, $k \in \mathbb{N}$
Q: $\exists A \subseteq V(G): |A| \geq k \;\&\; \forall u,v \in A: (u = v \vee \{u,v\} \in E(G))$

⑤ 0,1-EQUATIONS a.k.a. 2OE
Input: matrix A, vector b with 0/1 entries
Q: $\exists x \in \{0,1\}^n: Ax = b$ (evaluated in integers, not $\mathbb{Z}_2$)
↳ WLOG $b = 1$

⑥ SAT (Boolean satisfiability): Input: Boolean formula $\varphi(x_1 \ldots x_m)$ in CNF
Q: $\exists x_1 \ldots x_m \in \{0,1\}$ s.t. $\varphi(\vec{x})$ is true.

For all these: we are looking for something we can easily recognize (poly-time), but we don't know how to find it in poly time.

Reductions will again prove themselves useful:

Def: For languages K, L: $K \leq_m^P L \equiv \exists f \in PF$ s.t. $\forall x \in \Sigma^*: x \in K \Leftrightarrow f(x) \in L$.
(polynomial-time many-to-one reduction)

$\varphi \equiv (x_1 \vee \neg x_2 \vee x_3) \wedge (\ldots) \wedge (\ldots)$
(clause; literals: either x_i or $\neg x_i$)

(restriction to CNF is WLOG, see later)

(we cannot use thm. from Logic about equivalent formulas in CNF, because it blows up size exponentially)

Properties of $\leq_m^P$:

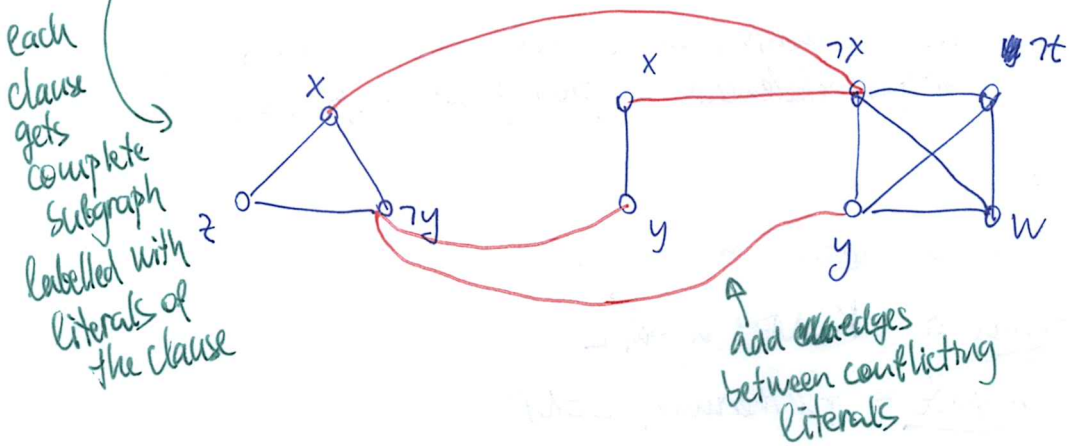
- reflexive & transitive (quasi-order)
- ∃ incomparable languages (exercise)
- $K \leq_m^P L$ and $L \in P \Rightarrow K \in P$ [composition of 2 algorithms running in poly. time is again poly-time]
  That is, P is closed under reductions.

size of input for ALG2 is at most time complexity of ALG1

Example: SAT $\leq_m^P$ INDEPSET

$$(x \vee y \vee z) \wedge (x \vee y \vee \dots) \wedge (\neg x \vee y \vee \neg t \vee w)$$

given a formula → produce a graph, $k :=$ # clauses → from each subgraph we must select exactly 1 vertex



- ∃ satisfying assignment: for each clause, pick 1 satisfied literal, put its vertex to the indep. set → got l.s. of size k
- ∃ indep. set of size k: each vertex in l.s. selects a variable which will be set to 0/1 to satisfy the corresponding clause,
  red edges guarantee that we won't set var to both 0 and 1
  - remaining variables set arbitrarily
  → got satisfying assignment
- the reduction runs in poly. time

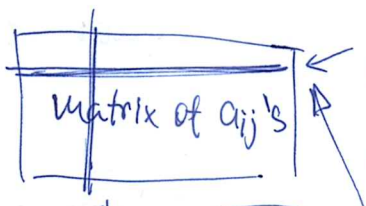
Example: INDEPSET $\leq_m^P$ SAT ... given G, k, construct formula φ s.t. φ is satisfiable ⇔ G has ind. set of size k

Variables: $x_1 \dots x_n$: vertex v_i selected to ind. set

a_{ij} for $1 \leq i \leq k$, $1 \leq j \leq n$: vertex j is i-th in the ind. set ← order on vertices of the set

Clauses: $\forall \{v_i, v_j\} \in E(G)$: $\neg x_i \vee \neg x_j$

$\forall i,j \; a_{ij} \Rightarrow x_j$ ← order describes the set (we allow unordered elements of set, which doesn't break anything)



$a_{ij} \Rightarrow \neg a_{ij'}$ no number used multiple times

$a_{ij} \Rightarrow \neg a_{i'j}$ no vertex used twice or more

$a_{i1} \vee \dots \vee a_{in}$ each number used at least once

implication $x \Rightarrow y$ is a clause $\neg x \vee y$

So we get CNF

Exercises: ① IndepSet $\leq^P_m$ Clique

② 3-Coloring $\leq^P_m$ SAT

Formalization of "search problems":

Df: Class of languages NP:

$L \in NP \equiv \exists V \in P$ (verifier)

$\forall \alpha \in \Sigma^*: \alpha \in L \Leftrightarrow \left(\exists \beta \in \Sigma^*: |\beta| \in poly(|\alpha|) \;\&\; V(\alpha,\beta) \right)$

∃ certificate of polynomial size which is accepted by the verifier

👁 $P \subseteq NP$ ... verifier does all the work & ignores β

👁 resembles proofs in logic: true statements have a proof, which is easy to verify
for false statements, no proof passes verification

Big question: Is $P = NP$?

↑ 1MB price by Clay Mathematical Institute (waits for you ☺)

Df: Language L is NP-hard $\equiv \forall K \in NP: K \leq^P_m L$

L is NP-complete $\equiv$ furthermore, $L \in NP$

Lemma: Let $K \leq^P_m L$. Then:

① if $L \in NP$, then $K \in NP$ (just compose verifier with reduction) ← NP is closed under reductions

② if K is NP-hard, then L is NP-hard. ($\forall M \in NP: M \leq K \leq L \Rightarrow M \leq L$)

②' if K is NP-complete & $L \in NP$, then L is NP-complete.

} makes it easy to prove NP-completeness once we have one NP-comp. problem

Lemma: If $L \in P$ is NP-complete, then $P = NP$.

Proof: $P \subseteq NP$ is trivial, will prove $NP \subseteq P$:

Let $K \in NP$. Then $K \leq L$, which implies $K \in P$.

Thm (Cook-Levin): SAT is NP-complete.

↳ will be proven later

MORE REDUCTIONS

3D MATCHING Input: sets B (boys), G (girls), C (cats)

$T \subseteq B \times G \times C$ (triples)

Output: $\exists T' \subseteq T$ s.t. each element of $B \cup G \cup C$ is contained in exactly 1 triple in T'

(generalizes bipartite matching, which is in P)