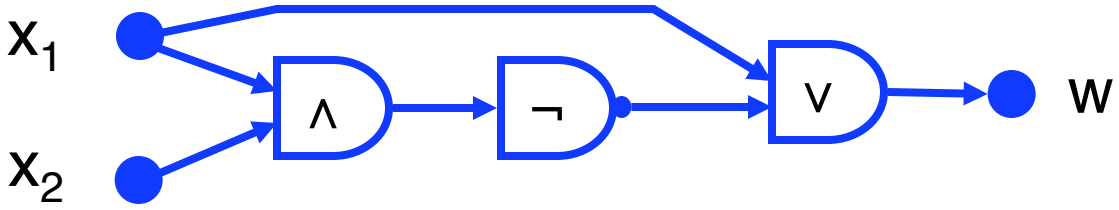
Lecture 25

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

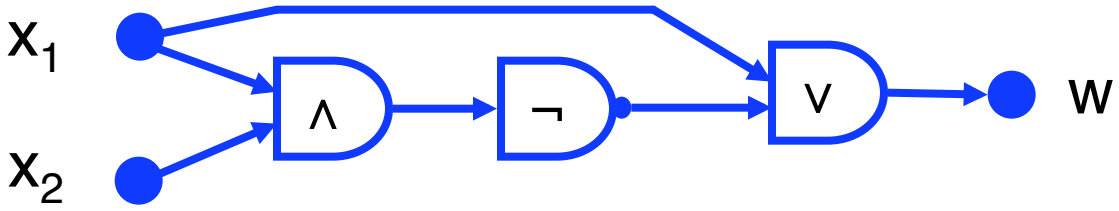
Vertices = Boolean logic gates (∧, ∨, ¬, …)

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?

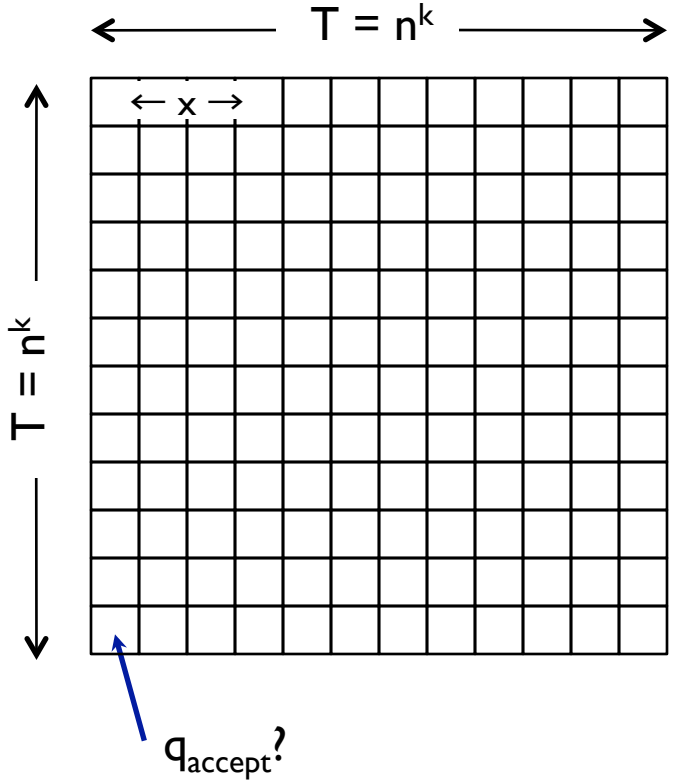
Complexity:

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

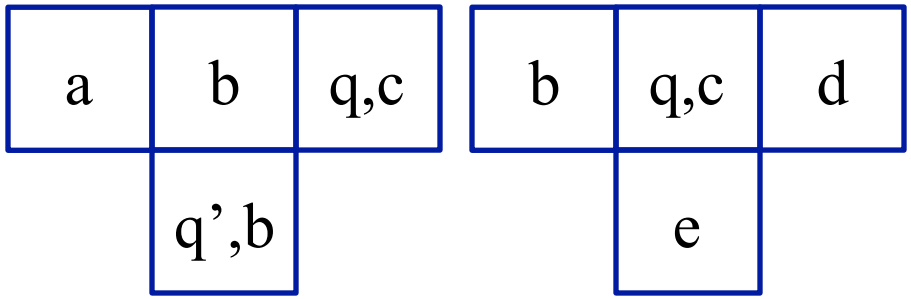
Given implementation of computers via Boolean circuits, it may be unsurprising that they are *complete* in P/NP, resp.

$\forall$ L $\in$ P, L $\leq_p$ 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 δ(q,c) = (q',e,-1):



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

Some Details

For $q \in Q$, $a \in \Gamma$, $1 \le i,j \le T$, let

state(q,i,j) = 1 if M in state q at time i w/ head in tape cell j, and
letter(a,i,j) = 1 if tape cell j holds letter a at time i.

$\text{writes}(i,j) = \bigvee_{q \in Q} \text{state}(q,i,j)$ — write cell i @ step j

$\text{letter}(b,i,j) = (\neg\text{writes}(i,j) \wedge b_{i-1,j}) \vee$ — no head, no change

$(\text{writes}(i,j) \wedge \bigvee_{(q,a)} \text{state}(q,i-1,j) \wedge \text{letter}(a,i-1,j))$ — "or" configs writing "b"

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

$\text{state}(p,i,j) = \bigvee_{(q,a,d)} \text{state}(q,i-1,j-d) \wedge \text{letter}(a,i-1,j-d)$, — "or" configs entering p

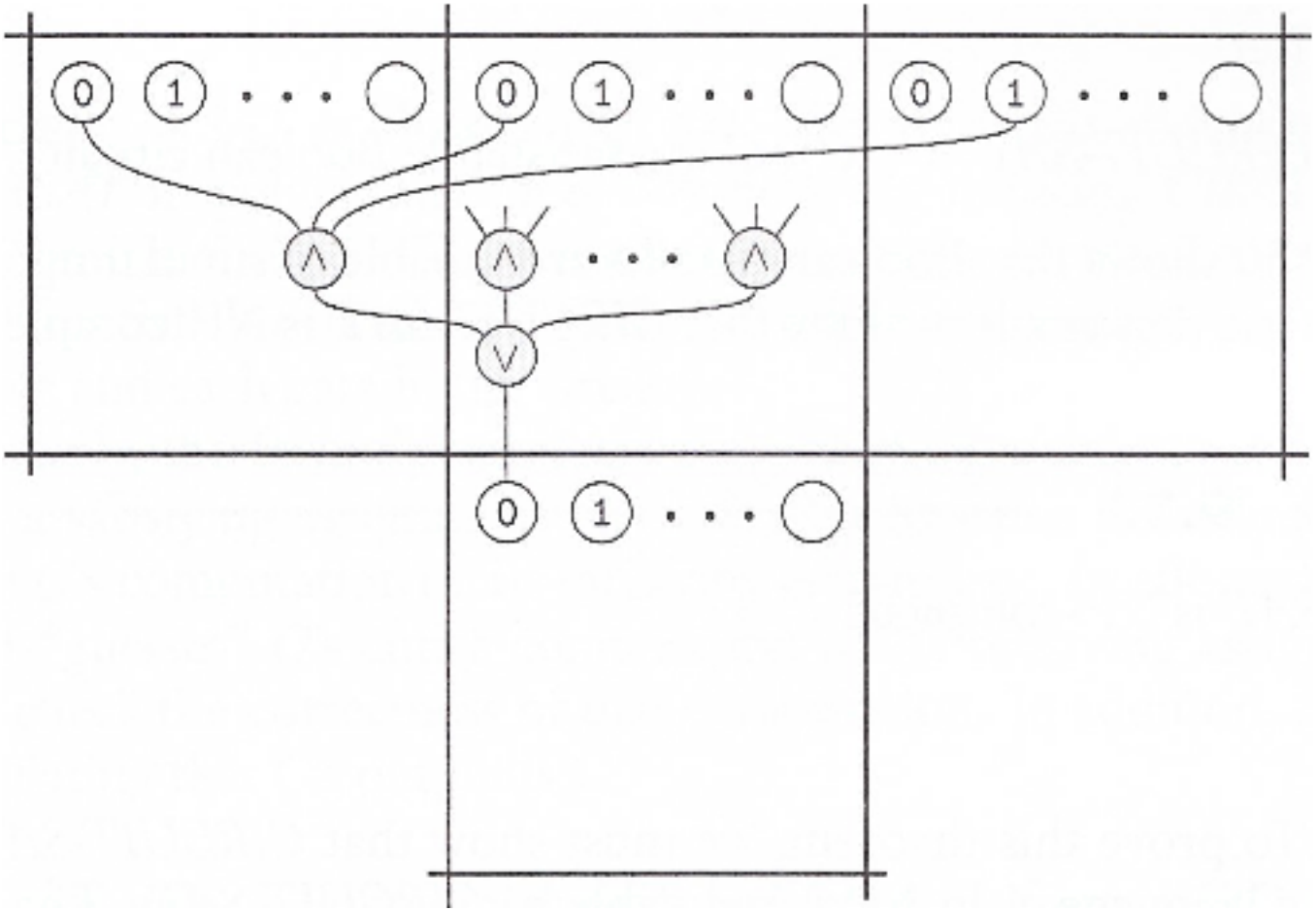
where the "or" is over $\{(q,a,d) \mid (p,-,d) = \delta(q,a)\}$, $d = \pm 1$

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

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

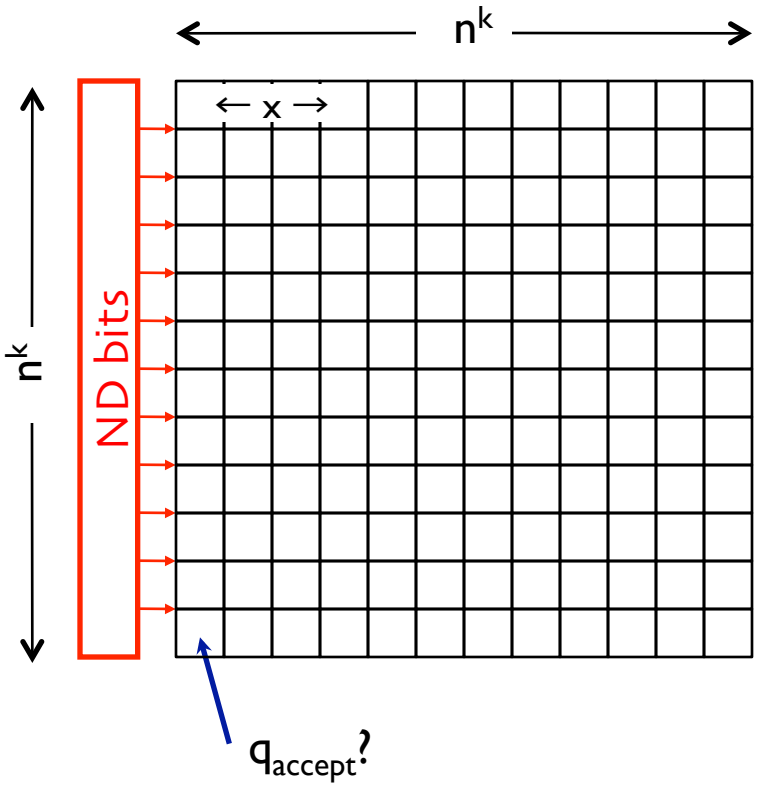
Again, not exactly the version in the book, but close in spirit…

Result is something vaguely like this:

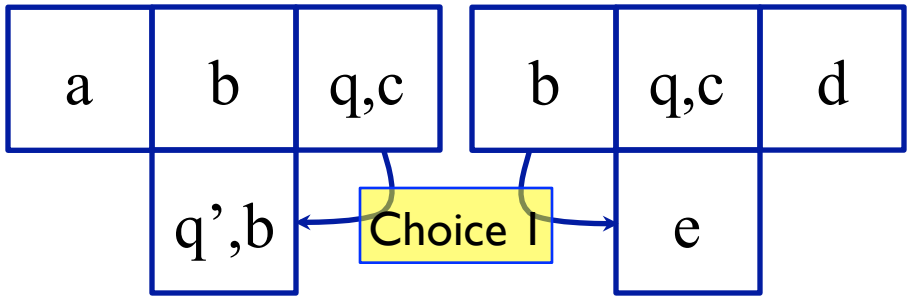


Similarly: $\forall L \in NP$, $L \leq_p$ Circuit-SAT

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 $q \in Q$, $a \in \Gamma$, $1 \le i,j \le T$, state(q,i,j), letter(a,i,j) as before. Let

choice(i) = 0/1 define which ND choice M makes at step i

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

$$\text{state}(p,i,j) = \neg\text{choice}(i\text{-}1) \wedge \left(\bigvee_{(q,a,d)} \text{state}(q,i\text{-}1,j\text{-}d) \wedge \text{letter}(a,i\text{-}1,j\text{-}d)\right) \vee$$

$$\text{choice}(i\text{-}1) \wedge \left(\bigvee_{(q',a',d')} \text{state}(q',i\text{-}1,j\text{-}d') \wedge \text{letter}(a',i\text{-}1,j\text{-}d')\right),$$

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 → circuit constants;
circuit inputs are the choice bits;
circuit is *satisfiable* iff ∃ seq of choices s.t. NTM accepts

Correctness

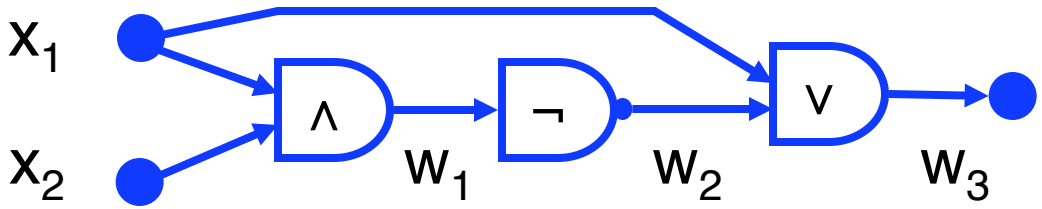
Poly time reduction:

Given δ, key subcircuit is fixed, size O(1). Calculate n = 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-SAT

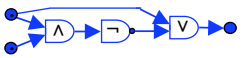


$$(w_1 \Leftrightarrow (x_1 \wedge x_2)) \wedge (w_2 \Leftrightarrow (\neg w_1)) \wedge (w_3 \Leftrightarrow (w_2 \vee x_1)) \wedge w_3$$

Replace with 3-CNF Equivalent:

¬ clause
↓
Truth Table
↓
DNF
↓
DeMorgan
↓
CNF

x_1	x_2	w_1	$x_1 \wedge x_2$	$\neg(w_1 \Leftrightarrow (x_1 \wedge x_2))$	
0	0	0	0	0	
0	0	1	0	1	$\leftarrow \neg x_1 \wedge \neg x_2 \wedge w_1$
0	1	0	0	0	
0	1	1	0	1	$\leftarrow \neg x_1 \wedge x_2 \wedge w_1$
1	0	0	0	0	
1	0	1	0	1	$\leftarrow x_1 \wedge \neg x_2 \wedge w_1$
1	1	0	1	1	$\leftarrow x_1 \wedge x_2 \wedge \neg w_1$
1	1	1	1	0	

f() = $(x_1 \vee x_2 \vee \neg w_1) \wedge (x_1 \vee \neg x_2 \vee \neg w_1) \wedge (\neg x_1 \vee x_2 \vee \neg w_1) \wedge (\neg x_1 \vee \neg x_2 \vee w_1) \ldots$

Build truth table clause-by-clause vs whole formula, so $n*2^3$ 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.