NP-Completeness

Thm [Cook-Levin] SAT is NP-complete

Thm [More convenient (Cook-Levin)] 3SAT is NP-complete

\[ SAT = \{ \phi \mid \phi \text{ is a satisfiable Boolean formula} \} \]
\[ 3SAT = \{ \phi \mid \phi \text{ is a satisfiable Boolean 3CNF} \} \]
\[ \phi \text{ is in } 3CNF \text{ if it is an AND of clauses, each clause 3 literals with OR} \]

Proof of Cook-Levin: To show that SAT is NP-complete

1. SAT \in NP
2. SAT is NP-hard, i.e. \forall L \in NP, L \leq_p SAT

\[ SAT \in NP \]

NIM for SAT:

- On input \( \phi(x_1, \ldots, x_n) \)
  1. Non-deterministically pick a truth assignment \( a_1, \ldots, a_n \)
  2. Evaluate \( \phi \) on this assignment, accept if \( \phi(a_1, \ldots, a_n) \)

Analysis:

Runtime: 1. Linear/poly in \( n \), which is size of input
2. poly-time - recursively called on smaller parts

Correctness: if \( \phi \in SAT \), then some branch of comp. will pick
a sat assignment, so will accept.
if \( \phi \not\in SAT \), no assignment will satisfy, so no branch will accept

\[ SAT \text{ is NP-hard:} \]

Let \( L \in NP \), let \( N \) be a NIM deciding \( L \), runtime \( p(n) \)

\[ we \ L \leq \ \exists \ \phi \text{, computable from } w \text{ in poly-time, is satisfiable} \]

we \( L \leq \exists \) an accepting computational history of \( N \) on input \( w \)

(accepting computational history):

- Since \( N \) runs in poly-time, it can use
  start config: \( \# q_0 w, u_1 \ldots w_n \# v \# \) at most \( p(n) \) spaces, which we will
  account for at the start of our CH.
  \[ \text{im config } \# S, \ldots, q_9, u_1, \ldots, w_n, \# \text{ last } \]
  - This must be a valid window
  \[ \text{im config } \# S, \ldots, q_9, u_1, \ldots, w_n, \# \text{ last } \]
  - w-1 transition in \( N \)

\[ c \leq p(n) \Rightarrow \text{ accept} \]

- Must have last configuration that accepts
Equivallently, $N$ has an accepting tableau if there is a table that encodes $(H)$ on input $w$:

A valid accepting tableau is a $(p(n)+1 \times p(n)+3)$ table where:

1. Each cell contains either $\#$, $q_0 \#$, $a \in \Gamma$ (tape alphabet)
2. First row corresponds to start configuration of $N$ on $w$.
3. Each row $i+1$ follows from row $i$ ($0 \leq i \leq p(n)-1$) according to a transition of $N$.
4. $\varnothing$ accepts appears somewhere.

We say $a$ valid accepting tableau of $N$ on $w$.

Formula $0$:

Variables: $x_{i,j,s}$ where $i$ is a row, $j$ is a column

$s \in \{\#, U, Q, U, R\}$

$x_{i,j,s} = T$ means $s$ in cell $i,j$ of the tableau

$0 = \varnothing_{\text{cell}} \land \varnothing_{\text{start}} \land \varnothing_{\text{move}} \land \varnothing_{\text{accept}}$

\[
\varnothing_{\text{cell}} = \bigwedge_{i,j} \varnothing_{ij}
\]

\[
\varnothing_{ij} = \bigvee_{s \in \{\#, U, Q, U, R\}} x_{i,j,s} \land \bigwedge_{s_1 \neq s_2} \neg x_{i,j,s_1} \land x_{i,j,s_2}
\]

Cell $i,j$ contains some $s$ if cell $i,j$ cannot contain both $s_i$ and $s_j$ for all $s_i \neq s_j$. 

\[
\text{cell } i,j \text{ contains some } s \\
\text{cell } i,j \text{ cannot contain both } s_i \text{ and } s_j \text{ for all } s_i \neq s_j
\]
Start = \bigwedge x_{0,1}, x_0,2, x_0,3, \ldots \bigwedge x_n, x_{n+1}, \ldots \bigwedge x_{n+5}, x_{n+6}, \ldots \\
\text{hard-coding value for start of tableau}

\text{accept = } \bigvee x, i,j, \text{accept} \quad \text{variable with accept appears in tableau}

\text{for move: should check every 2x3 window and make sure it is legal}
\text{if all true inputs in window: }
\begin{array}{ccc}
  a_1 & a_2 & a_3 \\
  a_2 & &
\end{array}
\text{if state in window: }
\begin{array}{ccc}
  a_1 & a_2 & a_3 \\
  & a_2 &
\end{array}

\text{for move: }
\bigwedge x, i,j, a \bigwedge x_{i,j+1}, b \bigwedge x_{i,j+1}, e \bigwedge x_{i+1,j}, c \bigwedge x_{i+1,j+1}, d

\begin{array}{llll}
  a & b & c & d \\
  e & f &
\end{array}
\text{legal windows}

\text{at each position check if the window belongs to one of the valid windows}
\text{valid windows are a finite set that can be calculated using description of N}
\text{reduction is valid, therefore SAT is NP-complete}

\text{now prove 3 SAT is NP-complete:}
\text{direct reduction from SAT to 3 SAT}
\text{- \text{start, accept, \text{can already in CNF}}}
\text{- \text{move is AND of ORs, which we can convert to CNF (converting DNF to CNF can make it exponentially larger,}
\text{but since legal windows is constant, exponentially}\n\text{larger of constant is still constant})}
\text{Next how to make CNF into 3 CNF:}
\text{- for clauses with less than 3 literals, just introduce new variables that have not appeared else where into clause}
- for clauses w/ more than 3 literals
  \[ x \lor v x_2 \lor v x_3 \lor v x_4 \]
  new var \( z \)
  \[(x_1 \lor v x_2 \lor z) \land (\neg v x_3 \lor v x_4)\]
- therefore the SAT reduction can be reduced into 3SAT in \( \text{poly} \), so 3SAT is also \( \text{NP-complete} \)
- \( \text{CLIQUE} \) is \( \text{NP-complete} \)
- in \( \text{NP} \)
- \( 3\text{SAT} \leq_p \text{CLIQUE} \)
  given a 3CNF \( (y_{1,1} \lor y_{1,2} \lor y_{1,3}) \land (y_{2,1} \land \ldots) \land (y_{m,1} \lor y_{m,2} \lor y_{m,3}) \)
  create a graph
  \[
  y_{1,1} \quad y_{1,2} \quad y_{1,3} \\
  y_{m,1} \quad y_{2,1} \\
  y_{m,2} \quad y_{2,2} \\
  y_{m,3} \quad y_{2,3} \\
  \]
- we want to make it so a satisfying assignment encodes a clique
- create edge \((y_{i,j}, y_{i',j'})\) if \( y_{i,j} \lor y_{i',j'} \)
- connect all literals that don't contradict each other