Computational Models — Lecture 13[1]

Handout Mode

Ronitt Rubinfeld and Iftach Haitner.

Tel Aviv University.

June 15/17, 2015

[1] Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University.

Talk Outline

- Reminder - poly-time reductions, NP-completeness and SAT
- SAT is NP-Complete
- SAT $\leq_P$ 3SAT
- Hamiltonian Path Is NP-Complete
- SUBSET-SUM Is NP-Complete

- Sipser, 7.4–7.5

Section 1

Reminder

Reminder – Poly-time Reductions

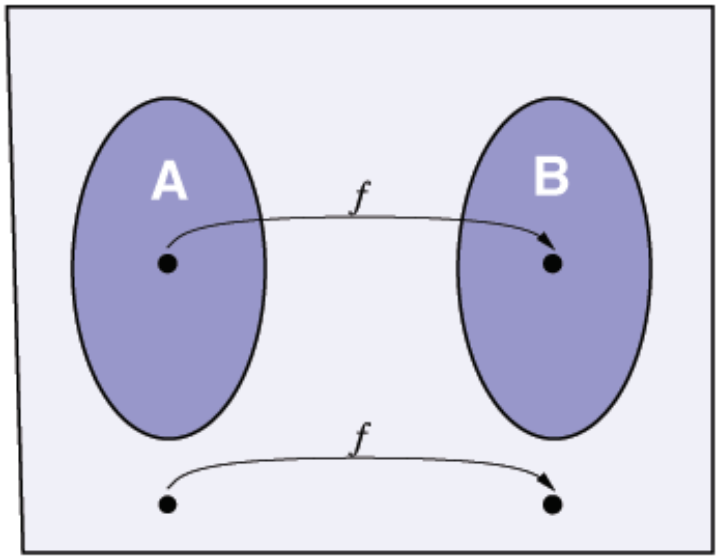
Definition 1

A language A is poly-time mapping reducible to B, denoted $A \leq_P B$, if exists poly-time computable function f such that

$$w \in A \iff f(w) \in B$$

for every $w \in \Sigma^*$.
The function f is called a polynomial-time reduction from A to B.

The mapping f efficiently converts questions about membership in A to membership in B.



Reminder – NP Completeness

Definition 2 (NP-complete)

A language B is NP-complete, if

- $B \in \mathcal{NP}$, and
- Every $A \in \mathcal{NP}$ is poly-time reducible to B

We let $\mathcal{NPC}$ denote the class of all NP-complete languages.

$A_{NP} = \{\langle M, x, 1^n\rangle : M \text{ is a TM} \wedge \exists c \in \Sigma^* \text{ s.t. } M(x, c) \text{ accepts within } n \text{ steps}\}$.

Theorem 3

$A_{NP} \in \mathcal{NPC}$.

Theorem 4

Assume that $B \in \mathcal{NP}$, $A \in \mathcal{NPC}$ *and* $A \leq_P B$, *then* $B \in \mathcal{NPC}$.

Boolean Formulas and SAT

A Boolean formula is an expression involving Boolean variables and operations.

$$\phi = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$$

Definition 5 (satisfiable formula)

A formula is satisfiable, if some assignment of 0s and 1s to the variables makes the formula evaluate to 1.

The formula $\phi = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$ is satisfiable by the assignment

$$x = 0$$
$$y = 1$$
$$z = 0$$

The language of satisfied formulas:

$$\text{SAT} = \{\langle\phi\rangle : \ \phi \text{ is a satisfiable Boolean formula}\}$$

Section 2

SAT is NP-Complete

SAT $\in \mathcal{NPC}$

Theorem 6 (Cook-Levin, early 70s)

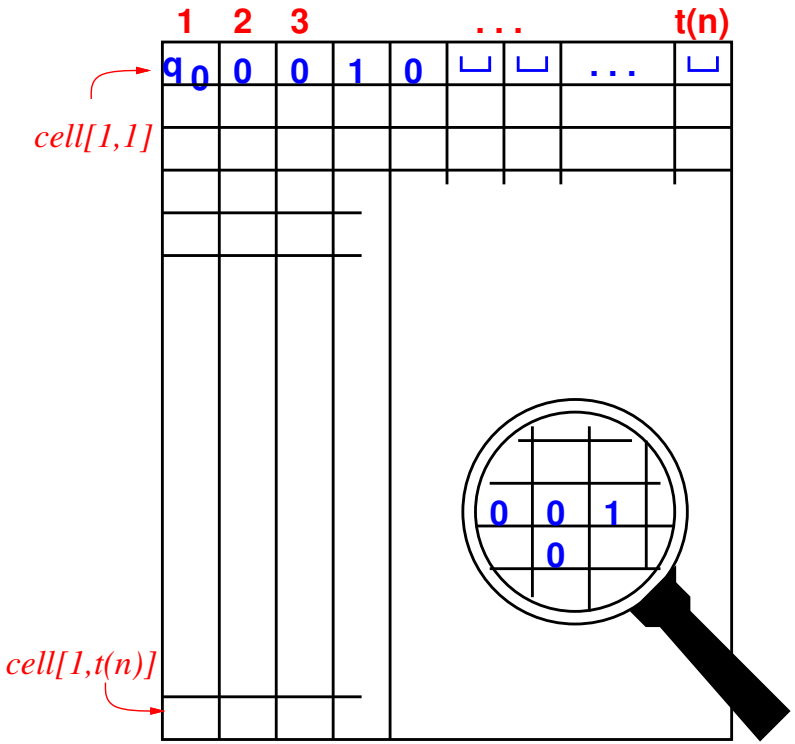
SAT $\in \mathcal{NPC}$.

Proof's idea:

- Suppose L $\in \mathcal{NP}$ and let $N = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ be an n^c-time NTM that accepts L.
- Given the string $w \in \{0,1\}^*$, construct in time $O(|w|^{2c})$ a formula $\phi_{N,w}$ such that: $\phi_{N,w} \in$ SAT iff N accepts w.
- Hence, the mapping $w \mapsto \phi_{N,w}$ is a poly-time reduction from L to SAT, establishing L $\leq_P$ SAT.
- In the following fix L, N and $w \in \{0,1\}^n$.
- We assume wlg. that $\delta(q_r, \cdot) = \emptyset$.

The Configuration-History Tableau

Consider the n^c-by-n^c Tableau that describes a computation history of N on input w.



- First row represents initial configuration of N on input w.
- i'th row represents the i-th configuration in a possible computation of N on input w.

The formula $\phi_{N,w}$

- Let $S = Q \cup \Gamma$ (the alphabet of the configuration history).
- $\phi_{N,w}$ uses Boolean variables $\{x_{i,j,s}\}_{i,j \in [n^c], s \in S}$.

$$\phi_{N,w} = \phi_{\text{Cell}(N)} \wedge \phi_{\text{Start}(w)} \wedge \phi_{\text{Move}(N)} \wedge \phi_{\text{Accept}(N)}$$

- Given an assignment z for $\phi_{N,w}$, let $T(z)$ be the $n^c \times n^c$ Tableau, defined by setting the j-th cell in i'th configuration to s, if $x_{i,j,s} = 1$ in z.

  ($T(z)$ is undefined, if $x_{i,j,s'} = x_{i,j,s} = 1$ for some $s \neq s' \in S$, or $x_{i,j,s'} = 0$ for all $s \in S$).

- $T(z)$ will represents a (possible) accepting execution of $N(w)$, iff z is an a satisfying assignment for $\phi_{N,w}$.

The formula $\phi_{\mathrm{Cell}(N)}$

$\phi_{\mathrm{Cell}(N)}$ guarantees that the variables encode legal configurations:

- Each cell (i,j) has at least one letter: $\bigvee_{s\in S} x_{i,j,s}$.
- No cell (i,j) has two or more letters $\bigwedge_{s\neq s'\in S} \overline{x_{i,j,s} \wedge x_{i,j,s'}}$.

Together:

$$\phi_{\mathrm{Cell}(N)} = \bigwedge_{i,j} \left[\left(\bigvee_{s\in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s\neq s'\in S} \overline{x_{i,j,s} \wedge x_{i,j,s'}} \right) \right]$$

Claim 7

If an assignment z satisfies $\phi_{\mathrm{Cell}(N)}$, then $T(z)$ is defined.

The formula $\phi_{\text{Start}(w)}$

$\phi_{\text{Start}(w)}$ guarantees that the first row encodes the initial configuration (i.e., $q_0 w$).

$$\phi_{\text{start}(w)} = x_{1,1,q_0} \wedge x_{1,2,w_1} \wedge x_{1,3,w_2} \wedge \ldots \wedge x_{1,n+1,w_n} \wedge x_{1,n+2,\sqcup} \wedge \ldots \wedge x_{1,n^c,\sqcup}$$

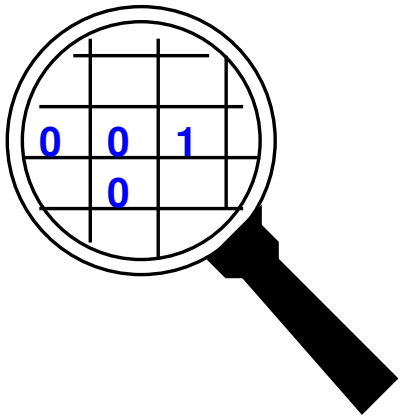
Claim 8

If z satisfies $\phi_{\text{Cell}(N)} \wedge \phi_{\text{Start}(w)}$, then the first line of $T(z)$ is $q_0 w \underbrace{\sqcup \ldots \sqcup}_{n^c - n - 1}$.

The formula $\phi_{\text{Move}(N)}$

$\phi_{\text{Move}(N)}$ is the "heart" of $\phi_{N,w}$. To construct it, we employ locality of computations.

- To verify the content of Tableau entry (i,j) (i.e., cell j in configuration i), it only requires to know M's table and the contents of the three Tableau entries: $(i-1,j-1),(i-1,j),(i-1,j+1)$.



- If head not in area, nothing changes.
- If head is in area, changes are local and are determined by δ.

$\phi_{\text{Move}(N)}$ – Rectangles

A rectangle is a 2×3 configuration sub-table.

Assume that
$\delta(q_1, a) = \{(q_1, b, \text{R})\}$ and $\delta(q_1, b) = \{(q_2, c, \text{L}), (q_2, a, \text{R})\}$.

- (some) Legal 2×3 rectangles:

a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

a	a	q_1
a	a	b

a	b	a
a	b	q_2

b	b	b
c	b	b

a	a	b
a	a	b

- (some) Illegal 2×3 rectangles:

a	b	a
a	a	a

a	q_1	b
q_1	a	a

b	q_1	b
q_2	b	q_2

$\phi_{\text{Move}(N)}$ – Characterizing legal rectangles

The formula “verifies" that all 2×3 rectangles in the Tableau are in the list C:

$\forall\ a, a', b, b', c \in \Gamma$, $q, q' \in Q$ and $* \in S$, add the following rectangles to C:

1.

a	b	c
$*$	b	$*$

2.

a	q	b
q’	a	b’

$(\text{L}, q', b') \in \delta(q, b)$

a	q	b
a	b’	q’

$(\text{R}, q', b') \in \delta(q, b)$

3.

q	$*$	$*$
$*$	$*$	$*$

$*$	$*$	q
$*$	$*$	$*$

- Some rectangles in C are clearly illegal.
- For rectangles on the left-most and right-most side of Tableau, we use slightly different first type rectangles.

$\phi_{\text{Move}(N)}$– formal definition

- For each entry $(i,j) \in [n^c] \times [n^c]$ and $c \in C$, let $\phi_{\text{Move},i,j,c}$ be the formula taking the value 1 iff the 2×3 table of cells in the Tableau whose upper-left corner is (i,j) is c.

  For instance, for entry $(1,1)$ and $c =$

a	q_1	b
q_2	a	c

  ,

  let $\phi_{\text{Move},1,1,c} = x_{1,1,a} \wedge x_{1,2,q_1} \wedge x_{1,3,b} \wedge x_{2,1,q_2} \wedge x_{2,2,a} \wedge x_{2,3,c}$

- Finally, let $\phi_{\text{Move}(N)} = \bigwedge_{(i,j)} \bigvee_{c \in C} \phi_{\text{Move},i,j,c}$.

Claim 9

If z satisfies $\phi_{\text{Cell}(N)} \wedge \phi_{\text{Start}(w)} \wedge \phi_{\text{Move}(N)}$, then $T(z)$ is a possible configuration history of $N(w)$.

Proof: By induction on the row index. Base case: z satisfies $\phi_{\text{Cell}(N)} \wedge \phi_{\text{Start}(w)}$.

Assume configuration defined in rows $1, \ldots, i$ is possible and head is in cell j. The configuration of rows $1, \ldots, i+1$ is also possible: Cells of indices not in $\{j-1, j, j+1\}$, by first type of rectangles in C. Other cells, by second type rectangles in C. Q: Why de we need the third type of cells?

The formula $\phi_{\text{Accept}(N)}$

$\phi_{\text{Accept}(N)}$ guarantees that some row encodes an accepting configuration (i.e., uq_av):

$$\phi_{\text{Accept}(N)} = \bigvee_{i,j} x_{i,j,q_a}$$

Claim 10

If z satisfies $\phi_{N,w} = \phi_{\text{Cell}(N)} \wedge \phi_{\text{Start}(w)} \wedge \phi_{\text{Move}(N)} \wedge \phi_{\text{Accept}(N)}$, then $T(z)$ is an accepting configuration history of $N(w)$.

Correctness of reduction

- The transformation $w \mapsto \phi_{N,w}$ is computable in time $O(n^{2c})$.
- An assignment satisfying $\phi_{N,w}$, corresponds to an accepting configuration history of $N(w)$.
- An accepting configuration history of $N(w)$ corresponds to an assignment satisfying $\phi_{N,w}$. (?)

  Therefore, N accepts w iff $\phi_{N,w} \in$ SAT. ♣.

- For complete details, consult Sipser chapter 7.4.

Section 3

3SAT Is NP-Complete

CSAT and 3SAT

- SAT $= \{\langle\phi\rangle : \ \phi$ is a satisfiable Boolean formula$\}$
- A Boolean formula is in conjunctive normal form (CNF) if it consists of clauses, connected with $\wedge$'s.

  For example: $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6})$
- CSAT $= \{\langle\phi\rangle : \ \phi$ is a satisfiable CNF formula$\}$
- One can slightly modify the proof of Cook-Levin theorem we just seen, to prove that CSAT $\in \mathcal{NPC}$! (take home exercise :-))
- A Boolean formula is in kCNF form, if it is a CNF formula, and all clauses have k literals.
- 3SAT $= \{\langle\phi\rangle \ : \ \phi$ is satisfiable 3CNF formula$\}$.
- Hence, to prove that 3SAT $\in \mathcal{NPC}$, it suffices to prove that CSAT $\leq_P$ 3SAT.

CSAT $\leq_P$ 3SAT

- The reduction maps CNF formulae to 3CNF ones "clause by clause".
- A clause with $d \leq 3$ literals is mapped to equivalent clause with 3 literals.

  $(x_1 \vee x_2) \longmapsto (x_1 \vee x_2 \vee x_2)$
- A clause with $d > 3$ literals is mapped to d clauses, built on the original literals together with $(d-3)$ new ones.

  $\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4 \vee x_8) \longmapsto$
  $\phi_3 = (x_1 \vee \overline{x_2} \vee y_1) \wedge (\overline{y_1} \vee \overline{x_3} \vee y_2) \wedge (\overline{y_2} \vee x_4 \vee x_8)$

Claim 11

ϕ has a satisfying assignment iff ϕ_3 does.

Proof's idea: (for case $d > 3$)
$\Leftarrow$ An assignment satisfying ϕ_3 cannot "rely" on new literals alone – at least one original literal must be satisfied.
$\Rightarrow$ An assignment satisfying ϕ, makes at least one literal per clause happy. In the "ϕ_3 clause" of this literal the new veriable is under no constraints. This enables propagation to a satisfying assignment that "relies" on new vars alone in rest of ϕ_3 clauses.

CSAT $\leq_P$ 3SAT, cont.

$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4 \vee x_8) \longmapsto$
$\phi_3 = (x_1 \vee \overline{x_2} \vee y_1) \wedge (\overline{y_1} \vee \overline{x_3} \vee y_2) \wedge (\overline{y_2} \vee x_4 \vee x_8)$

- Doing the above mapping to each clause of a CNF formula, we get a 3CNF that is satisfied iff the original one is.
- Since this mapping is polynomial time (?), we get CSAT $\leq_P$ 3SAT. ♣.

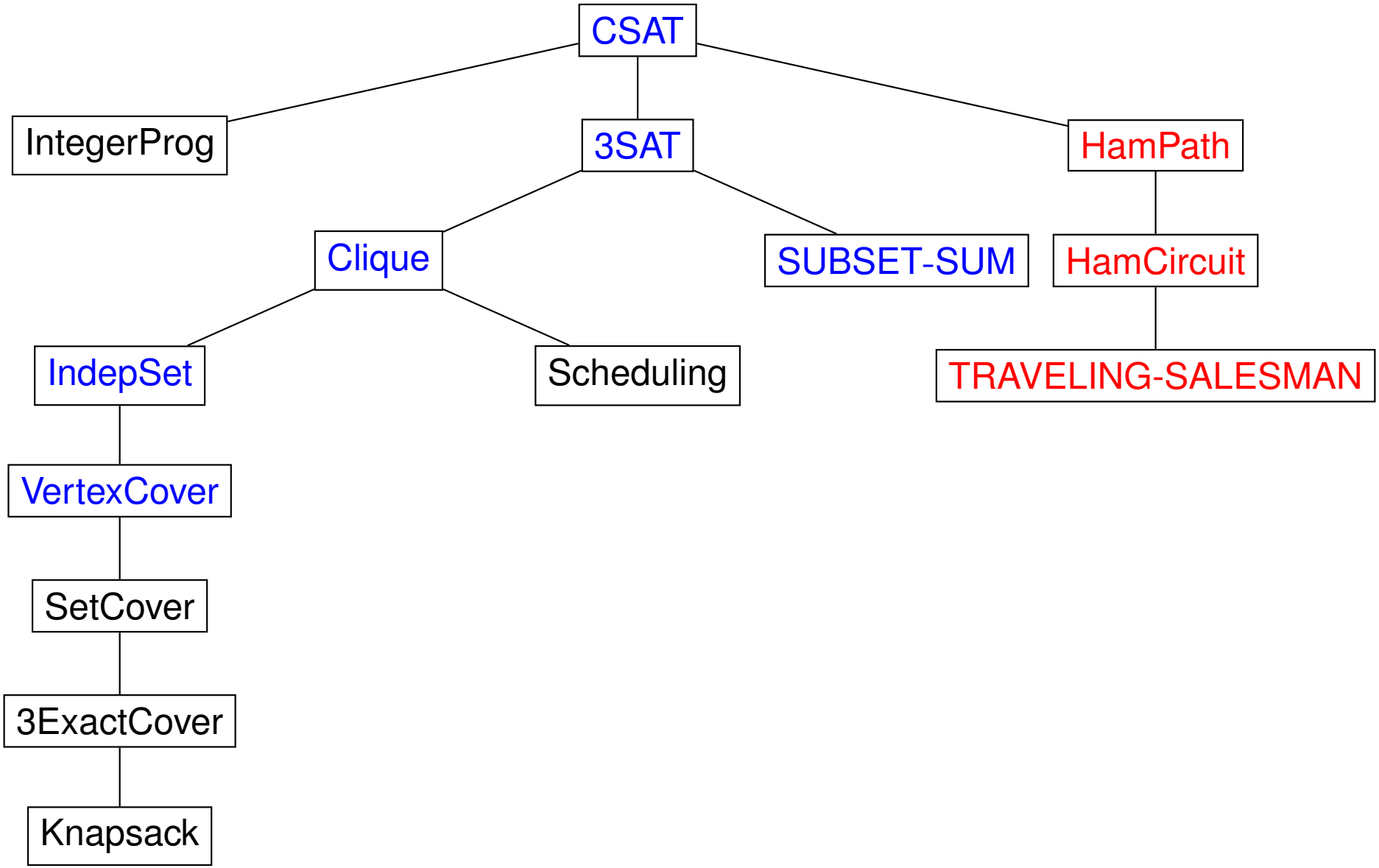
Section 4

Summary so far

Summary so Far

- We have seen that SAT and CSAT are NP-complete.
- CSAT $\leq_P$ 3SAT
- 3SAT $\leq_P$ CLIQUE
- CLIQUE $\leq_P$ IND-SET
- In recitation: CLIQUE $\leq_P$ VC (Vertex Cover), IND-SET $\leq_P$ PARTITION, PARTITION $\leq_P$ BIN-PACKING
- Hence, all of the above languages are NP-complete
- Full list of NP-complete languages contains hundreds or thousands of known NP-complete problems (from combinatorics, operation research, VLSI design, computational geometry, bioinformatics, . . .).
- NP-completeness of new and of old problems is still established these days.

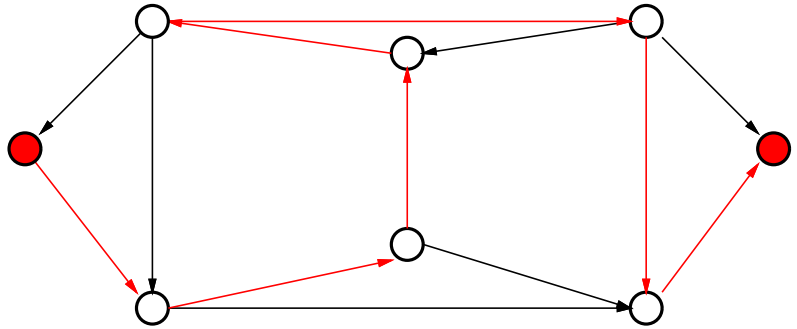
Chains of reductions: NPC languages



Section 5

Hamiltonian Path Is NP-Complete

(Directed) Hamiltonian path



A Hamiltonian path in a directed G visits each node exactly once.

$$\text{HAMPATH} = \{\langle G, s, t\rangle : G \text{ has Hamiltonian path from } s \text{ to } t\}$$

- We have seen that HAMPATH $\leq_P$ HAMCYCLE and HAMPATH $\leq_P$ TRAVELING-SALESMAN
- Will now show

Theorem 12

CSAT $\leq_P$ HAMPATH

thus establishing

Theorem 13

HAMPATH, HAMCYCLE, TRAVELING-SALESMAN $\in \mathcal{NPC}$

CSAT $\leq_P$ HAMPATH

For any CNF formula ϕ with clauses $c_1, \ldots, c_\ell$ and variables $x_1, \ldots, x_k$, we construct a directed graph G with vertices s and t, such that

ϕ is satisfiable iff $\exists$ a directed Hamiltonian path from s to t.

Turn to a separate pdf presentation:

http://tau-cm.wdfiles.com/local--files/course-schedule/ham-reduction-new.pdf

Section 6

SUBSET-SUM Is NP-complete

Reminder — SUBSET-SUM

$$\text{SUBSET-SUM} = \{\langle S = \{x_1, \ldots, x_k\}, t\rangle : \exists \{y_1, \ldots, y_m\} \subseteq S : \sum_{j=1}^{m} y_j = t\}$$

Example 14

- $\langle \{4, 11, 16, 21, 27\}, 25\rangle \in$ SUBSET-SUM because $4 + 21 = 25$.
- $\langle \{4, 11, 16, 21, 27\}, 26\rangle \notin$ SUBSET-SUM

- Clearly SUBSET-SUM $\in \mathcal{NP}$.
- Will now show

Theorem 15

3SAT $\leq_P$ SUBSET-SUM.

Thus establishing

Theorem 16

SUBSET-SUM $\in \mathcal{NPC}$.

3SAT $\leq_P$ SUBSET-SUM

Let ϕ be 3-CNF a boolean formula with

- variables $x_1, \ldots, x_k$
- clauses $C_1, \ldots, C_\ell$.

We "encode" ϕ into a set S containing $2k + 2\ell$ numbers in decimal notation, and a "target" number t, such that $\phi \in$ 3SAT $\iff$ $(S, t) \in$ SUBSET-SUM

Variable encoding

Variable x_i is encoded as two $(k+\ell)$-digit numbers: y_i and z_i.

Each decimal representation has two parts.

- Left-hand part, k digits: for both y_i and z_i, the i'th digit is 1, rest are 0s
- Right-hand part, ℓ digits: one digit for each clause
  - j^{th} digit of y_i is 1 if C_j contains x_i
  - j^{th} digit of z_i is 1 if C_j contains $\overline{x_i}$
  - rest are 0s

Example: $\phi = (x_1 \vee \overline{x_2} \vee \ldots) \wedge \ldots \wedge (\overline{x_1} \vee \overline{x_2} \vee \ldots)$

	1	2	...	k	C_1	...	C_ℓ
y_1	1	0	...	0	1	...	0
z_1	1	0	...	0	0	...	1
y_2	0	1	...	0	0	...	0
z_2	0	1	...	0	1	...	1

Clause encoding

Each clause C_i is encoded into two equal $(k+\ell)$-digit numbers: g_i and h_i.

In both numbers, the $(k+i)$'th digit is 1 and rest are 0.

	1	...	k	C_1	C_2	...	C_ℓ
$\vdots$							
g_1	0	...	0	1	0	...	0
h_1	0	...	0	1	0	...	0
g_2	0	...	0	0	1	...	0
h_2	0	...	0	0	1	...	0

Target encoding

The target t is assigned the following $(k+\ell)$-digit number

- first k digits are 1
- last ℓ digits are 3

	1	...	k	C_1	C_2	...	C_ℓ
$\vdots$							
t	1	...	1	3	3	...	3

Overall encoding

	1	2	...	k	C_1	C_2	...	C_ℓ
y_1	1	0	...	0	1	0	...	0
z_1	1	0	...	0	0	0	...	1
y_2	0	1	...	0	0	0	...	0
z_2	0	1	...	0	1	0	...	1
$\vdots$								
g_1	0	0	...	0	1	0	...	0
h_1	0	0	...	0	1	0	...	0
g_2	0	0	...	0	0	1	...	0
h_2	0	0	...	0	0	1	...	0
$\vdots$								
t	1	1	...	1	3	3	...	3

$S = \{y_1, z_1, y_2, z_2, \ldots, y_k, z_k, g_1, h_1, g_2, h_2 \ldots, g_\ell, h_\ell\}$.

Transformation takes $O(n^2)$ time.

ϕ is satisfiable $\implies$ $(S, t) \in$ SUBSET-SUM

Given satisfying assignment to ϕ, construct initial solution V to (S, t) as follows: If $x_i = 1$, add y_i to V; otherwise, add z_i to V.

- Each of the first k digits of $\sum_{v \in V} v$ is 1 (as in t)
- For $1 \leq j \leq \ell$, consider the $(k + j)$'th digit of $\sum_{v \in V} v$:
  - at least 1 and not more than 3 literals in C_j are 1
    $\implies$ the $(k + j)$'th digit is between 1 and 3.
  - Add "enough" $\{h_i, g_i\}$ to V to bring this digit up to 3.
    (Doing this does not effect the other digits of $(\sum_{v \in V} v)$).

Hence, $\sum_{v \in V} v = t$. ♣

$(S, t) \in$ SUBSET-SUM $\implies$ ϕ is satisfiable

Given a solution V to (S, t).

1. Summation causes no carries
   - All digits in the numbers of S are either 0 or 1.
   - Since ϕ is in 3CNF, each "column" contains at most five 1s.
2. For each $i \in [k]$, the subset V contains one of y_i or z_i, but not both.
3. For each $i \in [\ell]$, the subset V contains at least one element of $\{y_1, z_1, \ldots, y_k, z_k\}$ whose $(k+i)$'th bit is 1:
   - Each of final ℓ "columns" sums to 3.
   - The set $\{g_i, h_i\}$ contributes at most 2
   - So at least one must comes from literal taking the value 1

The satisfying assignment.

For $i \in [k]$: if $y_i \in V$, set $x_i = 1$; otherwise, set $x_i = 0$.

The assignment satisfies ϕ because all clauses are satisfied (see Item 3)

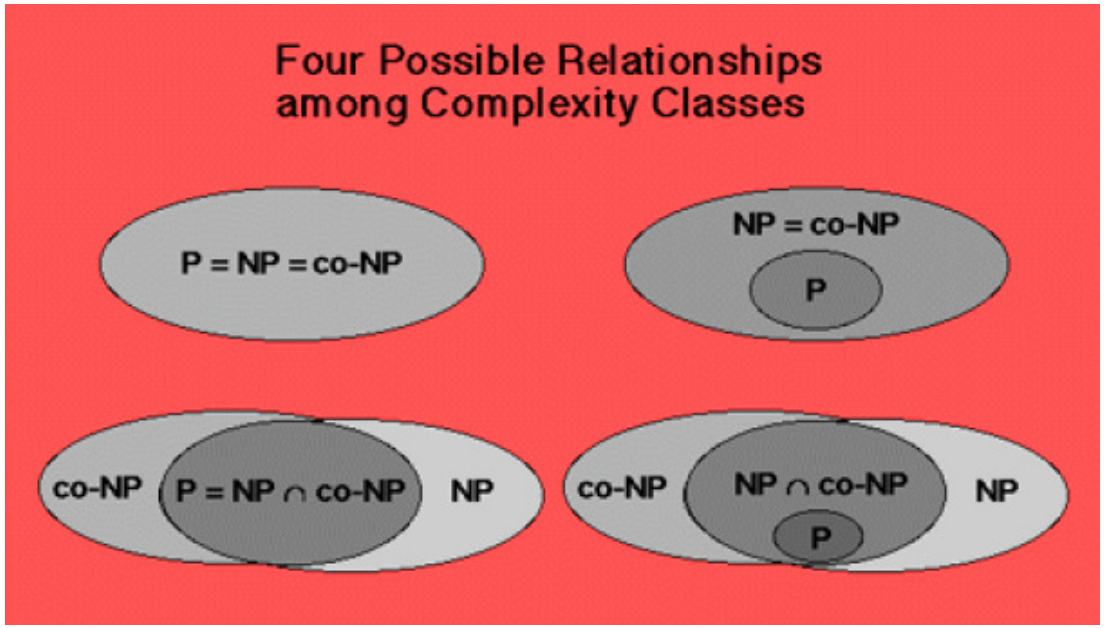
Section 7

Concluding Remarks

Ladies and Gentlemen, Boys and Girls

We have just ended the third part of the course:

Introduction to Computational Complexity (no more).



And with it, naturally, we've ended the whole course.

We hope you have enjoyed your flight, and look forward to meeting you in the future (but not in Moed B :-).

The dreaded exam

- All material covered in class and recitations, from all parts of course.
- Five “open” questions.
- You can bring and use two double sided A4 (normal size) pages marked with your name.
- See website for more info.
- Piece of cake.