Definition 1

$L \in NP$ if there exists a polynomial-time TM M (called the verifier for L) and polynomial $p : \mathbb{N} \to \mathbb{N}$ such that $\forall x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)} : M(x,u) = 1.$$

Note: TM M runs in polynomial time
$q(|x| + |u|) = (|x| + |x|^c)^d = O(|x|^{cd})$ for $q(n) = n^d$ and $p(n) = n^c$.

Note: Equivalent definition via Not-Deterministic TM's (cf. 2.1.2).

Examples: INDSET, TSP, SUBSUM, LP, 0-1 IP, GRAPHISO, COMPSITES, FACTORING, CONNECTIVITY

Definition 2 (alternative)

$L \in NP$ if $\exists$ poly-time non-deterministic TM $M(x) = 1 \Leftrightarrow x \in L$.

Definition 3

$EXP = \bigcup_{c \geq 0} DTIME(2^{n^c})$

Claim 1

$P \subseteq NP \subseteq EXP$

Proof $P \subseteq NP$: $L \in P \Rightarrow$ poly-time TM M decides L
$\Rightarrow M'(x, 0 \text{ or } 1) := M(|x|)$ is a verifier for L with $p(|x|) = |x|^0 = 1$

$NP \subseteq EXP$: $L \in NP \Rightarrow$ poly-time verifier $M(x, u)$ with $|u| = p(|x|)$
$\Rightarrow$ the following TM M' decides L in $O(2^{n^{p(|x|)}})$ time

---

Algorithm $M'(x)$

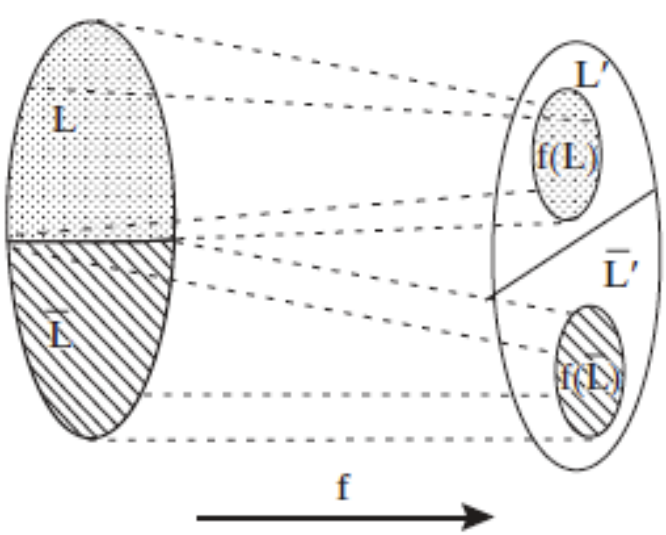
---

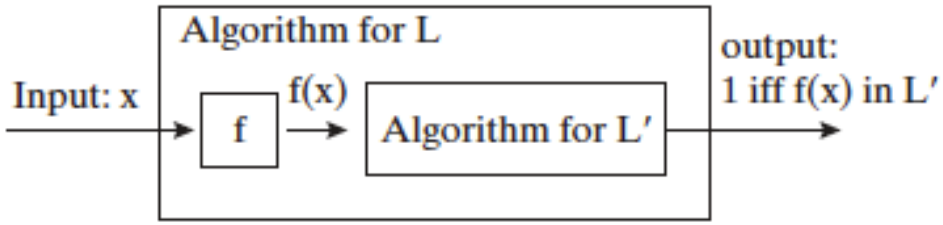
for each $u \in \{0,1\}^{p(|x|)}$ **do**
  if $M(x, u) = 1$ **then**
    return 1
return 0

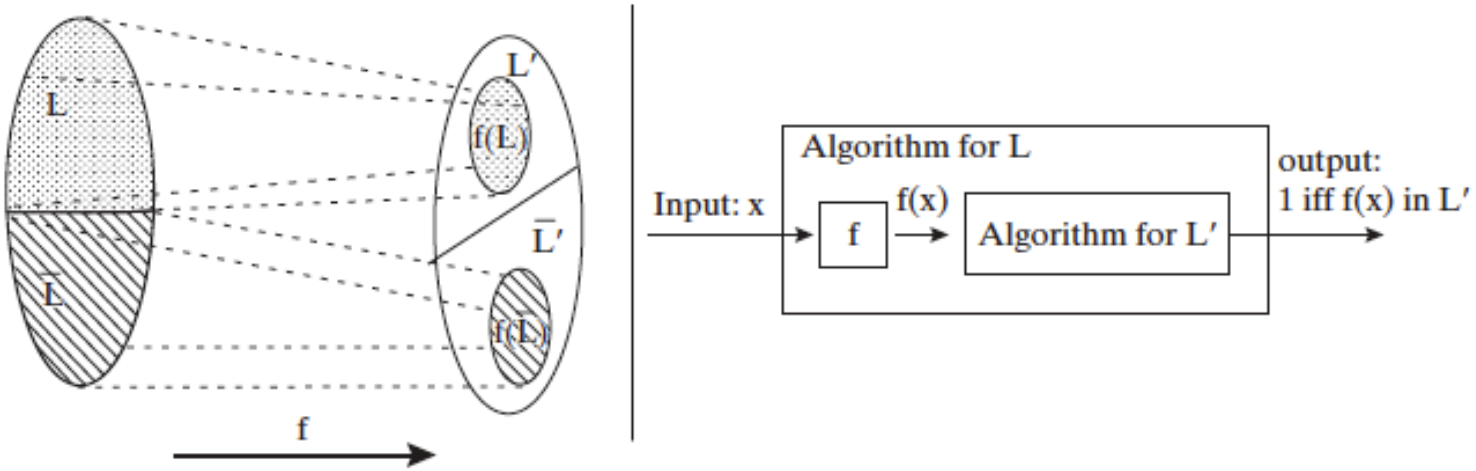
Definition 4 (Reductions)

L is poly-time Karp reducible to L' ($L \leq_P L'$) if there is poly-time computable function f s.t. $\forall x \in \{0,1\}^*$

$$x \in L \Leftrightarrow f(x) \in L'$$







- Why $L \leq_P L'$?
  - $L' \in P \Rightarrow L \in P$, i.e., if L' easy then L easy
  - $L \notin P \Rightarrow L' \notin P$, i.e., if L hard then L' hard
- Karp vs. Turing reductions: In $L \leq_P^T L'$ we are allowed to use a polynomial number of calls to L' (not just one).

Theorem 5 (Transitivity)

If $L \leq_P L'$ and $L' \leq_P L''$, then $L \leq_P L''$.

Proof:

- $x \to f_1(x)$ with $f_1(x)$ computable in $O(|x|^c)$ $(L \leq_P L')$
- $y \to f_2(y)$ with $f_2(y)$ computable in $O(|y|^d)$ $(L' \leq_P L'')$
- $x \to f_2(f_1(x))$ is $L \leq_P L''$, with $f_2(f_1(x))$ computable in $O((|x|^c)^d) = O(|x|^{cd})$.

since $x \in L \Leftrightarrow f_1(x) \in L' \Leftrightarrow f_2(f_1(x)) \in L''$. □

Definition 6 (NP-hardness)

L is NP-hard if $L' \leq_P L$ for every $L' \in NP$.

Definition 7 (NP-completeness)

L is NP-complete if

1. L is NP-hard, and
2. $L \in NP$.

Theorem 8

1. *If L is NP-hard and $L \in P$, then $P = NP$.*
2. *If L is NP-complete, then $L \in P \Leftrightarrow P = NP$.*

Proof:

1. $L' \in NP \overset{L \text{ is } NP-\text{hard}}{\Rightarrow} L' \leq_P L \overset{L \in P}{\Rightarrow} L' \in P$
2. If $P = NP \Rightarrow L \in P$. If L is NP-complete and $L \in P$ then $P = NP$ from (1).

□

Theorem 9

The following language is NP-complete:

$$TMSAT = \{\langle a, x, 1^n, 1^t\rangle : \exists u \in \{0,1\}^n \text{ s.t. } M_a(x,u) = 1 \text{ within } t \text{ steps}\}$$

Proof:

1. $TMSAT \in NP$ (easy)
2. $L \in NP \Rightarrow$ verifier $M(x,u)$ s.t.
   $x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|}$: $M(x,u) = 1$ in $q(|x| + p(|x|))$ steps
   $\Rightarrow x \to \langle \llcorner M \lrcorner, x, 1^{p(|x|)}, 1^{q(|x|+p(|x|))}\rangle$ gives $L' \leq_P TMSAT$.

□

...but too artificial, like rewriting the *NP*-completeness definition!

Definition 10 (CNF formula)

Given n boolean variables $u_1, u_2, \ldots, u_n$

- $u_1, \bar{u}_1$ are the literals for variable u_1.
- A clause is an OR of literals, e.g., $(u_1 \vee \bar{u}_3 \vee u_4)$.
- A CNF formula is an AND of clauses, e.g., $(u_3) \wedge (\bar{u}_1 \vee u_3 \vee \bar{u}_4) \wedge (u_2 \vee \bar{u}_3)$.
- CNF formula is satisfiable if there is truth assignment to vars that makes formula true.

Definition 11 (SAT)

Given a CNF formula with n vars and k clauses, is it satisfiable?

Theorem 12 (Cook-Levin)

SAT is NP-complete.

Proof:

1. $SAT \in NP$: easy
2. $\forall L \in NP : L \leq_P SAT$: Idea is like TMSAT, but for $L \in NP$ explicitly write the configuration of verifier $M(x, u)$ at every step as a big tableau of $O(q(|x| + |u|)) = O(q(|x| + p(|x|)))$ rows and $O(q(|x| + p(|x|)))$ columns
   $\Rightarrow$ encode certificate u bits as vars, and correctness conditions of transition from step i configuration to $i + 1$ configuration as clauses
   $\Rightarrow$ SAT formula $\phi_x(u)$ is satisfiable iff $\exists u$ to make $M(x, u)$ accept
   $\Rightarrow \phi_x(u) \in SAT$ iff $x \in L$

□

Levin-reductions

Reduction of Theorem 12 is Levin: One-to-one mapping between satisfying assignment for $\phi_x(u)$ and certificate for $x \in L$.

Proving decision problem L is *NP*-complete:

1. Prove that $L \in NP$.
2. Pick *NP*-complete problem L'. Show that $L' \leq_P L$.

Example: 3SAT

SAT with all clauses with 3 literals.

Theorem 13

3SAT is NP-complete.

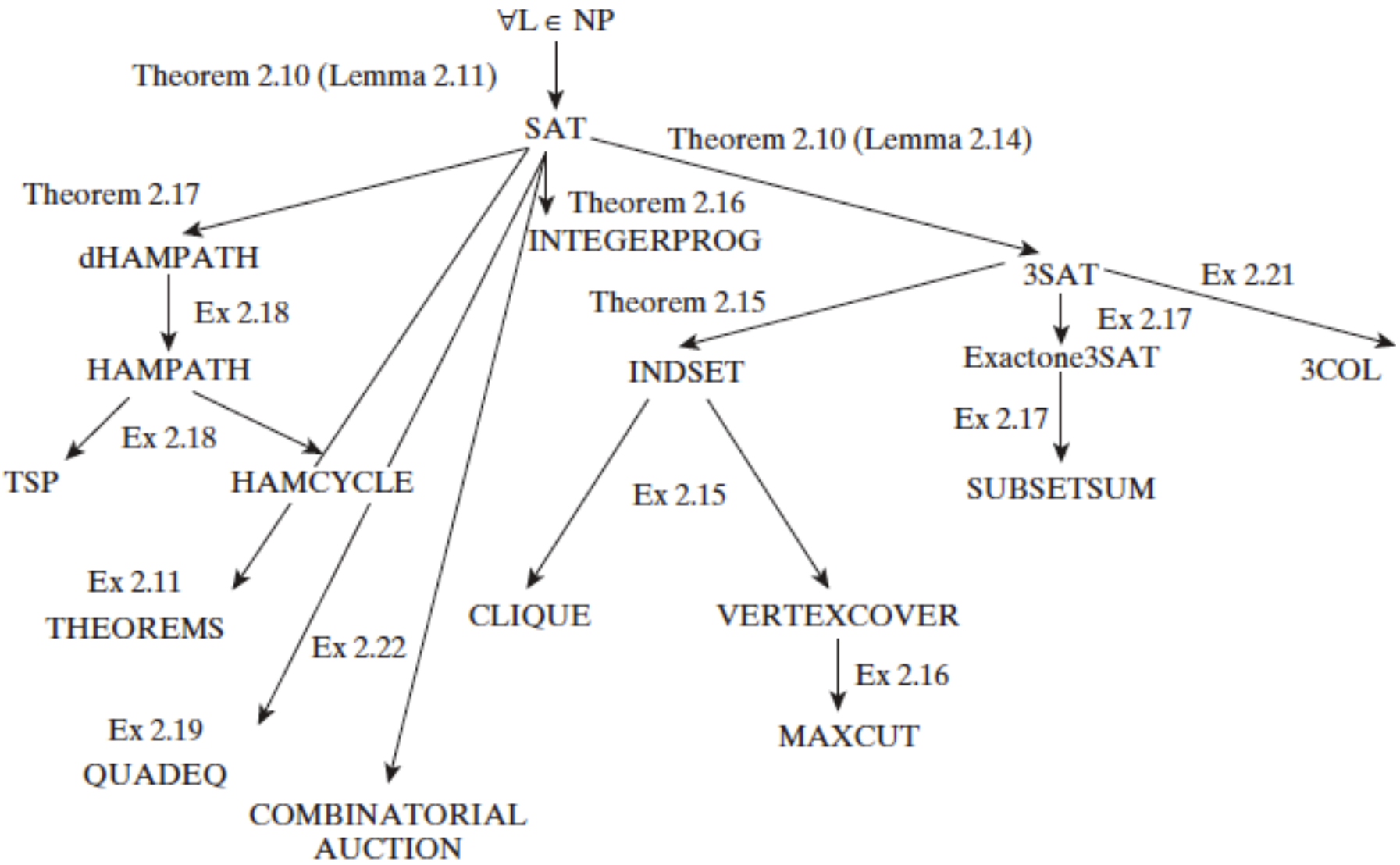
Proof:

1. $3SAT \in NP$: easy
2. $\forall L \in NP : L \leq_P SAT$: We show that $SAT \leq_P 3SAT$. Given a CNF formula $\phi(x)$ for SAT with n vars $x_1, x_2, \ldots, x_n$ and k clauses, we construct a 3SAT formula $\psi(y)$ s.t. $\phi(x) \in SAT \Leftrightarrow \psi(y) \in 3SAT$.
   - Keep all vars x
   - Let C be a clause of $\phi(x)$. If C has more than 3 literals break it into two clauses C', C'' using a new var z_c as follows:

$$(x_1 \vee \bar{x}_2 \vee x_4 \vee \bar{x}_6 \vee \bar{x}_7) \rightarrow \begin{cases} (x_1 \vee \bar{x}_2 \vee z_c) \\ (x_4 \vee \bar{x}_6 \vee \bar{x}_7 \vee \bar{z}_c) \end{cases}$$

   - If C fewer than 3: Repeat last literal

□


∀L ∈ NP
Theorem 2.10 (Lemma 2.11)
SAT
Theorem 2.10 (Lemma 2.14)
Theorem 2.17
Theorem 2.16
dHAMPATH
INTEGERPROG
3SAT
Ex 2.21
Ex 2.18
Theorem 2.15
Ex 2.17
HAMPATH
INDSET
Exactone3SAT
3COL
Ex 2.17
Ex 2.18
TSP
HAMCYCLE
Ex 2.15
SUBSETSUM
Ex 2.11
THEOREMS
CLIQUE
VERTEXCOVER
Ex 2.22
Ex 2.16
Ex 2.19
QUADEQ
MAXCUT
COMBINATORIAL
AUCTION

Theorem 14

If $P = NP$, then can also solve in poly-time the search version of SAT, i.e., compute a satisfying assignment.

Proof: Let A be a poly-time algorithm that decides SAT. Then the following algorithm B computes a satisfying assignment for CNF $\phi(x)$:

Algorithm $B(\phi(x))$

$V[1..n] =$ truth assignment for $x_1, x_2, \ldots, x_n$
if $A(\phi(0, x_2, \ldots, x_n)) = 1$ **then**
    $V[1] = 0$
    $V[2, \ldots, n] = B(\phi(0, x_2, \ldots, x_n)$
    return V
else if $A(\phi(1, x_2, \ldots, x_n)) = 1$ **then**
    $V[1] = 1$
    $V[2, \ldots, n] = B(\phi(1, x_2, \ldots, x_n)$
    return V
return No

□

Note: SAT is self-reducible

Theorem 15

If $P = NP$, then can also compute in poly-time a certificate of any $L \in NP$.

Proof:

The reduction $L \leq_P SAT$ is a Levin-reduction, i.e., if $x \in L$ then we can compute the certificate for $f(x) \in SAT$ in poly-time and from it the certificate for x.

□

Definition 16

$co - NP = \{L : \bar{L} \in NP\}$.

Example: $S\bar{A}T = \{\phi : \phi \text{ is unsatisfiable}\}$
Note: $L \in co - NP$ has a certifier for its "No" instances. Does it have one for its "Yes" instances...?

Definition 17

$L \in co - NP$ if there exists a polynomial-time TM M and polynomial $p : \mathbb{N} \to \mathbb{N}$ such that $\forall x \in \{0,1\}^*$,

$$x \in L \Leftrightarrow \forall u \in \{0,1\}^{p(|x|)} : M(x,u) = 1.$$

- Definitions 21 and 17 are equivalent (why?)
- Note that Definition 17 is exactly the same as our definition of NP *except* that $\forall u$ instead of $\exists u$.

Theorem 18

$TAUTOLOGY = \{\phi : \phi \text{ is a } tautology\}$ is $co - NP\text{-}complete$.

Proof:

- $TAUTOLOGY \in co - NP$ from Definition 17.
- $\forall L \in co - NP \rightarrow \bar{L} \in NP \rightarrow \bar{L} \leq_P SAT$ (Cook-Levin)
  $\rightarrow x \notin \bar{L} \Leftrightarrow \phi_x \notin SAT$
  $\rightarrow x \in L \Leftrightarrow \neg\phi_x \in TAUTOLOGY \rightarrow L \leq_P TAUTOLOGY$ $\square$

Theorem 19

$L \in co - NP\text{-}complete \Leftrightarrow \bar{L} \in NP\text{-}complete$.

Theorem 20

$P = NP \Rightarrow P = NP = co - NP$.

Definition 21

$NEXP = \bigcup_{c \geq 0} NTIME(2^{n^c})$

Theorem 22

$EXP \neq NEXP \Rightarrow P \neq NP$ (or $P = NP \Rightarrow EXP = NEXP$)

Proof: Use input size to cheat! (padding)
Obviously $EXP \subseteq NEXP$. Show $NEXP \subseteq EXP$.
$L \in NTIME(2^{n^c}) \Rightarrow L_{pad} = \{\langle x, 1^{2^{|x|^c}} \rangle : x \in L\}$. Poly-time NDTM:

Algorithm $M_{L_{pad}}(y)$

if $y \neq \langle z, 1^{2^{|z|^c}} \rangle$ for some z **then**
  return 0
return $M_L(z)$

$\Rightarrow L_{pad} \in NP \Rightarrow L_{pad} \in P \Rightarrow L \in EXP$ □

Chapter 2: *NP* and *NP*-completeness

- The philosophical importance of *NP* (read 2.7.1)
- *NP* and (short) mathematical proofs:

  $THEOREMS_{\mathcal{A}} = \{\langle \phi, 1^n \rangle : \phi \text{ has formal proof of } \leq n \text{ steps in system } \mathcal{A}\}$

- Is there anything between *P* and *NP*-complete? (factoring, graph isomorphism, Nash equilibrium, Ladner's theorem)
- Coping with *NP*-hardness (approximation algorithms, average-case complexity)
- Read chapter notes & history!