- Sps L is in NP w/ alphabet A
- Let M, p be t.m. and poly witnessing $L \in NP$.

  (Here using alt. def'n of NP above — t.m.'s instead of ntm's)

- Sps M has states $Q_s, Q_y, Q_n, Q_1, \dots, Q_m$ and alphabet of M is $B = \{\sigma_0, \dots, \sigma_N\}$ (which includes letters in A as well as separator ; etc)

- We define a polynomial-time reduction $f: A^{<\mathbb{N}} \to D^{<\mathbb{N}}$ where $D = \{x, 0, 1, \wedge, \vee, \neg, (,)\}$ is the alphabet we use to encode SAT

  $\hookrightarrow$ we encode variables x_i as $x(\dots)$ where $(\dots)$ is i in binary

  - Allows us to encode SAT probs w/ different #s of variables x_i.

- Cells of tape of M indexed by $\mathbb{Z}$. input word's first letter always in cell # 0.

- Note that on an input $w;u$ (word w w/ certificate u) with $|w| = n$

M only checks cells in ~~[illegible]~~ interval
$[-p(n), p(n)]$ before halt.

- Fix ~~[illegible]~~ w and $u \in A^{<\mathbb{N}}$ ~~[illegible]~~ with $|u| \leq p(|w|)$.
  (let $n = |w|$) (let $w' = w;u$)
- Want to define $f(w')$ to be a SAT problem encoding problem of checking $w \in L$ given certificate (i.e. checking if M halts on input $w;u$ in state Q_y)

- We associate to M, w, u the following propositional variables:
  - $S_{i,k}$ for $0 \leq i \leq p(n)$ and $k \in \{s, y, n, l, \ldots, m\}$

Truth value of $S_{i,k}$ reps M being in State Q_k at stage i of computation
  - $H_{i,j}$ for $0 \leq i \leq p(n)$, $-p(n) \leq j \leq p(n)$

" " $H_{i,j}$ reps head of M being on cell j " " "
  - $T_{i,j,l}$ for $0 \leq i \leq p(n)$, $-p(n) \leq j \leq p(n)$, $0 \leq l \leq N$. Reps jth cell of tape @ ith step of computation contains symbol σ_l.

– We define $f(w)$ to be set of following clauses (or really: their conjunction – a SAT problem)

For every $i \le p(n)$ the clauses

(1i) • $\bigvee_k S_{i,k}$ ("M is in some state at stage i")

• $\neg S_{i,k} \vee \neg S_{i,k'}$ for all $k \neq k'$ ("M cannot be in two distinct states")

• $\bigvee_{-p(n) \le j \le p(n)} H_{i,j}$ ("M is at some cell j")

• $\neg H_{i,j} \vee \neg H_{i,j'}$ for all $j \neq j'$ ("M not at two cells at once")

• $\bigvee_{\substack{l \le n \\ -p(n) \le j \le p(n)}} T_{i,j,l}$ ("there is some symbol σ_l in cell j")

• $\neg T_{i,j,l} \vee \neg T_{i,j,l'}$ for all j, and $l \neq l'$ ("not two symbols in cell$_j$")

Then also the clauses

② $S_{0,s}$, $H_{0,0}$, T_{0,j,w_j} for every j, where w_j is jth letter of w, if or empty symbol if $j > l(w)$ or < 0. (these instructions rep initial state)

③ $\bigvee_{i \le p(n)} S_{i,y}$ ("M ~~halts~~ halts in state q_y before $p(n)$")

Finally, for every instruction $(q_k, \sigma_\ell, \sigma_{\ell'}, D, q_{k'})$ if e.g. $D = R$, the clauses

- $\neg S_{i,k} \vee \neg H_{i,j} \vee \neg T_{i,j,\ell} \vee S_{i+1,k'}$

(if in state q_k @ stage i, w/ tape at cell j; symbol is σ_ℓ, go to state $q_{k'}$)

- $\neg S_{i,k} \vee \neg H_{i,j} \vee \neg T_{i,j,\ell} \vee T_{i+1,j,\ell'}$

(... and write $\sigma_{\ell'}$)

- $\neg S_{i,k} \vee \neg H_{i,j} \vee \neg T_{i,j,\ell} \vee H_{i+1,j+1}$

(... and move right)

analogously if $D = L$.

- Convince yourself: $f(w;u)$ can be computed in poly time from $w;u = w'$ (which is poly length in $|w|$)

- We claim: $w \in L$ (as checked by M on certificate u)

iff

$f(w;u) \in SAT$

($\Rightarrow$) if $w \in L$ (as checked by M on $w;u$) then we can assign truth values to variables above according to the computations of M on input $w;u$ — by construction this assignment satisfies the conjunction $f(w;u)$

($\Leftarrow$) If t is a truth assignment satisfying $f(w, u)$ for some $u \le p(|w|)$ then by induction ~~one~~ one can check that $t(S_{i,k})$ = true iff M is in state Q_k at step i of computation; likewise for the $H_{i,j}$'s and $T_{i,j,l}$'s.

Thus since $\bigvee_{i \le p(n)} S_{i,y}$ holds, M must halt in state Q_y before time $p(n)$ on input w, u

i.e. $w \in L$ ✓

<u>Problem</u> (3-SAT): Given a ~~...~~ set of 3-disjunctions $C_i, i \le n$ (i.e. each C_i of form $x_i \vee y_i \vee z_i$ each of x_i, y_i, z_i a var. or negated var.) determine if the conjunction $\bigwedge_{i \le n} C_i$ is satisfiable.

In contrast to 2-SAT, 3-SAT is already as complicated as possible among SAT problems

<u>Prop'n</u> 3-SAT is NP-complete.

<u>Pf</u>: - we reduce SAT to 3-SAT

i.e. we show that every SAT prob can be translated into an equiv 3-SAT prob in poly time.

- follows from following (x, y etc are vars or negated vars):

① x is equiv to
$(x \vee y \vee z) \wedge (x \vee \neg y \vee z) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg z)$

② $x \vee y$ is equiv to
$(x \vee y \vee z) \wedge (x \vee y \vee \neg z)$

③ $x \vee y \vee z$ equiv to itself

④ $x_1 \vee \dots \vee x_n$ (for $n > 3$) is satisfiable iff: $(x_1 \vee x_2 \vee y_1) \wedge (x_3 \vee \neg y_1 \vee y_2) \wedge (x_4 \vee \neg y_2 \vee y_3) \wedge \dots \wedge (x_{n-2} \vee \neg y_{n-4} \vee y_{n-3}) \wedge (x_{n-1} \vee x_n \vee \neg y_{n-3})$ is satisfiable.

- for ④: not saying expressions are equivalent (they're not), ~~[illegible]~~ the y_i are new variables (distinct from all others) - can choose truth value w/o constraint

- "is satisfiable" means relative to constraints imposed by other conjuncts.

- to see ④:

($\Rightarrow$) • if x_1 or x_2 true make conjunct true by choosing y_i false $\forall i$

• if x_{n-1} or x_n true, choose y_i true for all i

- if x_i true for $2 < i < n-1$, choose y_j true $\forall j \leq i-2$ and false $\forall j \geq i-1$ ✓

($\Leftarrow$) if second expression is true must have at least one x_i true: if not, then by induction all y_i true contradicting last conjunct.

So: given SAT problem $A_i C_i$ can translate to 3-SAT prob: replace every conjunct $x_1 \vee \dots \vee x_n$ w/ corresponding expression above; new 3-SAT prob is sat'able iff orig is; translation takes poly time ✓

- Also: many natural graph-theoretic problems are NP complete.

<u>Problem</u> (CLIQUE) Given a (finite) graph G and $n \in \mathbb{N}$ decide if G contains a copy of K_n (i.e. if there are n distinct vertices in G that are pairwise connected by an edge)

<u>Prop'n</u> CLIQUE is NP-complete.

- Clear that CLIQUE is in NP (why?)
- We reduce 3-SAT to CLIQUE