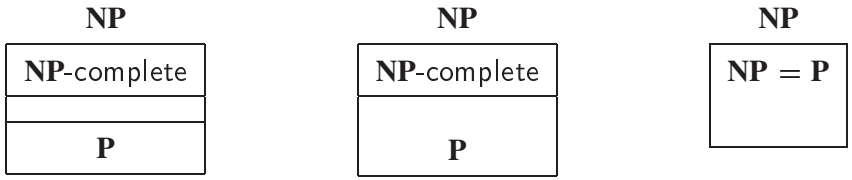
On P vs NP



- GRAPH ISOMORPHISM is not known to be in **P** or **NP**-complete.
- Are there problems in **NP** that are neither in **P** nor **NP**-complete? Yes, at least if $\mathbf{P} \neq \mathbf{NP}$.

Theorem. If $\mathbf{P} \neq \mathbf{NP}$, then there is a language in **NP** which is neither in **P** nor **NP**-complete.

© 2003 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

Isomorphism and Density

- All known **NP**-complete languages are *polynomially isomorphic*.
- Languages K, L are polynomially isomorphic if there is a function $h : \Sigma^* \mapsto \Sigma^*$ such that
  (i) h is bijection;
  (ii) for each $x \in \Sigma^*$, $x \in K$ iff $h(x) \in L$;
  (iii) h and h^{-1} are polynomial-time computable.
- A reduction is not necessarily a polynomial time isomorphism (not a bijection) but with a padding function an isomorphism is obtained.

© 2003 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

Isomorphism—cont'd

- A function $pad : (\Sigma^*)^2 \mapsto \Sigma^*$ is a padding function for L if
  (i) pad is computable in logarithmic space;
  (ii) for each $x, y \in \Sigma^*$, $pad(x,y) \in L$ iff $x \in L$;
  (iii) for each $x, y \in \Sigma^*$, $|pad(x,y)| > |x| + |y|$;
  (iv) there is a logarithmic-space algorithm which given $pad(x,y)$ recovers y.

Example. A padding function for SAT:

Given x (a conjunction of m clauses with n variables), $pad(x,y)$ is all clauses of x together with $m + |y|$ new clauses and $|y| + 1$ more variables where the m new clauses are copies of the clause x_{n+1} and where the $m+i$th new clause is either $\neg x_{n+i+1}$ or x_{n+i+1} depending on whether the ith symbol in y is 0 or 1.

© 2003 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

Isomorphism—cont'd

- If R is a reduction from K to L and pad is a padding function for L, $pad(R(x),x)$ is a length-increasing one-to-one reduction. Furthermore, there is a logarithmic-space algorithm for the inverse of $pad(R(x),x)$.
- If there is a reduction from K to L and a reduction from L to K and the reductions are length-increasing, one-to-one and logarithmic-space invertible, then K, L are polynomially isomorphic.
- Corollary: The following **NP**-languages are polynomial isomorphic: SAT, HAMILTON PATH, CLIQUE, MAX CUT, TRIPARTITE MATCHING, KNAPSACK, ...

© 2003 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

Density

Density of a language L:

$$\mathrm{dens}_L(n) = |\{x \in L : |x| \leq n\}|$$

Proposition. If $K, L \subseteq \Sigma^*$ are polynomially isomorphic, then dens_K and dens_L are polynomially related. $(\mathrm{dens}_L(n) \leq \mathrm{dens}_K(p(n)))$

- A *sparse* language: polynomially bounded density function
- A *dense* language: superpolynomial density function

© 2003 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

- Examples:
  Any unary language $(\subseteq \{0\}^*)$ is sparse $(\mathrm{dens}_L(n) \leq n)$.
  All **NP**-complete language seen this far are dense.

Theorem. If a unary language $U \subseteq \{0\}^*$ is **NP**-complete, then $\mathbf{P} = \mathbf{NP}$.

Theorem. If a sparse language is **NP**-complete, then $\mathbf{P} = \mathbf{NP}$.

© 2003 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

ORACLES

- The idea: study complexity in a setting where a part of the computation comes "for free".
- Can be used for exploring conjectures (like $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$) in an alternative setting.
- Can isolate orthogonal (independent) sources of complexity.

© 2003 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

Oracles—cont'd

- An oracle Turing machine $M^?$:
  query string, query state $q?$, answer states q_{YES}, q_{NO} (depending on whether for y on the query string $y \in A$ for the oracle set A)
  Note that *a query* is performed *in one step*!
  Computation of $M^?$ with oracle A on input x: $M^A(x)$.
- For any time complexity class $\mathcal{C}$ and oracle A there is a corresponding complexity class $\mathcal{C}^A$.

Theorem. There is an oracle A for which $\mathbf{P}^A = \mathbf{NP}^A$.

Proof: Let A be **PSPACE**-complete. Then

$$\mathbf{PSPACE} \subseteq \mathbf{P}^A \subseteq \mathbf{NP}^A \subseteq \mathbf{NPSPACE} \subseteq \mathbf{PSPACE}$$

© 2003 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

Oracles—cont'd

Theorem. There is an oracle B for which $\mathbf{P}^B \neq \mathbf{NP}^B$.

Proof. B is constructed by "diagonalization".

Lessons to be learned w.r.t. $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$

- By the above $\mathbf{P} \neq \mathbf{NP}$ is possible; non-trivial conjecture.
- "Ordinary proof techniques" are not sufficient for establishing $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$.

  They are not affected by oracles.

© 2003 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

Monotone Circuits

- **Conjecture B.** **NP**-complete problems have no polynomial circuits, uniform or not.
- Lower bounds on the size of circuits for families of functions are hard to establish.
- Consider a weaker circuit model: monotone circuits (ones without NOT gates).
- Monotone circuits can only compute monotone functions: output cannot changed from **true** to **false** by changing input from **false** to **true**.

© 2003 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio

Monotone Circuits—cont'd

- Some **NP**-complete problems are monotonic, e.g., HAMILTON PATH and CLIQUE.
- How small can the monotone circuits coding these problems be?
- Let $\text{CLIQUE}_{n,k}$ be the Boolean function deciding whether a graph $G = (V,E)$ with n nodes has a clique of size k.

Theorem. (Razborov's Theorem): There is a constant c such that for large enough n all monotone circuits for $\text{CLIQUE}_{n,k}$ with $k = n^{1/4}$ have size at least $2^{cn^{1/8}}$.

- To show $\mathbf{P} \neq \mathbf{NP}$ it would be sufficient to establish that all monotone languages in **P** have polynomial monotone circuits.
- However, this does not hold (e.g., for MATCHING).

© 2003 Teknillinen korkeakoulu, Tietojenkäsittelyteorian laboratorio