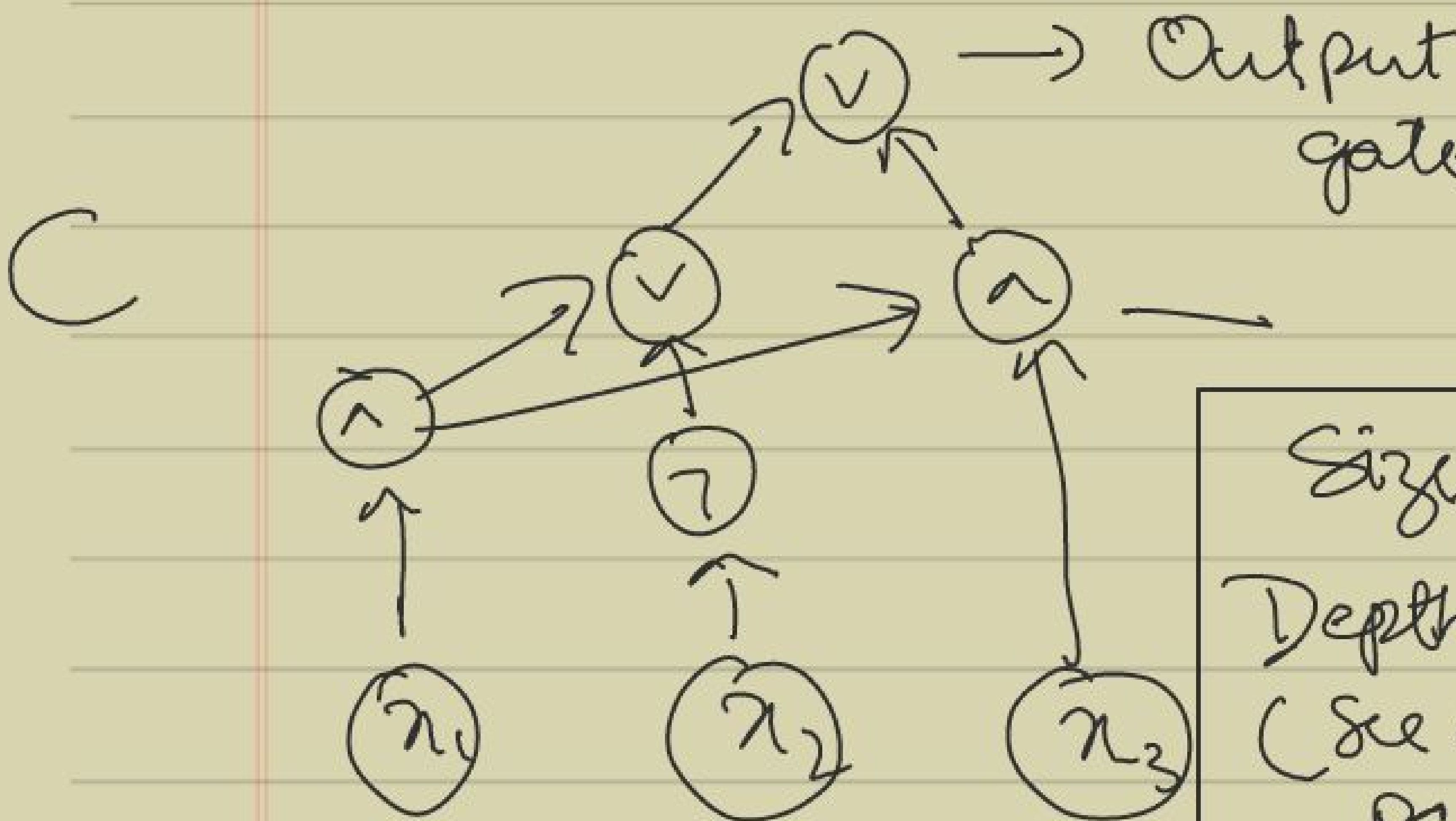
# Boolean Circuits & P/poly

Want to show: SAT has no poly-time algorithms.

→ SAT seems to be hard at each input length. Why not try to understand the most efficient way to solve SAT at each input length $n$ & show that this running time is not a polynomial function of $n$.?

→ Leads to computational models that work with inputs of a fixed length.

## Boolean circuits




Size = 8

Depth = 3 (see next page)

- Directed acyclic graphs (DAG)
- Sources labelled by variables.
- Internal node labelled by $\neg, \wedge, \vee$ & have either 1, 2, or 2 in-neighbours respectively. Called "gates".
- One marked output gate (can also have more)
- Computes $f : \{0,1\}^{n = \#\text{variables}} \longrightarrow \{0,1\}$

Compare with Boolean formulas:

→ Circuits are more general

→ A formula corresponds to a directed tree.

We think of a circuit as an algorithm computing a function on inputs of a fixed length.

Complexity of algorithm measured by

1. Size = # of vertices. (analogous to running time)
2. Depth = length of longest path from variable to output.

To talk about circuits for a language, we need one for each input length.

$\{C_n\}_{n\in\mathbb{N}}$ - family of circuits

($C_n$ depends on $n$ inputs).

Say $\{C_n\}_{n\in\mathbb{N}}$ has size $T(n)$ if

$|C_n| \leq T(n)$ for each $n$.

---

$\{C_n\}_{n\in\mathbb{N}}$ decides a language $L \subseteq \{0,1\}^*$

if for any $x \in \{0,1\}^n$

$$x \in L \iff C_n(x) = 1.$$

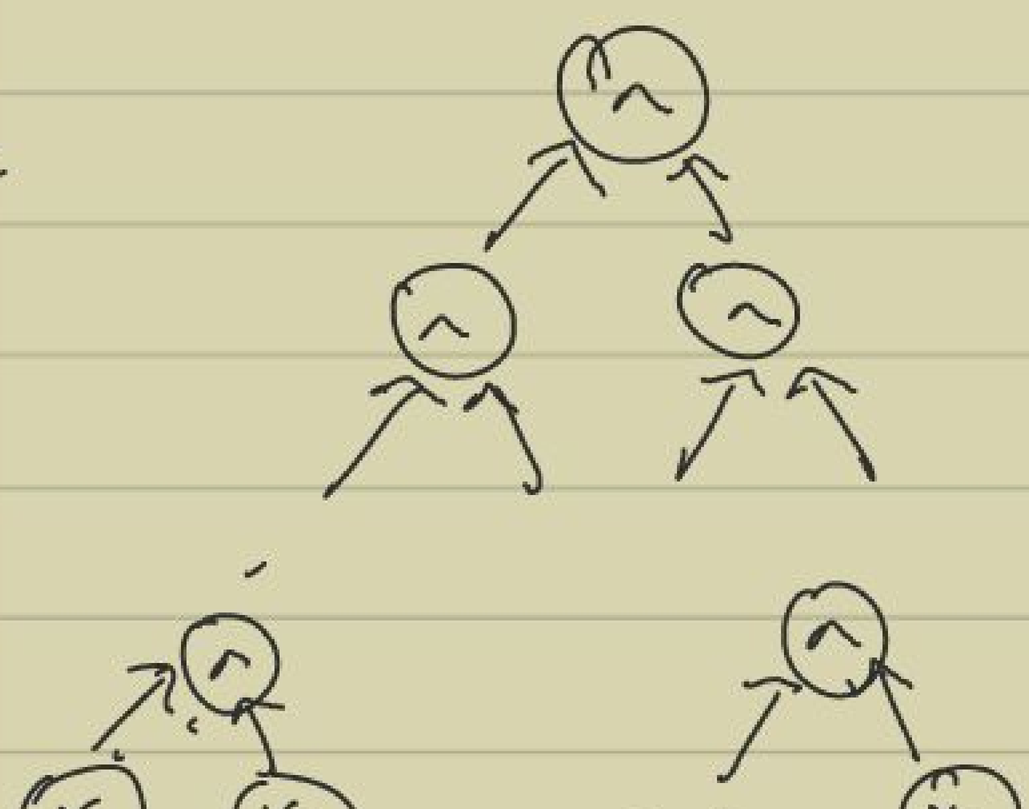
$\mathrm{SIZE}(T(n)) = \{L \mid L$ decided by a circuit family of size $T(n)\}$.

$$P/\text{poly} = \bigcup_c \text{SIZE}(n^c).$$

(i.e languages decided by a polynomial-sized circuit family)

Eg: $L = \{1^n \mid n \in \mathbb{N}\} \in P/\text{poly}$

$C_n =$




More generally, any unary language.

(see next page)

Another definition: TMs with advice

DTM augmented with an "advice string" that depends on the length of the input

Eg: $L$ a unary language i.e

$$L \subseteq \{1^n \mid n \in \mathbb{N}\}$$

So at each input length, $L$ contains either 0 or 1 string. Hence, given one bit of advice (does $1^n \in L$?) a DTM $M$ can decide $L$ in polynomial-time.

$L \in \mathrm{DTIME}(T(n))/a(n)$

(decidable by $T(n)$-time TMs with $a(n)$ bits of advice)

if there exist $y_n \in \{0,1\}^{a(n)}$ for each $n$ s.t.

$x \in L \iff M(x, y_n) = 1$

where $M$ runs in time $T(n)$.

---

$L$ a unary language $\Rightarrow$

$L \in \mathrm{DTIME}(O(n))/1$

---

Even includes some undecidable languages!

[Eg: Unary-Halt $= \{1^n \mid n$th TM halts on empty input$\}$ ]

Thm 1: $P/\text{poly} = \bigcup_{c,d} \text{DTIME}(n^c)/n^d$.

Pf ($\Rightarrow$) Assume $L \in P/\text{poly}$

Then $L$ is decided by a circuit family $\{C_n\}_{n \in \mathbb{N}}$ where $|C_n| \le \text{poly}(n)$

Then we can also decide $L$ with $\text{poly}(n)$ bits of advice "encoding" the circuit $C_n$

describe the circuit in a reasonable way.

The machine $M$ on input $(x, y_n)$ just runs $C_n$ on input $x$, which can be done in polynomial time

($\Leftarrow$) Say L decided by poly-time DTM M with poly(n) bits of advice.

Want: a circuit $C_n$ on inputs of length $n$.

Idea: Go back to proof of Cook-Levin.

We can assume M is an oblivious k-tape TM. On input $(x, y_n)$ ↳ advice

the machine M produces a sequence of snapshots (current state & symbol being read)

$z_1, \ldots, z_{poly(n)}$.

Each $z_i$ is a constant number of bits & can be computed from

$z_{i_1}, \ldots, z_{i_k}$ where
$i_1, \ldots, i_k < i$ are the previous
time-steps where $M$ scanned
the same location of the tape in
the $k$ tapes.

The dependence of $z_i$ on $z_{i_1}, \ldots, z_{i_k}$
is determined by the rules of $M$
& we can write an $O(1)$-sized
circuit that implements these
rules.

Thus, we can construct a circuit
$C_n$ that reconstructs all the snap-
-shots & accepts if & only if the
final snapshot is accepting. $\square$

Corollary 2: $P \subseteq P/poly$.

In fact, if $L \in P$, then $L$ is decidable by a circuit family $\{C_n\}_{n \in \mathbb{N}}$ where $C_n$ can be constructed by an algorithm in poly(n) time. (The above proof shows this. The entire proof is algorithmic, except for the constr-uction of the advice string $y_n$.)

→ Such circuit families are called P-uniform

This gives us a new approach to

$P$ vs $NP$.

Show that some problem in $NP$ does not have polynomial-sized circuits

---

Is this feasible?

After all, $P/\text{poly}$ contains even some undecidable languages!

But we believe it is true that

$NP \not\subseteq P/\text{poly}$ because ...

Thm 2 [Karp-Lipton thm]:

If $NP \subseteq P/\text{poly}$, then $PH = \Sigma_2^P$

Proof: We will show that if $NP \subseteq P/poly$, then $\Pi_2^P = \Sigma_2^P$

Sufficient to show: $\Pi_2^P \subseteq \Sigma_2^P$
(ex.)

Say $L \in \Pi_2^P$. There is a poly-time DTM $M$ s.t.

$$x \in L \iff \forall y_1\, \exists y_2\; M(x, y_1, y_2) = 1$$

strings of length poly(n)

Define:

$$L' = \{ (x, y_1) \mid \exists y_2\, M(x, y_1, y_2) = 1 \}$$

Obs: (1) $L' \in NP$

(2) $L = \{ x \mid \forall y_1\; (x, y_1) \in L' \}$

Since $L' \in NP$, any instance of $L'$ can be the reduced in poly-time to an instance of SAT of length $m = \text{poly}(n)$

Idea 1: Use the first certificate of the $\Sigma_2^P$ algorithm to get a circuit that solves SAT on inputs of length $m$.

DTM $M'$ to show $L \in \Sigma_2^P$:

$M'(x, y_0, y_1)$:

certificate hopefully encoding $C$ solving SAT on inputs of length $m$

(1) Reduce $(x, y_1) \overset{?}{\in} L'$ to checking $\varphi \overset{?}{\in} SAT$

(2) Check that the circuit $C$ outputs 1 on $\varphi$. If so accept & o/w reject.

**Problem:** What if $y_0$ is not a circuit solving SAT correctly?

**Eg:** $y_0$ encodes a circuit $C$ that accepts everything! Then we also accept $x \notin L$.

**Fix:** Use $y_0$ to get a circuit $C$ that outputs a satisfying assignment of a satisfiable CNF.

**Ex:** If $NP \subseteq P/poly$, then there is a multi-output poly-sized circuit family that outputs a satisfying assignment of any satisfiable CNF $\varphi$.

With the fix, we can no longer be fooled into accepting when we should reject.

So final (correct) version of $M'$:

$M'(x, y_0, y_1)$

1. Reduce $(x, y_1) \overset{?}{\in} L'$ to checking $\varphi \overset{?}{\in} SAT$
2. Check that circuit $C$ encoded by $y_0$ outputs a satisfying assignment of $\varphi$. If so, accept & otherwise reject.

Quick proof of correctness:

$x \in L \Rightarrow \forall y_1\ (x, y_1) \in L'$

$\Rightarrow \exists y_0\ \forall y_1\ M'(x, y_0, y_1) = 1$

the "correct" circuit $C$

Conversely,

$x \notin L \Rightarrow \exists y_1\ (x, y_1) \notin L'$

$\Rightarrow \exists y_1\ \varphi$ is not satisfiable

$\Rightarrow \forall y_0\ \exists y_1\ M'(x, y_0, y_1) = 0$

(because $M'$ does not get a satisfying assignment to $\varphi$).

# Shannon's lower bound

We expect that $NP \not\subseteq P/poly$ but so far we don't know

$$PSPACE \not\subseteq P/poly$$

or even $EXP \not\subseteq P/poly$ [Note: we know $EXP \not\subseteq P$!]

or even $NEXP \not\subseteq P/poly$

Given this, why should we expect to prove "circuit lower bounds"?

i.e a statement of the form

"$L$ does not have polynomial-sized circuits."

Theorem 3 [Shannon]: For any $n$, there exist Boolean functions $f: \{0,1\}^n \to \{0,1\}$ with no circuits of size $2^n/10n$.

Proof: Counting Argument

Count # of Boolean functions $= N$

Count # of Circuits of size $s = M$

If $M < N$, there is a function with no circuit of size $s$.

$N$ is easy: $2^n$ inputs in $\{0,1\}^n$

2 choices per input

$\Rightarrow N = 2^{2^n}$. } Doubly exponential!

M is only slightly harder:

We can construct a circuit of size $\leq s$ by adding the vertices in topological order:

→ First $n$ vertices are variables $x_1, \ldots, x_n$.

→ Every subsequent gate is $\neg$, $\wedge$ or $\vee$ & is connected to 1 or 2 of the previous gates

Number of choices $= \underbrace{s}_{\text{for } \neg} + \underbrace{s^2}_{\text{for } \wedge} + \underbrace{s^2}_{\text{for } \vee}$

$\leq 3s^2$

→ Number of choices for output gate $\leq s$

Thus, $M \le (3s^2)^s = 2^{s\log(3s^2)}$

Check: 'if $s < 2^n/10n$', $M < \sqrt{N}$

Thus, there is a function $f: \{0,1\}^n \to \{0,1\}$ that has no circuits of size $< 2^n/10n$. $\blacksquare$

In fact, the proof shows that most functions have no small circuits!

Can we find one?

"Like finding hay in a haystack"

- Howard Karloff.

Closely related: Does Randomness help in computation? Next Week!