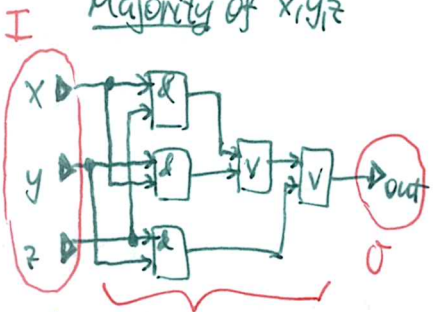
Brief detour: From formulas to Boolean circuits...

Df: A Combinatorial Circuit consists of:

- a finite alphabet Σ,
- finite sets I (input terminals) $= \{i_1, \dots, i_{|I|}\}$, O (output terminals) $= \{o_1, \dots, o_{|O|}\}$, H (gates) $= \{h_1, \dots, h_{|H|}\}$ } pairwise disjoint
- directed acyclic multigraph $(I \cup O \cup H, E)$
- arity: $a: H \to \mathbb{N}$
- assignment of functions to gates $F: h \mapsto (f_h: \Sigma^{a(h)} \to \Sigma)$
- assignment of gate inputs to incoming edges: $\ell: (u,v) \in E \mapsto i \in \{1 \dots a(v)\}$

Example: Majority of x, y, z



Where:

- $\forall i \in I \ \deg^{in}(i) = 0$
- $\forall o \in O \ \deg^{in}(o) = 1, \ \deg^{out}(o) = 0$
- $\forall h \in H \ \deg^{in}(h) = a(h)$ & $\forall i \in \{1 \dots a(h)\} \ \exists!$ edge $(x,h) \in E: \ell((x,h)) = i$

Df: Boolean Circuit: Comb. circuit with $\Sigma = \{0,1\}$

Df: Computation of a circuit proceeds in steps.

Sketch of:
- Step 0: input terminals and arity-0 gates (constants) have defined values.
- Step $i+1$: gates whose input is defined in step at most i produce output.

→ Layer #i: gates/terminals that produce output in time i, but not earlier



As the graph is acyclic, gate outputs never change and every gate/terminal (& output terminals) is defined within finite # steps.

⇒ the circuit computes a function from $\Sigma^{|I|}$ to $\Sigma^{|O|}$.

Bounding arity: Since a single gate of high arity can compute anything in 1 step, we will bound arity by 2. (Actually, any fixed constant > 1 would work.)

Circuit complexity: Time $\approx$ # layers (# steps of computation)
Space $\approx$ # gates
} BTW circuits are an interesting model of parallel computing



Boolean formulas $\approx$ circuits with tree structure (except for inputs)

Lemma: Every function $f: \{0,1\}^k \to \{0,1\}^l$ can be computed by a Boolean circuit (in fact it's a formula in DNF) consisting only of AND, OR and NOT gates (and constants). (OR can be replaced by $\overline{\bar{x} \& \bar{y}}$)

Proof:
1. n-input AND/OR can be computed by a tree of 2-input ANDs/ORs.
2. Function with multiple-bit output: replace by l single-bit functions.
3. Function whose truth table contains exactly one 1:
   e.g. $\neg x_1 \& \neg x_2 \& \neg x_3 \& x_4$ ... 1 at position 0001
4. Truth table with multiple 1's: OR functions for each 1.
5. Otherwise it's constant 0.

} produces circuits of exponential size, but good enough for k, l constant

Corollaries: ① can simulate arbitrary gates of fixed arity with $O(1)$ space/time overhead. (26)
② can simulate arbitrary comb. circuit by a Boolean circuit (binary-encoded Σ^*)

Problem: A circuit handles inputs of constant size only.
↳ a "program" is a family of circuits $C_0, C_1, \ldots$
where C_n solves the problem for inputs of size n.

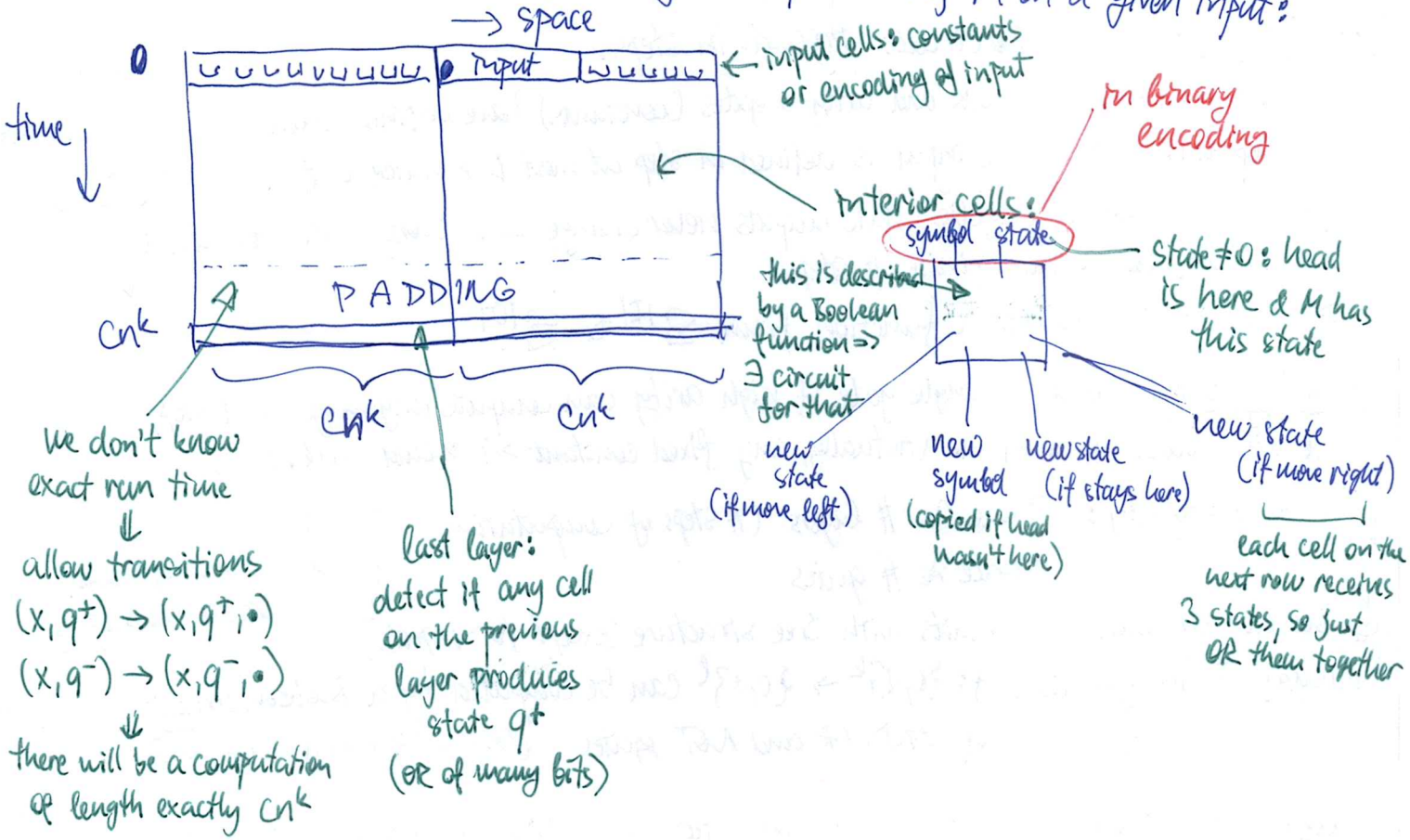
But: If we allow arbitrary sequences, we can compute undecidable problems:
$L = \{\alpha \mid |\alpha| \text{ written in binary with leading 1 removed} \in L_u\}$ (arbitrary enumeration of binary strings)

- So we usually require the family to be **uniform**: there is an algorithm which for every n produces C_n in time $\text{poly}(n)$.

So languages decidable by uniform circuit families $= P$

Theorem: For every $L \in P$ there is $f \in PF$ s.t. for every n, $f(n)$ is an (encoding of) Boolean circuit with n inputs and 1 output which decides L for strings of length n.

Proof: Let M be a 1-tape TM deciding L in time at most $c \cdot n^k$ for some $c, k \in \mathbb{N}$. (WLOG ↙ with obvious meaning (computes char. function of L))
We will build a circuit producing a computation of M on a given input:



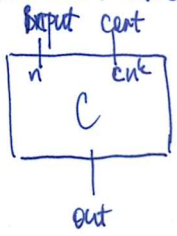
Df: CIRCUIT-SAT: given a Boolean circuit with 1 output, is there an input for which the output is true?

↳ Obviously, this is in NP.

Thm: CIRCUIT-SAT is NP-complete.

Proof: • When reducing from $L \in NP$ to C.-SAT: consider verifier $V \in P$ & upper bound cn^k for certificate size.

- Adapt verifier to accept certificates of size exactly cn^k (using reversible padding like 10*)
- Find Boolean circuit for V on inputs of size $n + cn^k$:



& fix input terminals to input α

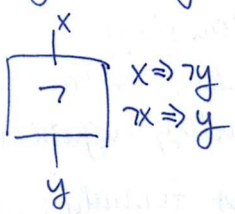
↓

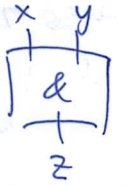
SAT for $C_\alpha(\text{cert})$ computes $\alpha \in L$

done inside the reduction when receiving input α of size n

Lemma: CIRCUIT-SAT $\leq_m^P$ SAT.

Proof: Assume WLOG that all gates are AND and NOT.
Introduce new variables for gate outputs.
Add consistency-checking clauses:





$x \& y \Rightarrow z$
$\neg x \Rightarrow \neg z$
$\neg y \Rightarrow \neg z$

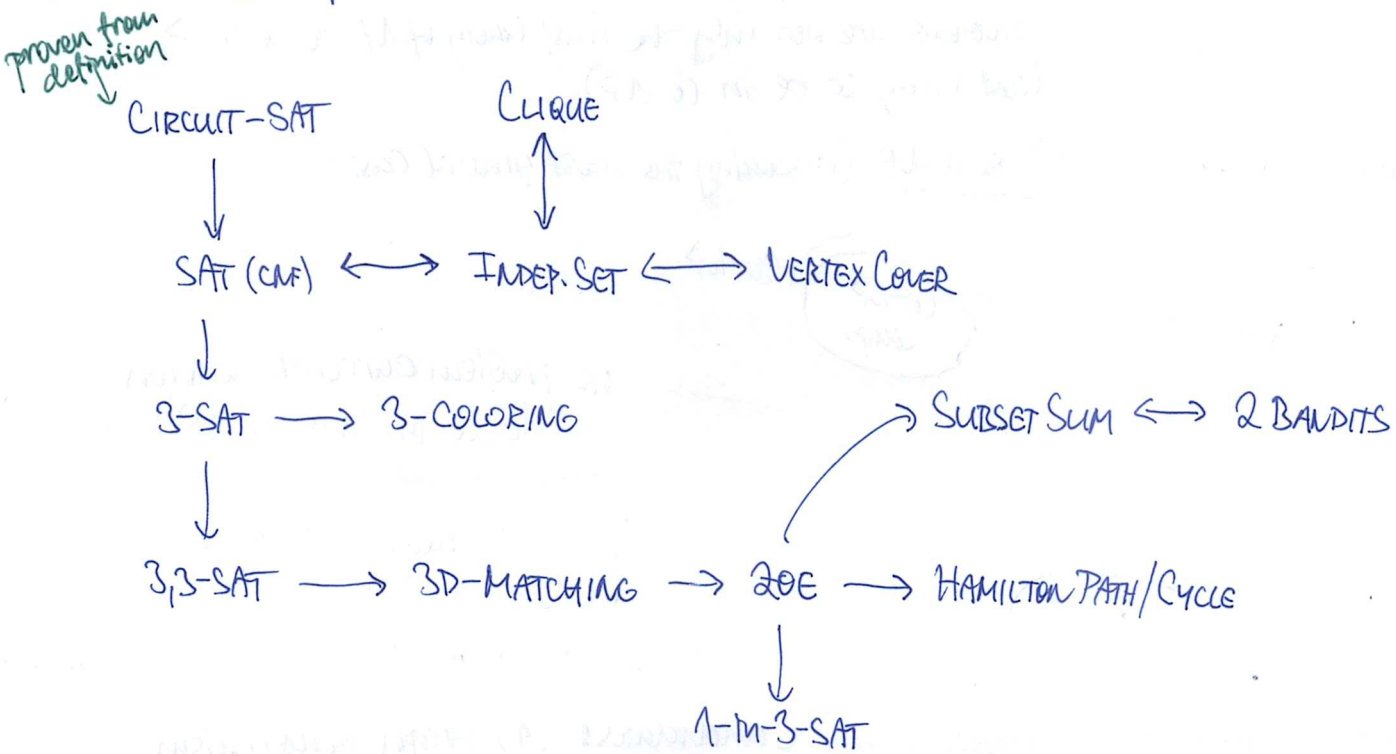
this is $\neg x \vee \neg y \vee z$ in CNF

this also means that SAT for general (non-CNF) formulas is reducible to SAT

BTW we produced an instance of 3-SAT ☺

Corollary: SAT is NP-complete. [This is Cook-Levin theorem!]

Map of NP-complete problems we encountered until now:



Further SAT variants: 2-SAT is in P

E3, E3-SAT (clauses of size exactly 3, vars have exactly 3 occurrences)
surprise: all instances satisfiable ?

The class co-NP

Df: For a language $L \subseteq \{0,1\}^*$ we define its complement $\overline{L} := \{0,1\}^* \setminus L$

Df: For a class $\mathcal{C}$ of languages: $\text{co-}\mathcal{C} := \{\overline{L} \mid L \in \mathcal{C}\}$ → $\text{co-P} = \text{P}$

$A = B \Leftrightarrow \text{co-}A = \text{co-}B$

$A \subseteq B \Leftrightarrow \text{co-}A \subseteq \text{co-}B$

$\text{co-co-}A = A$

Let's study co-NP...

- $P \subseteq NP \cap \text{co-NP}$ ... open if the inclusion is strict
- if $P = NP$, then $NP = \text{co-NP}$ (contrapositive)
- if $NP \neq \text{co-NP}$, then $P \neq NP$ ~~(because $P = \text{co-P}$)~~
- as $K \leq_m^P L \Leftrightarrow \overline{K} \leq_m^P \overline{L}$, we have: L is NP-complete $\Leftrightarrow$ $\overline{L}$ is co-NP-complete
- certificate-based def.: $L \in \text{co-NP} \equiv \exists V \in P: (x \in L \Leftrightarrow \forall \beta \in \{0,1\}^*, |\beta| \in \text{poly}(|x|)\ V(x,\beta))$

↳ so $\overline{SAT}$ is co-NP-complete

↑ this is not UNSAT (unsatisfiability), because for strings which do not encode a formula, we still have to answer 1 in $\overline{SAT}$ but 0 in UNSAT (for CNF)

↳ but $\overline{SAT} \leq_m^P$ UNSAT, so UNSAT is co-NP-comp.

↳↑ $\exists \vec{x}\, \varphi(\vec{x}) \Leftrightarrow \forall \vec{x}\, \neg\varphi(x)$ → so $\neg\varphi$ is a tautology & if φ is in CNF, $\neg\varphi$ can be written in DNF by propagating negation

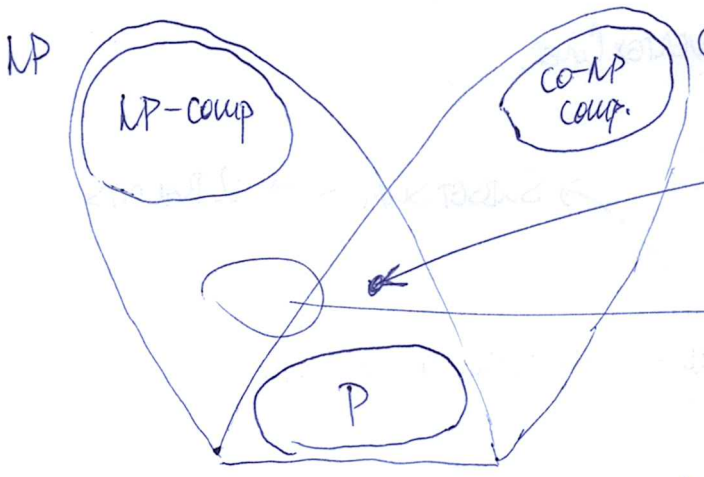
So: TAUTOLOGY := $\{\alpha \mid \alpha$ is (encoding of) DNF formula which is tautological$\}$ is also co-NP-complete (this is the most standard co-NP-c. problem)

Formally: TAUTOLOGY $\leq_m^P$ UNSAT $\leq_m^P \overline{SAT}$

Exercise: If $L \in \text{co-NP}$ is NP-complete, then $NP = \text{co-NP}$.

(so NP-comp. problems are not only the least likely of NP to be in P, but also least likely to be in co-NP).

Landscape of P vs. NP vs. co-NP (assuming the most general case)



no problem currently known to be in $(NP \cap \text{co-NP}) \setminus P$

(former candidate: primality testing, but it's known to be in P now)

NP-intermediate problems in $NP \setminus (\text{NP-comp.} \cup P)$

Ladner's thm: if $P \neq NP$, $\exists L$ NP-intermediate.

Candidates: ① graph isomorphism (known to be in DTIME($n^{\log n}$))

② factoring (given x,a,b: does x have a factor in $[a,b]$?)

Non-deterministic TM (NTM)

extend $\delta: Q \times \Gamma^k \to \mathcal{P}(Q \times \Gamma^k \times \{\leftarrow, \rightarrow, \circ\}^k)$ ... choice of instruction from the set

- successor relation is not a function → multiple computations for a given output
- if $\delta(q,x) = \emptyset$, we assume rejection.
- input α accepted ≡ ∃ at least one accepting computation
- halting ≡ all computations halt
- time & space: maximum over all computations

↓ wlog $|\delta(q,x)| \leq 2$

encumeration works again (NM_α = NTM with code α), we have an universal NTM &c.

Exercise: k-tape → 2-tape NTM with only constant-factor slowdown

Df: Non-deterministic complexity classes: NTIME(f), NTIMEF(f) for functions
└ all accepting computations must agree on result, ∃ at least one accepting comp.

Theorem: $NP = NTIME(poly(n)) := \bigcup_{k \geq 0} NTIME(n^k)$

Proof: ⊆ the NTM guesses the certificate using non-determinism & then it runs the verifier
⊇ the certificate encodes non-deterministic choices

Example: The following problem is NP-complete:
$\{\langle \alpha, \beta, 1^t \rangle \mid NM_\alpha$ accepts input β within t steps$\}$ → ∈NP: simulate $NM_\alpha(\beta)$ using universal NTM with an "alarm clock" (reject after t steps)

↓ reduction: calculate $t = p(n)$, $\alpha :=$ code of NTM solving source problem, pass β

Space Complexity

We want to count only "work space" of the TM.

3 types of tapes:

- input tape: read-only, head doesn't leave more than 1 cell before/after input string
- k work tapes: read-write
- output tape: write-only, head cannot move left

} space used by computation := # visited cells on the work tapes (for NTM: max over computations)

This doesn't change time complexity classes: we can copy input → work → output with constant slowdown

We can encode information in position of head on input tape.
- this makes a difference if work space $\in o(\log n)$
- otherwise we can keep track of the head position in binary

Space classes: (defined using machines which always halt)

DSPACE(f), NSPACE(f) } decision problems
& DSPACEF(f), NSPACEF(f) } functions

$PSPACE = DSPACE(poly(n))$
$NPSPACE = NSPACE(poly(n))$

We want f to be:
1. non-decreasing
2. space-constructible
   ∈ $f(n)$ can be computed from 1^n, result in binary, in space $O(f(n))$
3. usually $f(n) \geq \log n$

} proper space-complexity function