Computational Models Spring 11: Lecture 13

- More Poly-Time Reductions:
- 3SAT $\leq_P$ DirHamPath
- 3SAT $\leq_P$ SUBSET-SUM
- SUBSET-SUM $\leq_P$ PARTITION (not in book)
- PARTITION $\leq_P$ BIN_PACKING (not in book)
- 3SAT $\leq_P$ IP (not in book)
- Sipser, Chapter 7, Section 7.5

The Language SAT (reminder)

Definition: A Boolean formula is in conjunctive normal form (CNF) if it consists of clauses, connected with $\wedge$s. Each clauseis a disjunction ($\vee$s) of literals.

For example $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6})$

Definition: SAT $= \{\langle \phi \rangle \mid \phi$ is satisfiable CNF formula$\}$

3SAT (reminder)

Definition: A Boolean formula is in 3CNF form if it is a CNF formula, and each clause has at most three literals.

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4)$$

Define

$$\text{3SAT} = \{\langle \phi \rangle \mid \phi \text{ is satisfiable 3CNF formula}\}$$

Clearly, if ϕ is a satisfiable 3CNF formula, then for any satisfying assignment of ϕ, every clause must contain at least one literal assigned 1.

NPC – Reminder

A language $\mathcal{B}$ is NP-complete if it satisfies

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

coNP-Completeness (analog notion)

A language $\mathcal{C}$ is coNP-complete if it satisfies

- $\mathcal{C} \in coNP$ (namely its complement is in NP), and
- For every $\mathcal{D}$ in coNP, $\mathcal{D} \leq_P \mathcal{C}$.

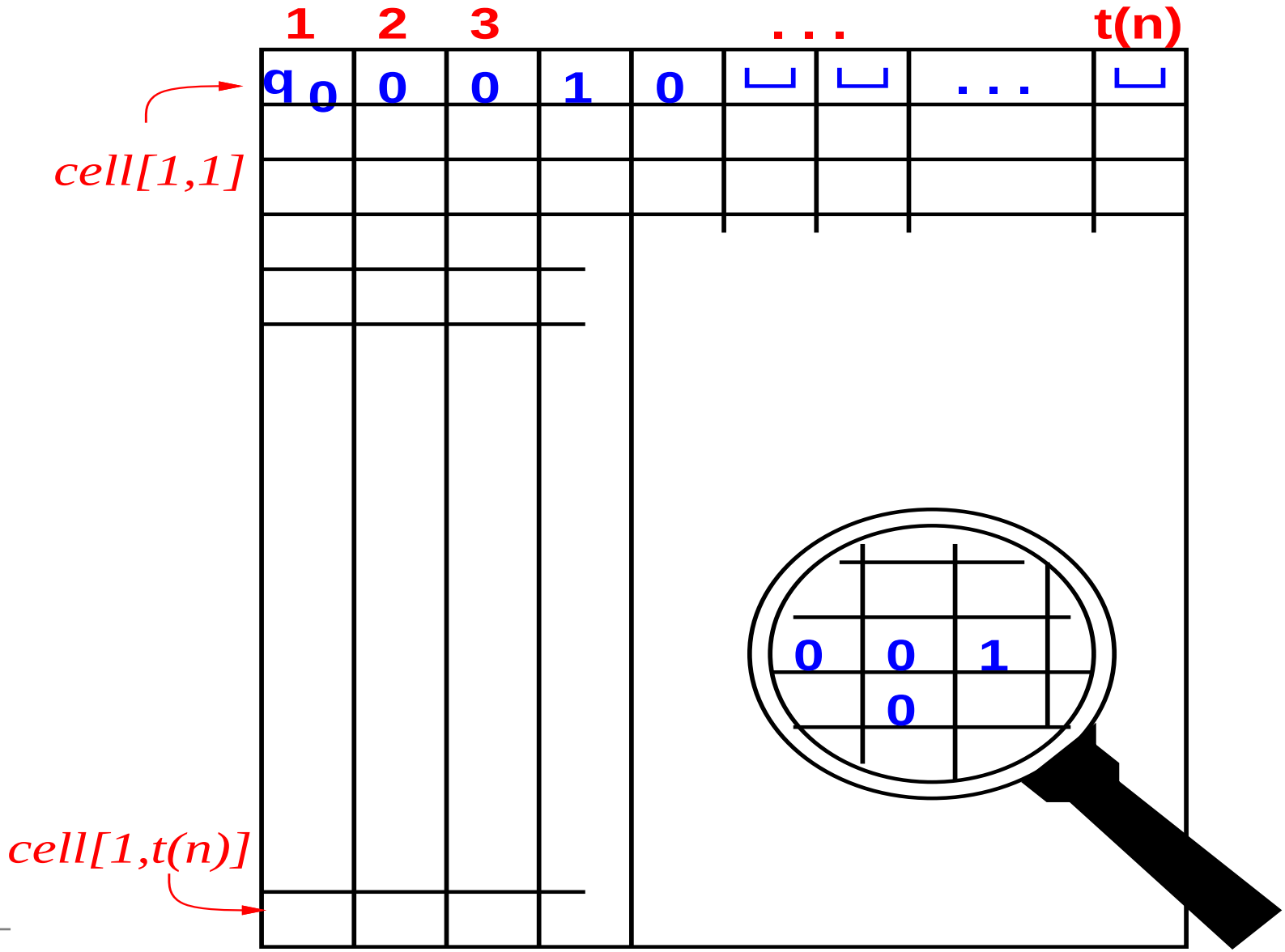
Cook-Levin (early 70s)

Theorem: SAT is NP complete.

Prf.: Membership ($SAT \in NP$) is easy, using satisfying assignment as certificate.

- Hard part is showing that every NP problem reduces to SAT in poly-time.
- **Idea**: Suppose $\mathcal{L} \in \mathcal{NP}$, and M is an NTM that accepts $\mathcal{L}$.
- On input w of length n, M runs in time $t(n) = n^c$.
- We consider the n^c-by-n^c tableau that describes the computation of M on input w.

The Tableau



Saw a Few Reductions So Far

- SAT $\leq_P$ 3SAT ($\Rightarrow$ 3SAT is NP-complete)
- 3SAT $\leq_P$ Clique ($\Rightarrow$ Clique is NP-complete)
- Clique $\leq_P$ Independent Set ($\Rightarrow$ IS is NP-complete)
- In recitation: Clique $\leq_P$ Vertex Cover ($\Rightarrow$ VC is NP-complete)
- DirHamPath $\leq_P$ DirHamCircuit
- DirHamCircuit $\leq_P$ DirTSP

- Will now show 3SAT $\leq_P$ DirHamPath, thus establishing NP-completeness of DirHamPath, DirHamCircuit, and DirTSP.

Directed Hamiltonian Path

For any 3CNF formula ϕ,

- we construct a directed graph G
- with vertices s and t
- such that ϕ is satisfiable iff there is a directed Hamiltonian path from s to t.

Directed Hamiltonian Path

Here is a 3CNF formula ϕ:

$$(a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots (a_k \vee b_k \vee c_k) \wedge$$

where

- each a_i, b_i, c_i is x_i or $\overline{x_i}$
- the ℓ clauses are $C_1, \ldots, C_\ell$,
- the k variables are $x_1, \ldots, x_k$.

DirHamPath: NP Completeness Proof

Turn to a separate pdf presentation:

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

Reminder: SUBSET-SUM

An instance of the problem

- A collection of numbers in binary/decimal, $r_1, \ldots, r_k$
- Target number t
- Question: does some subcollection add up to t?

$$\begin{aligned} \text{SUBSET-SUM} &= \{\langle S, t\rangle \mid S = \{r_1, \ldots, r_k\} \\ \exists T &\subseteq \{1, \ldots k\} \ \sum_{j \in T} r_j = t\} \end{aligned}$$

Collections are multi-sets: repetitions are allowed.

Subset Sum

Theorem: SUBSET-SUM is NP-complete.
Must demonstrate

- SUBSET-SUM $\in NP$
- every $A \in NP$ poly-time reducible to SUBSET-SUM.

Note that

- we have already seen the first (the subset is the certificate),
- we will now show that 3SAT $\leq_P$ SUBSET-SUM.

Subset Sum

For every 3-CNF formula, ϕ, we generate a SUBSET-SUM instance:

- Construct very large numbers in decimal notation (binary would work as well, but not unary).
- For each variable x_i of the formula ϕ, the set S contains a pair of numbers y_i, z_i.
- To "hit" the target, t, we will have to choose exactly one of these y_i, z_i:
  Take y_i if x_i is assigned T, take z_i if F.

Subset Sum

For every 3-CNF formula, ϕ, we generate a SUBSET-SUM instance:

- Construct very large numbers in decimal notation (binary would work as well, but not unary).
- For each clause C_j of the formula ϕ, the set S contains a pair of equal numbers g_j, h_j.
- If ϕ has ℓ variables $x_1, \ldots, x_\ell$ and k clauses $C_1, \ldots, C_k$, all numbers y_i, z_i, g_j, h_j are $\ell + k$ digits long.

- Clauses will also correspond to distinct positions in the decimal representations of these numbers.

Subset Sum

Let ϕ be a boolean formula with

- variables $x_1, \ldots, x_\ell$
- clauses $C_1, \ldots, C_k$.

We construct a table whose rows are $2\ell + 2k$ numbers in decimal notation.

Subset Sum

Rows of table are labeled with

- variable encodings: $y_1, z_1, y_2, z_2, \ldots, y_\ell, z_\ell$,
- clause encodings: $g_1, h_1, g_2, h_2, \ldots, g_k, h_k$,
- goal t

Subset Sum

Each x_i encoded as y_i, z_i.
Decimal representation has two parts.

- left-hand part: digit i is 1, rest 0
- right-hand part: one digit for each clause
  - j^{th} digit of y_i is 1 if C_i contains x_i
  - j^{th} digit of z_i is 1 if C_i contains $\overline{x_i}$

Subset Sum Mapping: Example

$$(x_1 \vee \overline{x_2} \vee \cdots) \wedge \cdots \wedge (\overline{x_1} \vee \overline{x_2} \vee \cdots)$$

	1	2	...	ℓ	C_1	...	C_k
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

Subset Sum

S also contains g_j and h_j for each clause C_j

- they are equal
- digit $\ell + j$ is 1, rest are 0.

	1	...	ℓ	C_1	C_2	...	C_k
$\vdots$							
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

Subset Sum

Bottom line of table is goal t

- first ℓ digits are 1
- last k digits are 3 (three!)

	1	$\ldots$	ℓ	C_1	C_2	$\ldots$	C_k
$\vdots$							
t	1	$\ldots$	1	3	3	$\ldots$	3

Subset Sum

	1	2	...	ℓ	C_1	C_2	...	C_k
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

Subset Sum

Claim: If ϕ is satisfiable, then some subset of S sums to t.

- select y_i if x_i is true.
- select z_i if x_i is false.

Adding them up

- yields a 1 in the first ℓ digits
- matching t so far

Subset Sum

Claim: If ϕ is satisfiable, then some subset of S sums to t.

- select y_i if x_i is true.
- select z_i if x_i is false.

Each of the last k digits

- at least one literal is true,
- number of true literals is between 1 and 3,
- sum so far: at least 1, at most 3.
- so pick enough h_i, g_i to bring this digit up to 3.
- this can be done separately for each clause.

This completes one direction.

Subset Sum

Claim: If some subset of S sums to t, then ϕ is satisfiable.

- All digits in S are 0 or 1.
- Because ϕ is in 3CNF, each column contains at most five 1s (why?).
- Therefore, summation causes no carries between columns.
- Subset must contain one of y_i or z_i, but not both.

Subset Sum

Here is the satisfying assignment.

- if subset includes y_i, then x_i is true
- if subset includes z_i, then x_i is false

This assignment must satisfy ϕ because

- each of final k columns sums to 3.
- at most 2 come from g_i, h_i
- so at least one must comes from true literal

Final point: transformation takes $O(n^2)$ stages. ♠

“Paradox” ?!

- Saw a polynomial time dynamic programming algorithm.
- To resolve this “paradox”, look at the fine print.
- The algorithm is polynomial time in what?
- What do you mean? Polynomial time in the input length, of course!
- But the inputs are integers. How are they encoded?
- In the NP completeness proof, they are encoded in binary/decimal (for a ϕ with 100 clauses and 50 variables, numbers will be about 10^{150}).
- In the algorithm, numbers are encoded in unary. So the algorithm can handle numbers around 150 comfortably, but nothing like 10^{150}!

Partition

An instance of the problem

- List of integers $r_1, \ldots r_k$
- Question: Can we partition the items such that the sum of the two subsets is identical.

$$\sum_{i \in S} r_i = \sum_{j \notin S} r_j$$

$PARTITION =$
$\left\{ \langle r_1, \ldots, r_k \rangle | \exists S \subset \{1, \ldots k\} \sum_{i \in S} x_i = \sum_{j \notin S} x_j \right\}$

Observation: PARTITION is a special case of SUBSET-SUM

Partition

Theorem: SUBSET-SUM $\leq_P$ PARTITION

Reduction from SUBSET-SUM:

- Input: $r_1 \ldots r_k$ and t
- Let $A = \sum_{i=1}^{k} r_i$ and $U = \{1, \ldots, k\}$.
- Output: $r_1, \ldots, r_k$, b and c,
  where $b = 2A - t$ and $c = A + t$.

Correctness:

- If there exists an S for the SUBSET-SUM, then $S \cup \{b\}$ solves PARTITION.
- Assume $(S : U - S)$ solves the PARTITION.
- W.l.o.g. $b \in S$ and $c \notin S$
- Then $S \setminus \{b\}$ is a solution to SUBSET-SUM.

Bin Packing

An instance of the problem

- List of integers $r_1, \ldots r_k$; bin size B; number of bins m
- Question: Is there a partition to m subsets $S_1, \ldots, S_m$, such that

$$\sum_{i \in S_j} r_i \leq B$$

$BIN\_PACKING =$

$\left\{ \langle r_1, \ldots, r_k, B, m \rangle | \exists \text{ partition } S_1 \ldots S_m \forall j \sum_{i \in S_j} r_i \leq B \right\}$

Bin Packing

Theorem: Partition $\leq_P$ BIN_PACKING

Reduction from PARTITION:

- Input: $r_1 \ldots r_k$
- Let $A = \sum_{i=1}^{k} r_i$
- Output: $r_1, \ldots, r_k$, $B = A/2$ and $m = 2$.

Integer Programming (IP)

- Definition: A linear inequality has the form

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \leq b$$

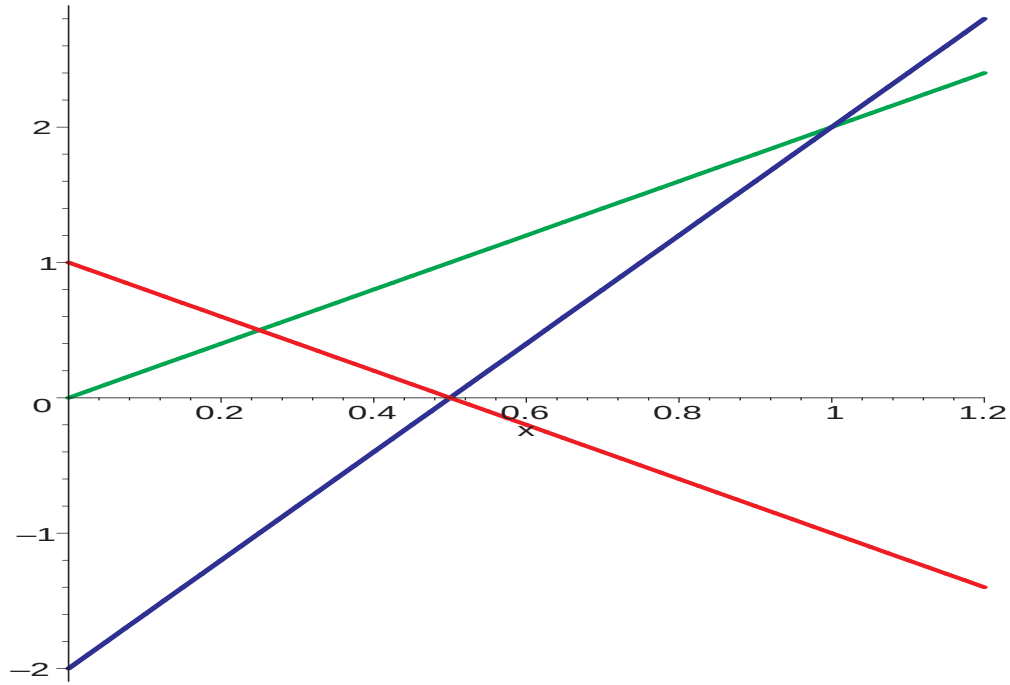
  where $a_1, \ldots, a_n, b$ are real numbers, and $x_1, \ldots, x_n$ are real variables.

- The Integer Programming (IP) problem:
- Input: A set of m linear inequalities with integer coefficients (a_i, b) in n variables $x_1, x_2, \ldots, x_n$.
- The language IP is the collection of all systems of linear inequalities that have a solution where all x_i are integers.

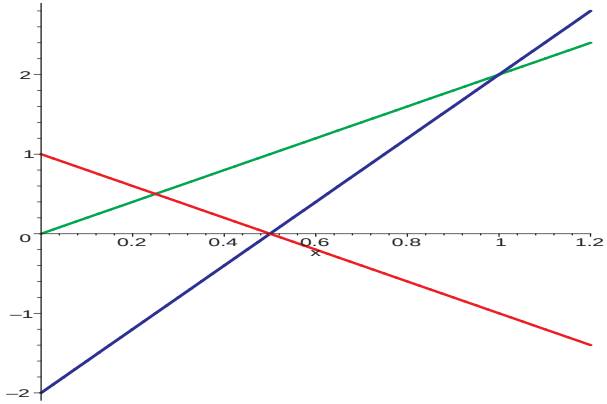
Integer Programming: Example

Consider the following system of linear inequalities

$$
\begin{aligned}
y &\leq 2x && \text{green line} \\
-2x + 1 &\leq y && \text{red line} \\
4x - 2 &\leq y && \text{purple line} \\
0 \leq x &\leq 1 \\
0 \leq y &\leq 2
\end{aligned}
$$



Integer Programming: Example



This set does have a unique solution: the right hand corner of the solid triangle, $(1, 2)$.

But if we change the constraint on y to $0 \leq y \leq 1$, then we'd have no solution with integer coordinates, even though there are many solutions with rational, or real, coordinates.

Will now show IP is NP complete.

Membership in NP non-trivial (why?)

Integer Programming: NP

Consider the following equations

$$\begin{aligned} x_0 &= 1 \\ x_1 &= 2x_0 \\ &\vdots \\ x_n &= 2x_{n-1} \end{aligned}$$

The solution $x_n = 2^n$.

Solving a linear set of n equations with n unknowns: $Ax = b$, where $|a_{i,j}| < M$ and $|b_i| < M$.

Q: How large can $|x_i|$ get?!

Recall Cramer's Rule, using determinants. $|A| < n!M^n$.

SAT $\leq_P$ IP

SAT $= \{\langle\phi\rangle \mid \varphi$ is a satisfiable CNF formula$\}$

For example, the following formula is in SAT:
$(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6})$

Let φ be a CNF formula with m clauses and n variables $x_1, \ldots, x_n$ (either $x_i, \bar{x}_i$, or both, can appear in φ).

Will reduce φ to an IP instance with $2n$ variables $x_1, y_1, \ldots, x_n, y_n$ and $m + 2n$ linear inequalities, and n linear equalities (how ?).

SAT $\leq_P$ IP

- Each x_i in φ corresponds to the variable x_i in IP.
- Each $\bar{x}_i$ in φ corresponds to the variable y_i in IP.

- For each i, we add the inequalities $x_i \geq 0$, $y_i \geq 0$, and the equality $x_i + y_i = 1$
  (what do these three express?)
- For each clause k, we add the inequality
  $\sum_{z_j \in Clause_k} z_j \geq 1$
  (what does this inequality express?)
- For example, $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4)$ is translated to $x_1 + y_2 + y_3 + x_4 \geq 1$.

SAT $\leq_P$ IP: Example

$\varphi = (x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6})$

translates to

$$
\begin{aligned}
x_1 + y_2 + y_3 + x_4 &\geq 1 \\
x_3 + y_5 + x_6 &\geq 1 \\
x_3 + x_6 &\geq 1 \\
x_1 \geq 0,\ y_1 \geq 0,\ x_1 + y_1 &= 1 \\
x_2 \geq 0,\ y_2 \geq 0,\ x_2 + y_2 &= 1 \\
x_3 \geq 0,\ y_3 \geq 0,\ x_3 + y_3 &= 1 \\
x_4 \geq 0,\ y_4 \geq 0,\ x_4 + y_4 &= 1 \\
x_5 \geq 0,\ y_5 \geq 0,\ x_5 + y_5 &= 1 \\
x_6 \geq 0,\ y_6 \geq 0,\ x_6 + y_6 &= 1
\end{aligned}
$$

SAT $\leq_P$ IP: Validity (sketch)

Should show

(a) Reduction g is poly-time computable

(b) $\varphi \in$ SAT $\implies g(\varphi) \in IP$

(c) $g(\varphi) \in IP \implies \varphi \in SAT$.

- Poly time: easy (verify details!).
- Suppose $\varphi \in$ SAT. Take a satisfying assignment.
  If $x_i = 1$ assign $x_i = 1, y_i = 0$ in IP.
  If $x_i = 0$ assign $x_i = 0, y_i = 1$ in IP.
- So "sanity check" constraints satisfied. "Clause constraints" are satisfied due to at least one literal satisfied in each clause., implying $g(\varphi) \in IP$.
- $g(\varphi) \in IP \implies \varphi \in SAT$ is similar. ♣

Chains of Reductions: NPC Problems