Lecture 25
As a supplement to Paul Beame’s guest lecture, here are a few slides of mine on roughly the same topics. Again, this won’t be exactly the same as what he did or as what’s in the book, but hopefully another perspective will help clarify it all.
Boolean Circuits

Directed acyclic graph
Vertices = Boolean logic gates ($\land$, $\lor$, $\neg$, …)
Multiple input bits ($x_1$, $x_2$, …)
Single output bit ($w$)
Gate values as expected (e.g. by induction on depth to $x_i$'s)
Boolean Circuits

Two Problems:

Circuit Value: given a circuit and an assignment of values to its inputs, is its output = 1?

Circuit SAT: given a circuit, is there an assignment of values to its inputs such that output = 1?
Boolean Circuits and Complexity

Two Problems:

Circuit Value: given a circuit and an assignment of values to its inputs, is its output = 1?

Circuit SAT: given a circuit, is there an assignment of values to its inputs such that output = 1?

Complexity:

- Circuit Value Problem is in P
- Circuit SAT Problem is in NP

Given implementation of computers via Boolean circuits, it may be unsurprising that they are complete in P/NP, resp.
∀ L ∈ P, L ≤_p CVP

Let M be a 1-tape, poly time TM. WLOG M accepts at left end of tape. “History” of M on input x:

Every cell in tableau is a simple, discrete function of 3 above it, e.g., if \( \delta(q,c) = (q',e,-1) \):

- \( q_{accept} \)
- \( T = n^k \)
- \( T = n^k \)
- \( q_{accept} \)

Bool encoding of cell content; fixed circuit computes new cell; replicate it across tableau
For $q \in Q, a \in \Gamma, 1 \leq i, j \leq T$, let

$$\text{state}(q,i,j) = 1 \text{ if } M \text{ in state } q \text{ at time } i \text{ w/ head in tape cell } j,$$

and

$$\text{letter}(a,i,j) = 1 \text{ if tape cell } j \text{ holds letter } a \text{ at time } i.$$ 

$$\text{writes}(i,j) = \bigvee_{q \in Q} \text{state}(q,i,j)$$

$$\text{letter}(b,i,j) = (\neg \text{writes}(i,j) \land b_{i-1,j}) \lor \left( \text{writes}(i,j) \land \bigvee_{(q,a)} \text{state}(q,i-1,j) \land \text{letter}(a,i-1,j) \right)$$

where the “or” is over $\{(q,a) \mid (-,b,-) = \delta(q,a)\}$

$$\text{state}(p,i,j) = \bigvee_{(q,a,d)} \text{state}(q,i-1,j-d) \land \text{letter}(a,i-1,j-d),$$

where the “or” is over $\{(q,a,d) \mid (p,-,d) = \delta(q,a)\}$, $d = \pm 1$

Row 0: initial config; columns -1,T+1: all false

Output: $\text{state}(q_{\text{accept}},T,1)$

Again, not exactly the version in the book, but close in spirit…
Result is something vaguely like this:
Similarly: $\forall L \in \text{NP}, L \leq_p \text{Circuit-SAT}$

Let $M$ be a 1-tape, poly time NTM. WLOG $M$ accepts at left end of tape.

“History” of $M$ on input $x$:

Every cell in tableau is a simple, discrete function of 3 above it, plus 1 ND choice bit;

e.g., if $(q', e, L) \in \delta(q, c)$:

- Bool encoding of cell content; fixed circuit computes new cell; replicate it across tableau
Some Details

Additionally, assume NTM has only 2 nondet choices at each step. For $q \in Q$, $a \in \Gamma$, $1 \leq i, j \leq T$, state($q, i, j$), letter($a, i, j$) as before. Let

\[
\text{choice}(i) = 0/1 \text{ define which ND choice } M \text{ makes at step } i
\]

Then, letter() and state() circuits change to incl choice, e.g.:

\[
\text{state}(p, i, j) = \neg\text{choice}(i-1) \land (\bigvee_{(q,a,d)} \text{state}(q, i-1, j-d) \land \text{letter}(a, i-1, j-d)) \lor
\]

\[
\text{choice}(i-1) \land (\bigvee_{(q',a',d')} \text{state}(q', i-1, j-d') \land \text{letter}(a', i-1, j-d'))
\]

where the “ors” are over

\[
\{(q,a,d) \mid (p,-,d) = \delta(q, a, \text{choice}=0)\},
\]
\[
\{(q',a',d') \mid (p,-,d') = \delta(q', a', \text{choice}=1)\}, \quad d = \pm 1
\]

AND

$TM$ input $\rightarrow$ circuit constants;
$circuit$ inputs are the choice bits;
$circuit$ is $satisfiable$ iff $\exists$ seq of choices s.t. $NTM$ accepts
Correctness

Poly time reduction:

Given $\delta$, key subcircuit is fixed, size $O(1)$. Calculate $n = \text{input length}$, $T = n^k$. Circuit has $O(T^2) = O(n^{2k})$ copies of that subcircuit, (plus some small tweaks at boundaries).

Circuit exactly reflects $M$’s computation, given the choice sequence. So, if $M$ accepts input $x$, then there is a choice sequence s.t. circuit will output 1, i.e., the circuit is satisfiable. Conversely, if the circuit is satisfiable, then any satisfying input constitutes a choice sequence leading $M$ to accept $x$.

Thus, Circuit-SAT is NP-complete.
Circuit-SAT \( \leq_p 3\text{-SAT} \)

\[
(w_1 \iff (x_1 \land x_2)) \land (w_2 \iff (\neg w_1)) \land (w_3 \iff (w_2 \lor x_1)) \land w_3
\]

Replace with 3-CNF Equivalent:

\[
\neg (w_1 \iff (x_1 \land x_2)) = (x_1 \lor x_2 \lor \neg w_1) \land (x_1 \lor \neg x_2 \lor \neg w_1) \land (\neg x_1 \lor x_2 \lor \neg w_1) \land (\neg x_1 \lor \neg x_2 \lor w_1) \ldots
\]

Build truth table clause-by-clause vs whole formula, so \(n \times 2^3\) vs \(2^n\) rows
Correctness of “Circuit-SAT $\leq_p$ 3-SAT”

Summary of reduction: Given circuit, add variable for every gate’s value, build clause for each gate, satisfiable iff gate value variable is appropriate logical function of its input variables, convert each to CNF via standard truth-table construction. Output conjunction of all, plus output variable. 

*Note*: as usual, does not know whether circuit or formula are satisfiable or not; does not try to find satisfying assignment.

Correctness:

Show it’s poly time computable: A key point is that formula size is linear in circuit size; mapping basically straightforward; details omitted.

Show c in Circuit-SAT iff f(c) in SAT:

$(\Rightarrow)$ Given an assignment to $x_i$’s satisfying c, extend it to $w_i$’s by evaluating the circuit on $x_i$’s gate by gate. Show this satisfies f(c).

$(\Leftarrow)$ Given an assignment to $x_i$’s & $w_i$’s satisfying f(c), show $x_i$’s satisfy c (with gate values given by $w_i$’s).

Thus, 3-SAT is NP-complete.