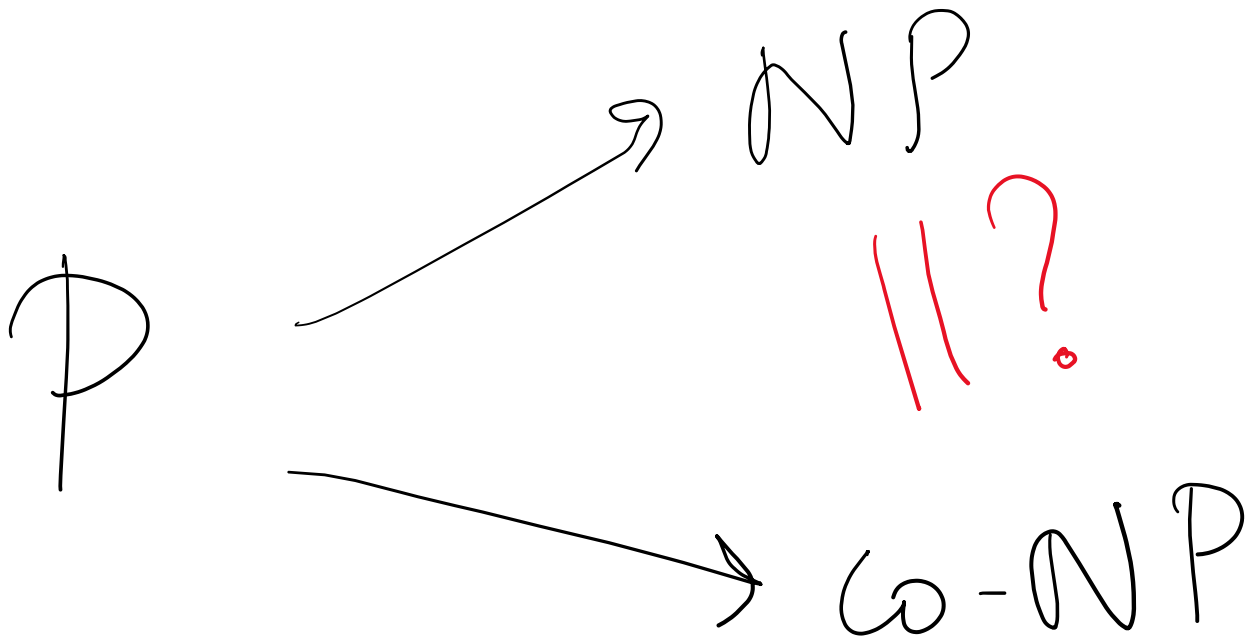
$\log CFL / SAC^1$ is closed under complementation

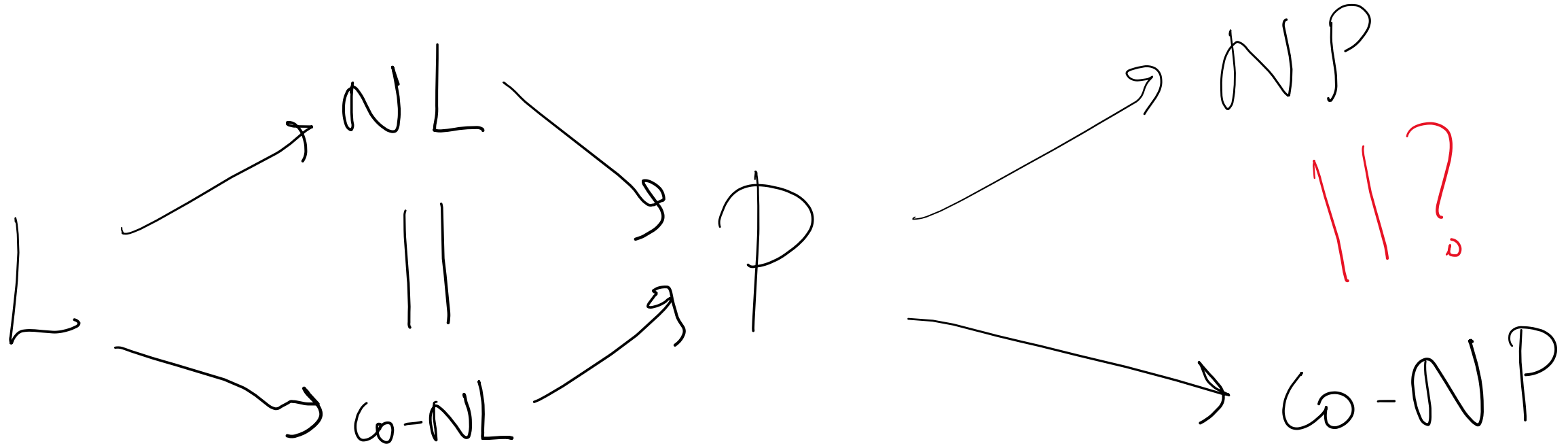
4/3/21

Complexity Classes



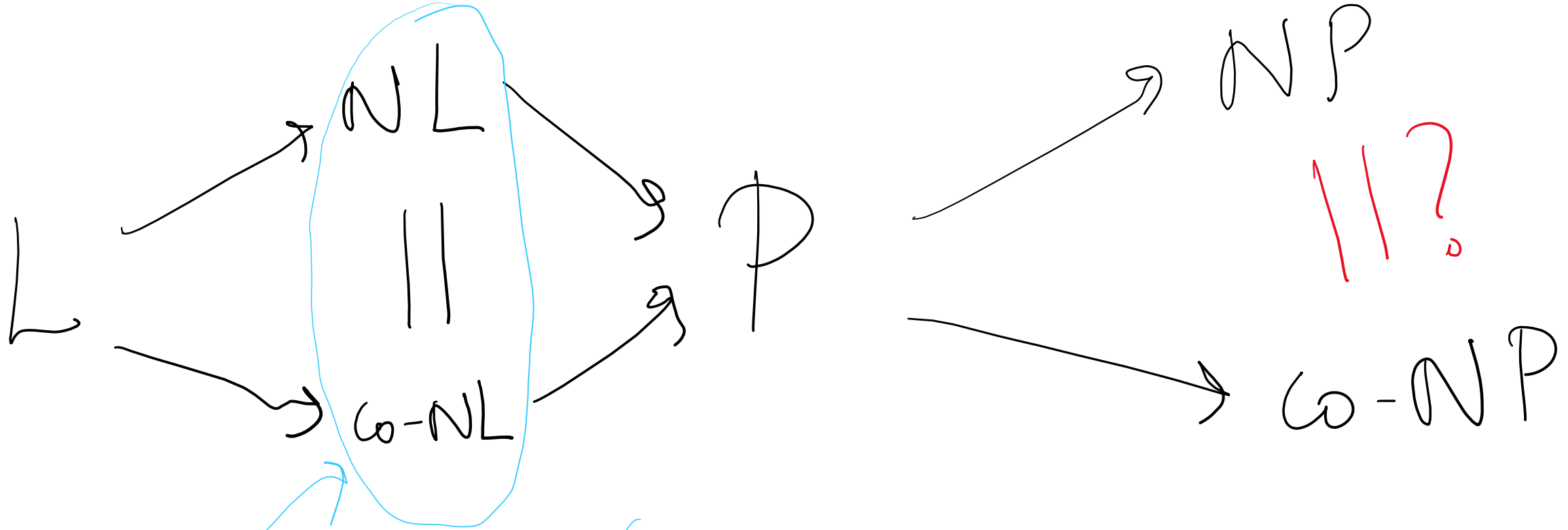
> "Complementation" an important theme in the study of non-det. complexity classes

Complexity Classes



> "Complementation" an important theme in the study of non-det. complexity classes

Complexity Classes



Surprising result (Immerman '88, Szelepcsényi '87)

(... detour) Circuits.

> Unlike TMs, combinatorial models of comp.

(... detour) Circuits.

- Unlike TMs, combinatorial models of comp.
- inherently 'non-uniform' (one for each n)

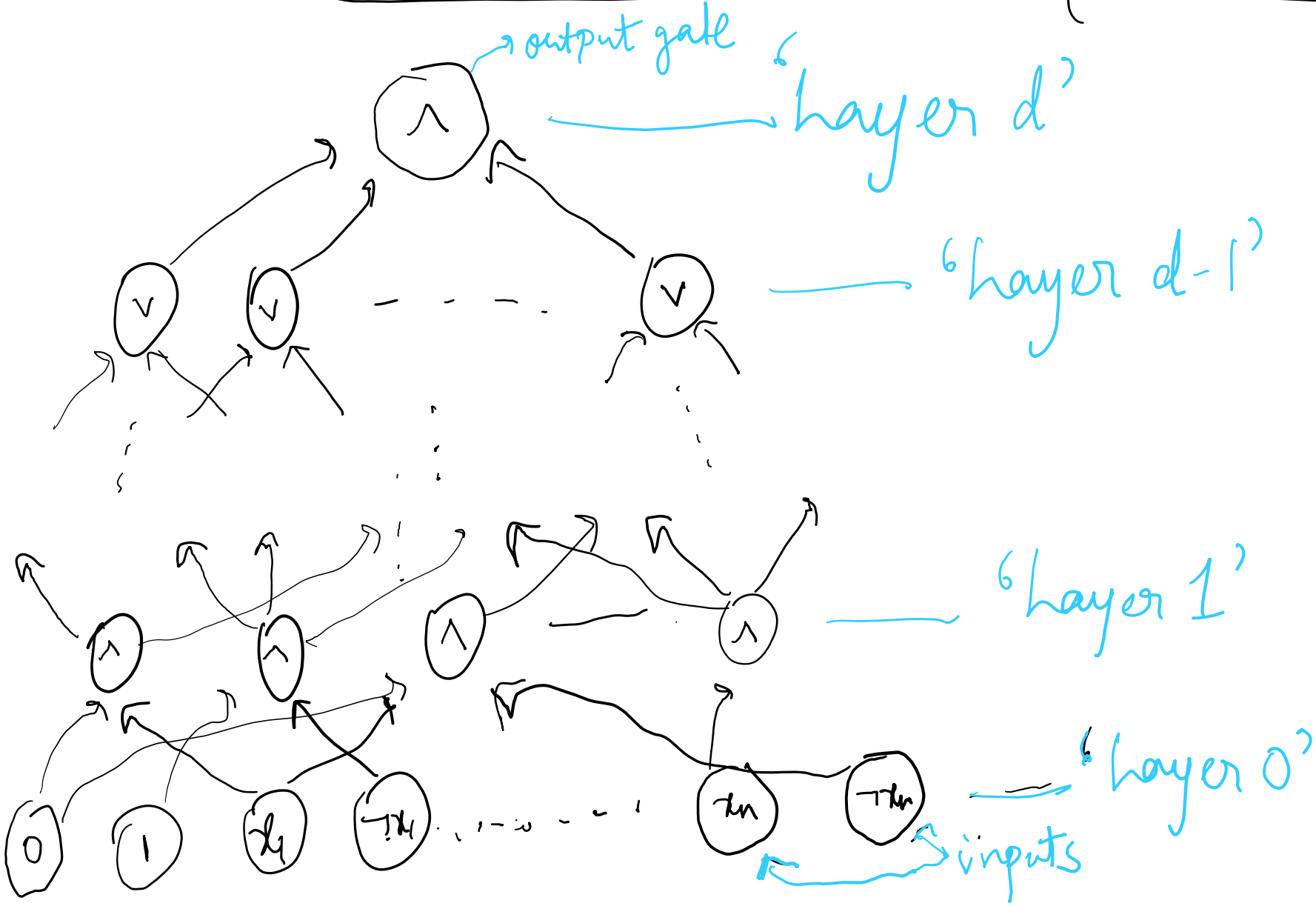
(... detour) Circuits.

> Unlike TMs, combinatorial models of comp.
> inherently 'non-uniform' (one for each n)
> makes them immensely powerful (for eg. HALT can be computed by (exp-sized) circuits)

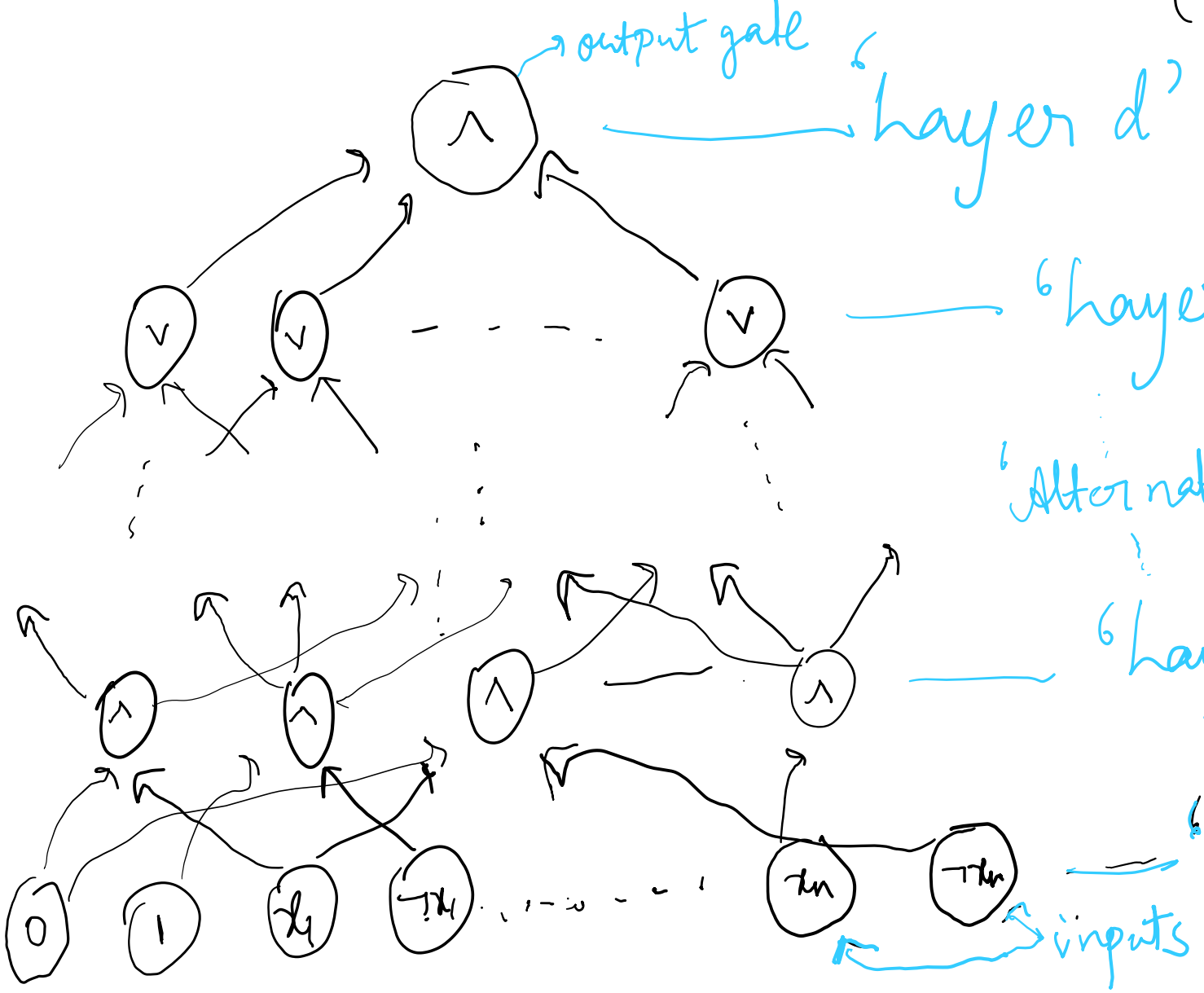
(... detour) Circuits.

> Unlike TMs, combinatorial models of comp.
> inherently 'non-uniform' (one for each n)
> makes them immensely powerful (for eg. HALT can be computed by (exp-sized) circuits)
> so makes sense to 'restrict' them in some way and see how their power compares with our beloved TM classes

(Boolean) Alternating Circuits



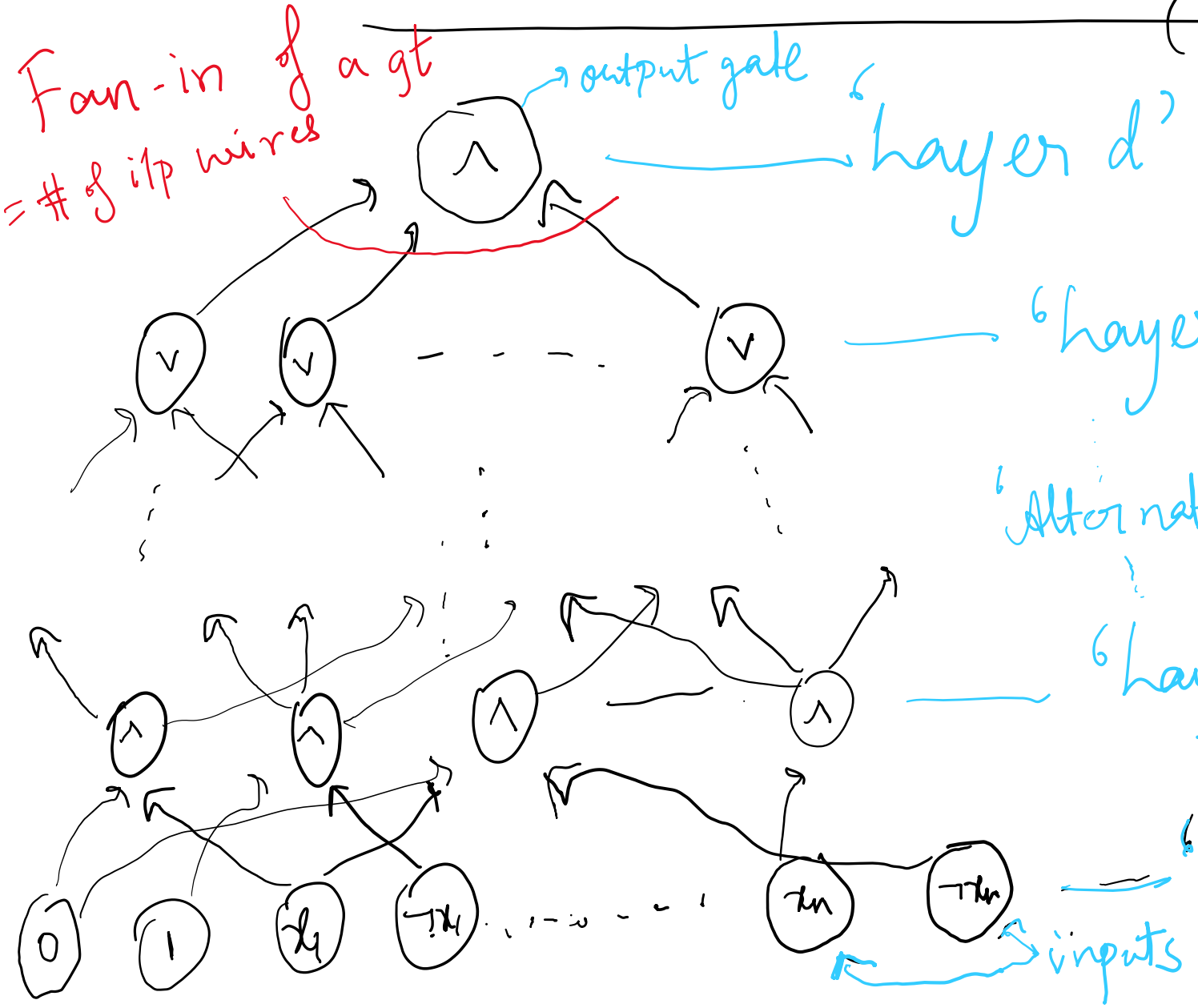
(Boolean) Alternating Circuits



- Each internal node labelled by $\wedge$ / $\vee$
- wires go b/w successive layers
- Two imp. parameters
  - size s = # nodes
  - depth d

$s = s(n)$, $d = d(n)$

(Boolean) Alternating Circuits



> Each internal node labelled by ∧ / ∨

> wires go b/w successive layers

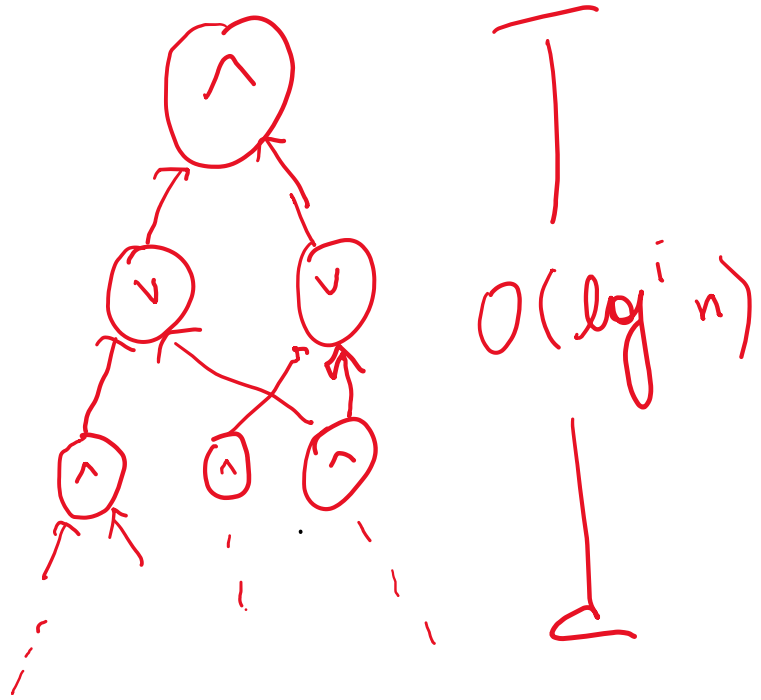
> ~~Two~~ Three imp. parameters

- size s = # nodes
- depth d

$s = s(n)$, $d = d(n)$

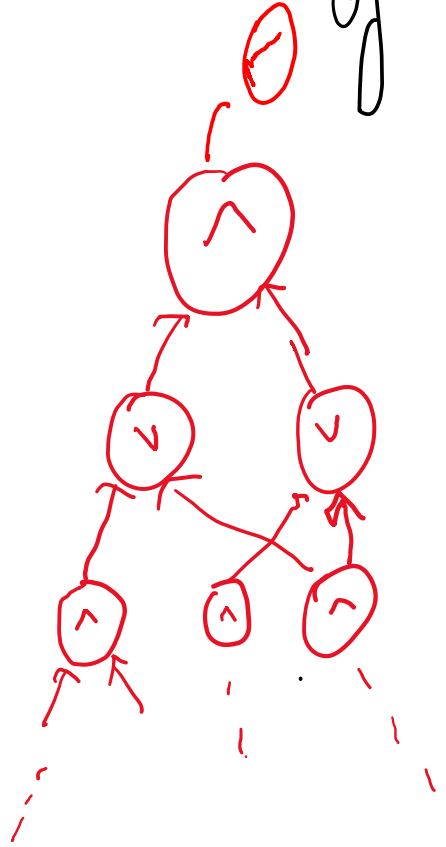
Circuit Classes

NC^i = Class of languages computed by circuits of poly(n) size, $O(\log^i n)$ depth, fan-in = 2.



Circuit Classes

NC^i = Class of languages computed by circuits of poly(n) size, $O(\log^i n)$ depth, fan-in = 2.



$O(\log^i n)$

We have

$$NC^0 \subset NC^1 \subset \cdots \subset NC = \bigcup NC^i$$

$$\cap$$

$$P$$

Circuit Classes

NC^i = Class of languages computed by circuits of $poly(n)$ size, $O(\log^i n)$ depth, fan-in = 2.

AC^i = Class of languages computed by circuits of $poly(n)$ size, $O(\log^i n)$ depth, unbounded fan-in

The NC Hierarchy

> clearly then, $NC^i \subseteq AC^i$.

The NC Hierarchy

- clearly then, $NC^i \subseteq AC^i$
- Also not too hard to see $AC^i \subseteq NC^{i+1}$

The NC Hierarchy

> clearly then, $NC^i \subseteq AC^i$

> also not too hard to see $AC^i \subseteq NC^{i+1}$

> moreover, neat relationship with our favourite classes:

$$NC^0 \subseteq AC^0 \subseteq NC^1 \subseteq L \subseteq NL \subseteq AC^1 \subseteq NC^2 \subseteq P$$

The NC Hierarchy

> clearly then, $NC^i \subseteq AC^i$.

> Also not too hard to see $AC^i \subseteq NC^{i+1}$

> Moreover, neat relationship with our favourite classes:

$$NC^0 \subseteq AC^0 \subseteq NC^1 \subseteq L \subseteq NL \subseteq AC^1 \subseteq NC^2 \subseteq P$$

$(\neq)$ $(\neq)$

The NC Hierarchy

- clearly then, $NC^i \subseteq AC^i$
- Also not too hard to see $AC^i \subseteq NC^{i+1}$
- Moreover, neat relationship with our favourite classes:

$$NC^0 \subseteq AC^0 \subseteq \boxed{NC^1 \subseteq L \subseteq NL \subseteq AC^1} \subseteq NC^2 \subseteq P$$

A very rich landscape of classes beneath P.

The Class $\log CFL$, finally

The Class $\log CFL$, finally

> Don't care!

The Class $\log CFL$, finally

> Don't care!

> Equivalence (Venkateswaran '87, Sudborough '78):

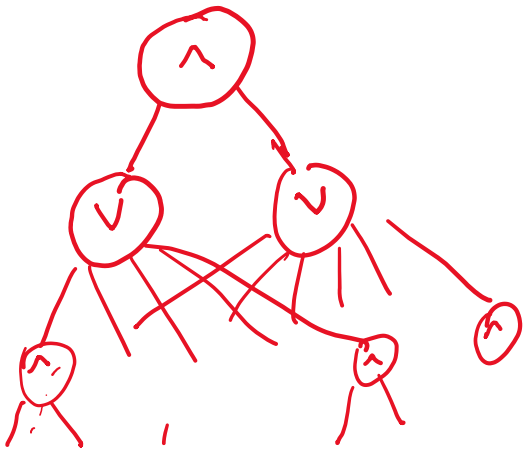
$\log CFL = SAC^1$ ("semi-unbounded Alternating")

The Class $\log CFL$, finally

> Don't care!

> Equivalence (Venkateswaran '87, Sudborough '78):

$$\log CFL = SAC^1$$ ("semi-unbounded Alternating")



poly(n) size

T

$O(\log n)$

$\wedge$ gates - fan-in = 2

$\vee$ gates - unrestricted

The Class $\log CFL$, finally

> Don't care!

> Equivalence (Venkateswaran '87, Sudborough '78):

$$\log CFL = SAC^1 = NAuxPDA(n^{O(1)}, \log n)$$

The Class $\log CFL$, finally

> Don't care!

> Equivalence (Venkateswaran '87, Sudborough '78):

$$\log CFL = SAC^1 = NAuxPDA(n^{O(1)}, \log n)$$

A certain non-det. class.

$\Rightarrow$ question of complementation is interesting!

The Class $\log CFL$, finally

> Don't care!

> Equivalence (Venkateswaran '87, Sudborough '78):

$$\log CFL = SAC^1 = NAuxPDA(n^{O(1)}, \log n)$$

Moreover, this class fits as follows:

[Borodin, Cook, Dymond, Ruzzo, Tompa '88]

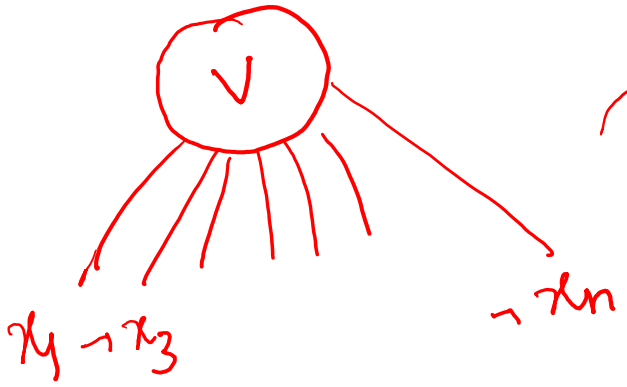
$$SAC^1 = co\text{-}SAC^1$$

[Borodin, Cook, Dymond, Ruzzo, Tompa '88]

$$SAC^1 = co\text{-}SAC^1$$

Given a semi-unbounded circuit C on n input vars, of poly-size, log-depth, construct another SAC^1 circuit D that computes $\neg C$.

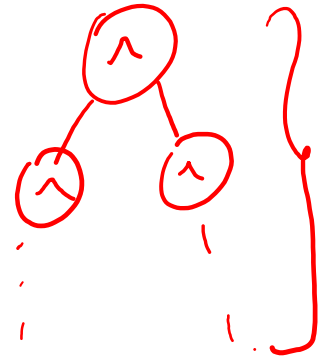
Note: this is not trivial! The simple application of De-Morgan Laws that works for AC^0 doesn't work here.



NOT of this

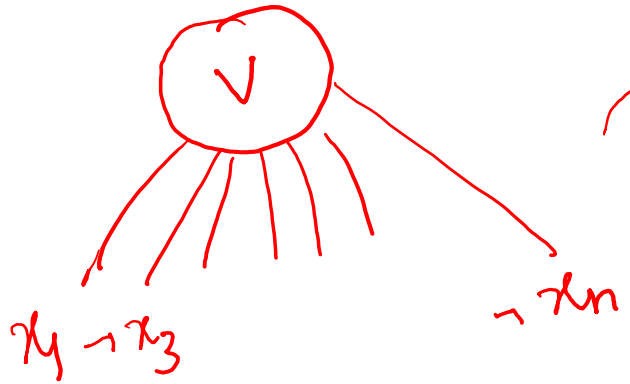
via DM becomes a large AND.

with only fan-in 2 AND gates.



requires log depth

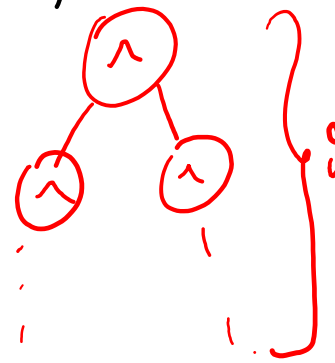
Note: this is not trivial! The simple application of De-Morgan Laws that works for AC^0 doesn't work here.



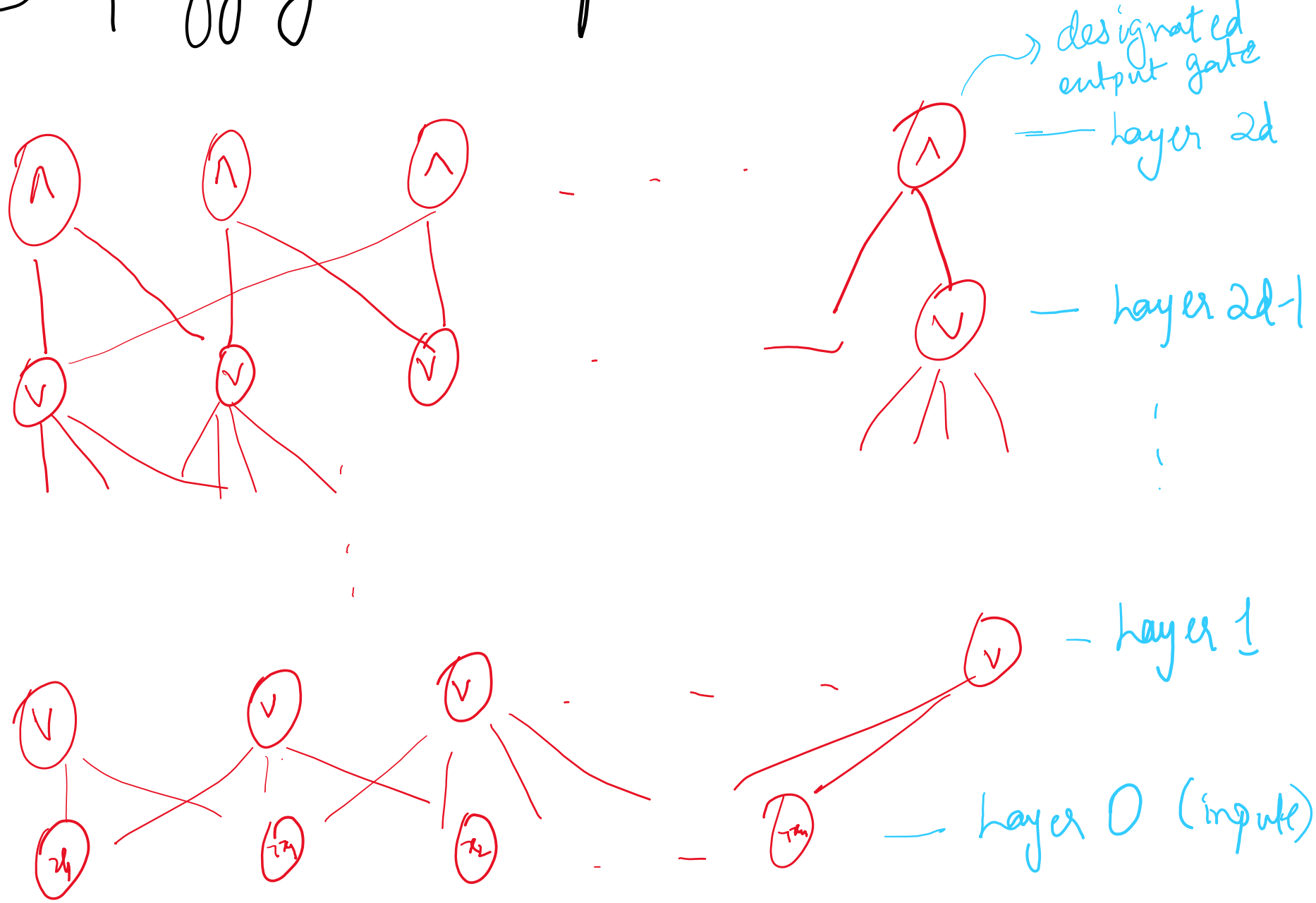
NOT of this via DM becomes a large AND.

But original circuit already has log-depth, this'll blow-up depth to $\log^2 n$.

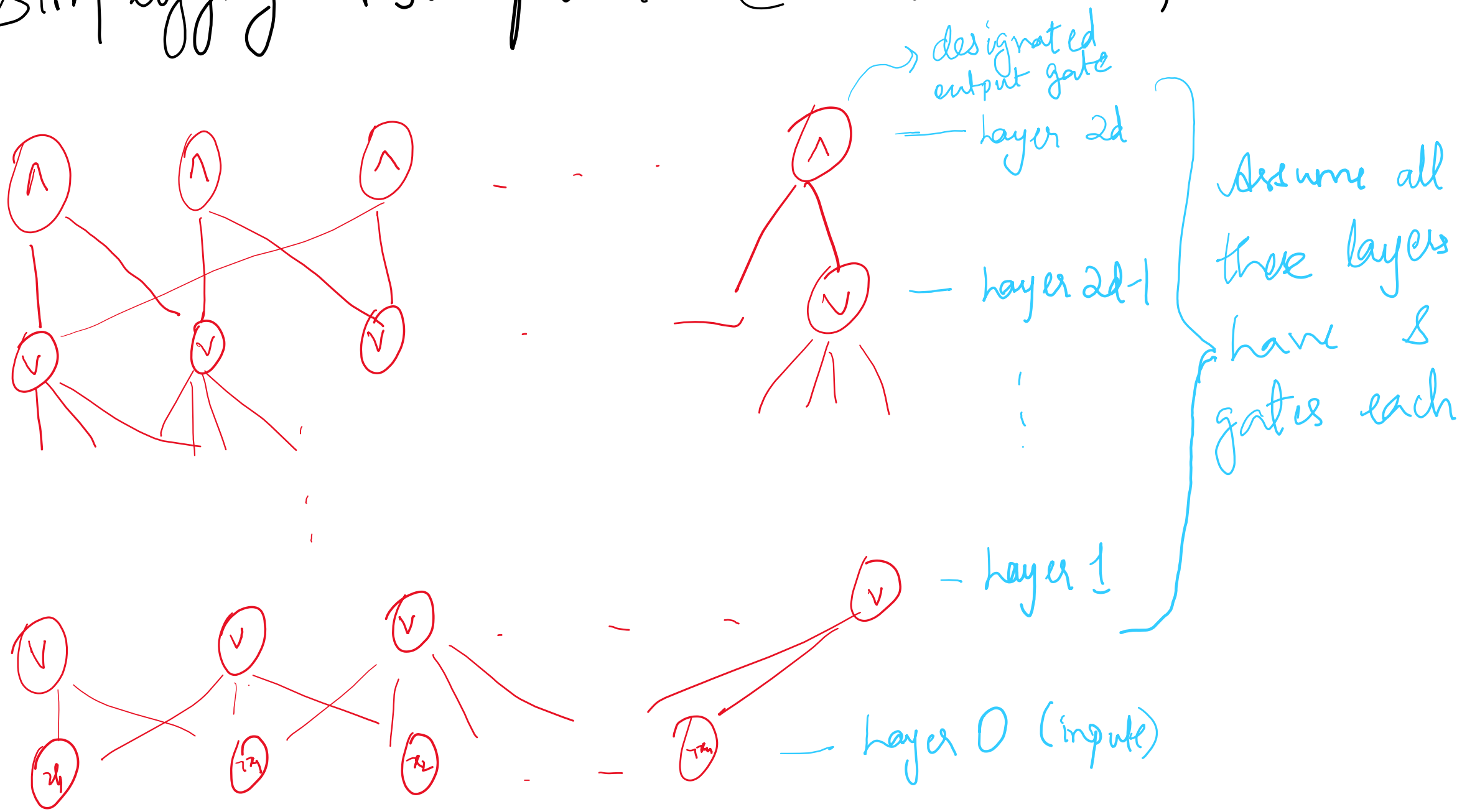
with only fan-in 2 AND gates.



Simplifying Assumption: C looks as follows:



Simplifying Assumption: C looks as follows:



Main Idea: "Inductive Counting"

Main Idea: "Inductive Counting"

> Keep track of $P_k :=$ # of gates that evaluate to 1 on layer $2k$ (AND layer).

Main Idea: "Inductive Counting"

> Keep track of $P_k :=$ # of gates that evaluate to 1 on layer $2k$ (AND layer).

> $P_0 = n$

Main Idea: "Inductive Counting"

> Keep track of $P_k :=$ # of gates that evaluate to 1 on layer $2k$ (AND layer).

> $P_0 = n$

> Want to construct a predicate

$$\text{count}(c, k) = \begin{cases} 1, & \text{if } P_k = c \\ 0, & \text{if } P_k \neq c \end{cases}$$

Main Idea: "Inductive Counting"

> Keep track of $P_k :=$ # of gates that evaluate to 1 on layer $2k$ (AND layer).

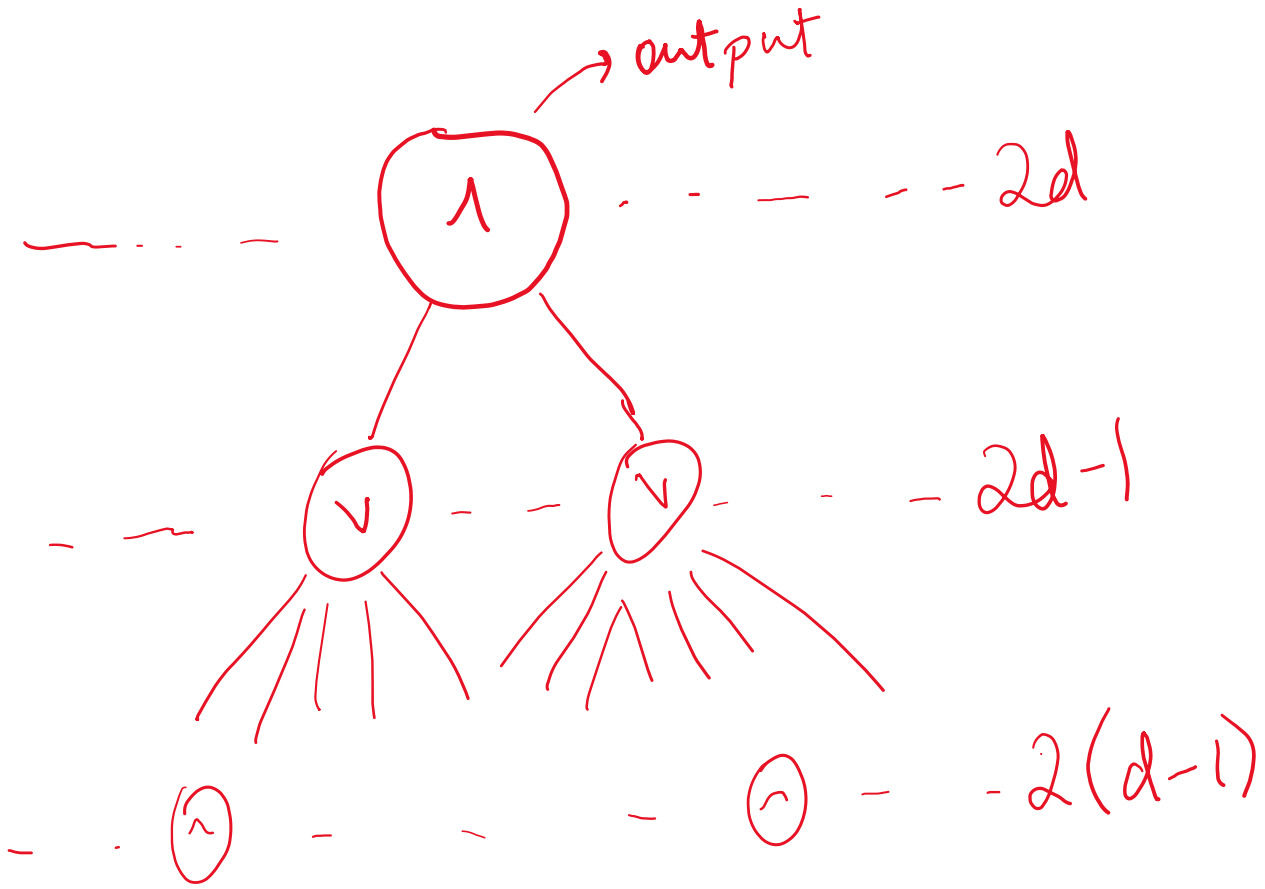
> $P_0 = n$

> Want to construct a predicate

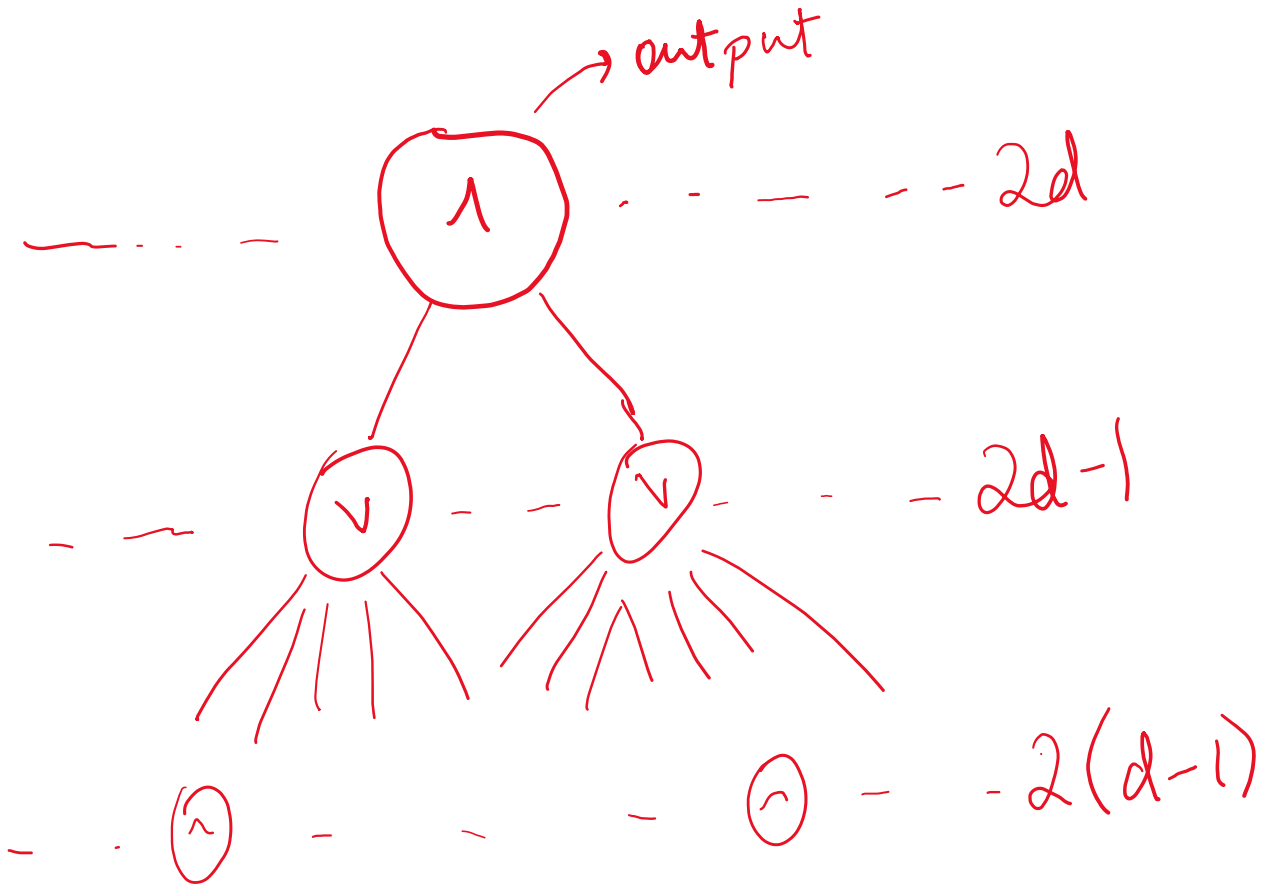
$$\text{count}(c, k) = \begin{cases} 1, & \text{if } P_k = c \\ 0, & \text{if } P_k \neq c \end{cases}$$

Here, $1 \leq c \leq s$, $1 \leq k \leq d$.

What else is needed?

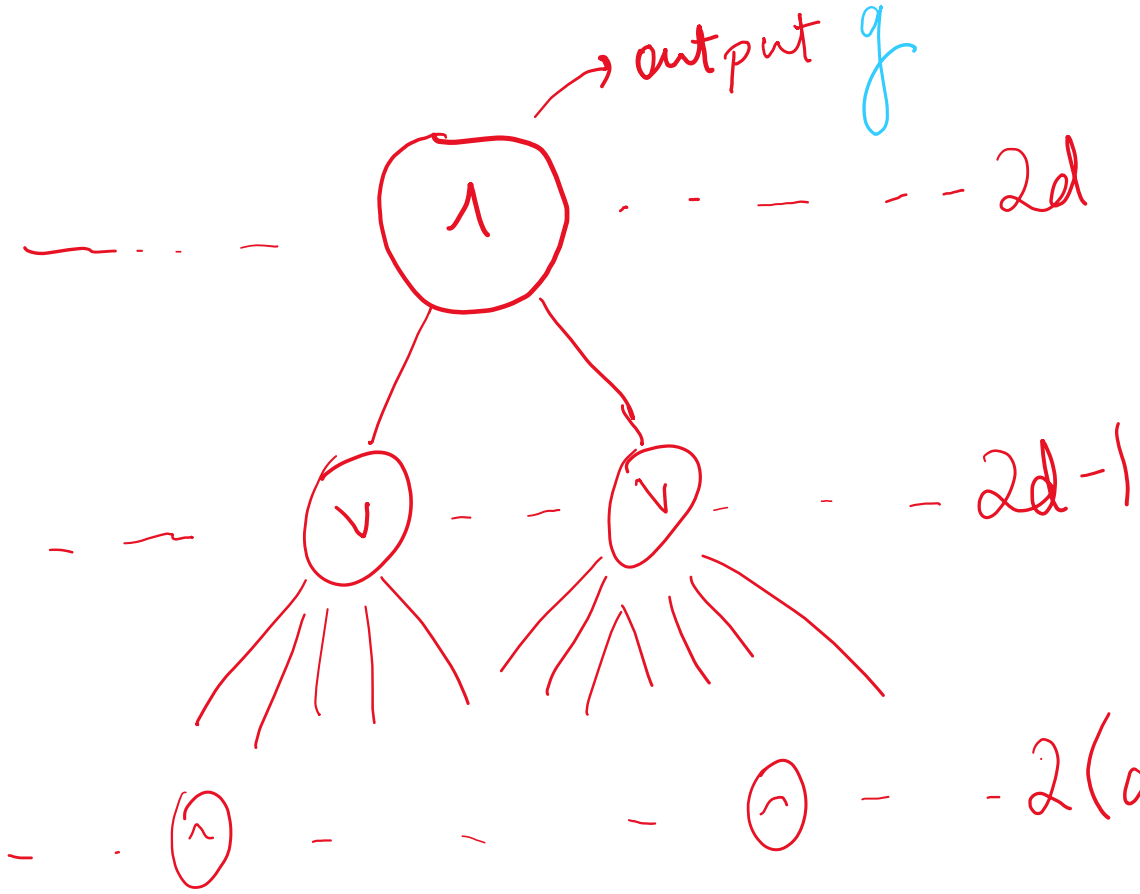


What else is needed?



Know P_{d-1} = the only c for which $\text{count}(c, d-1) = 1$

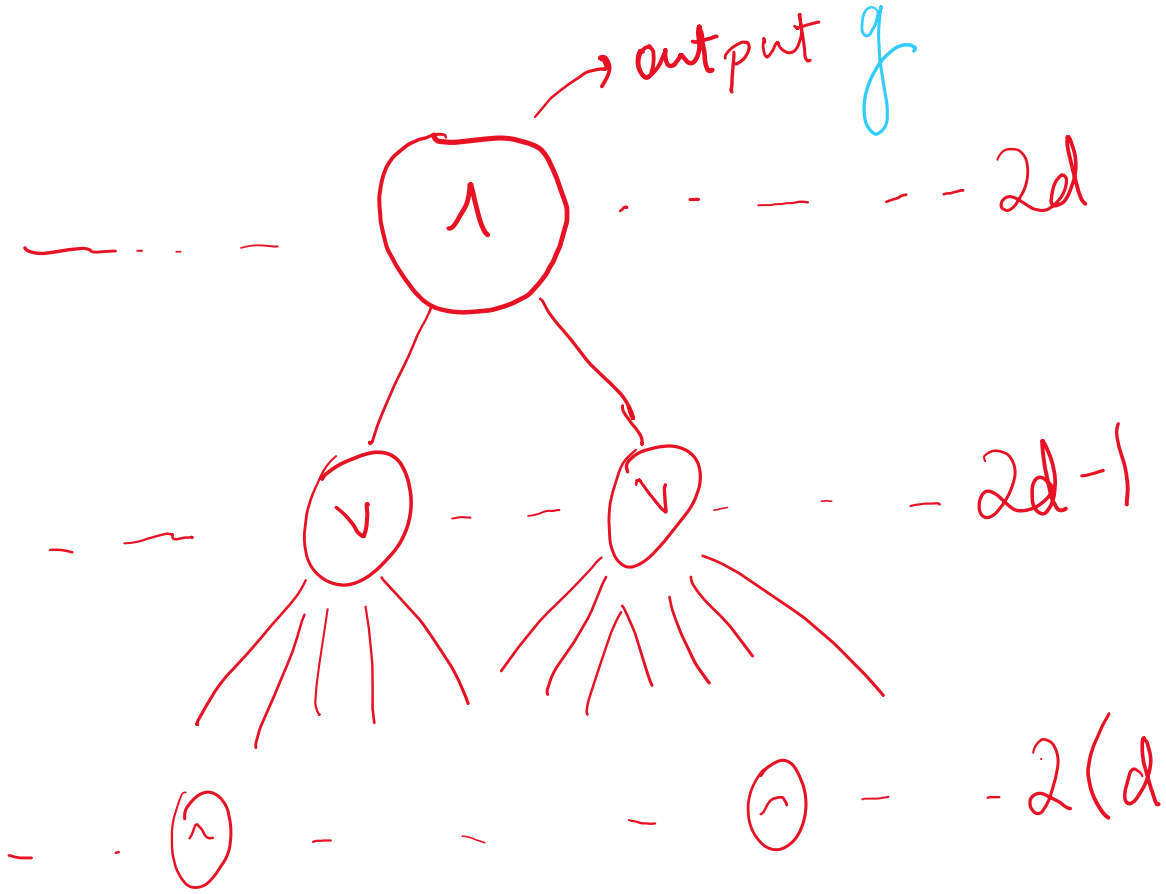
What else is needed?



Simply the complement of g contingent on the fact that $P_{k-1} = C$

Know P_{d-1}

What else is needed?



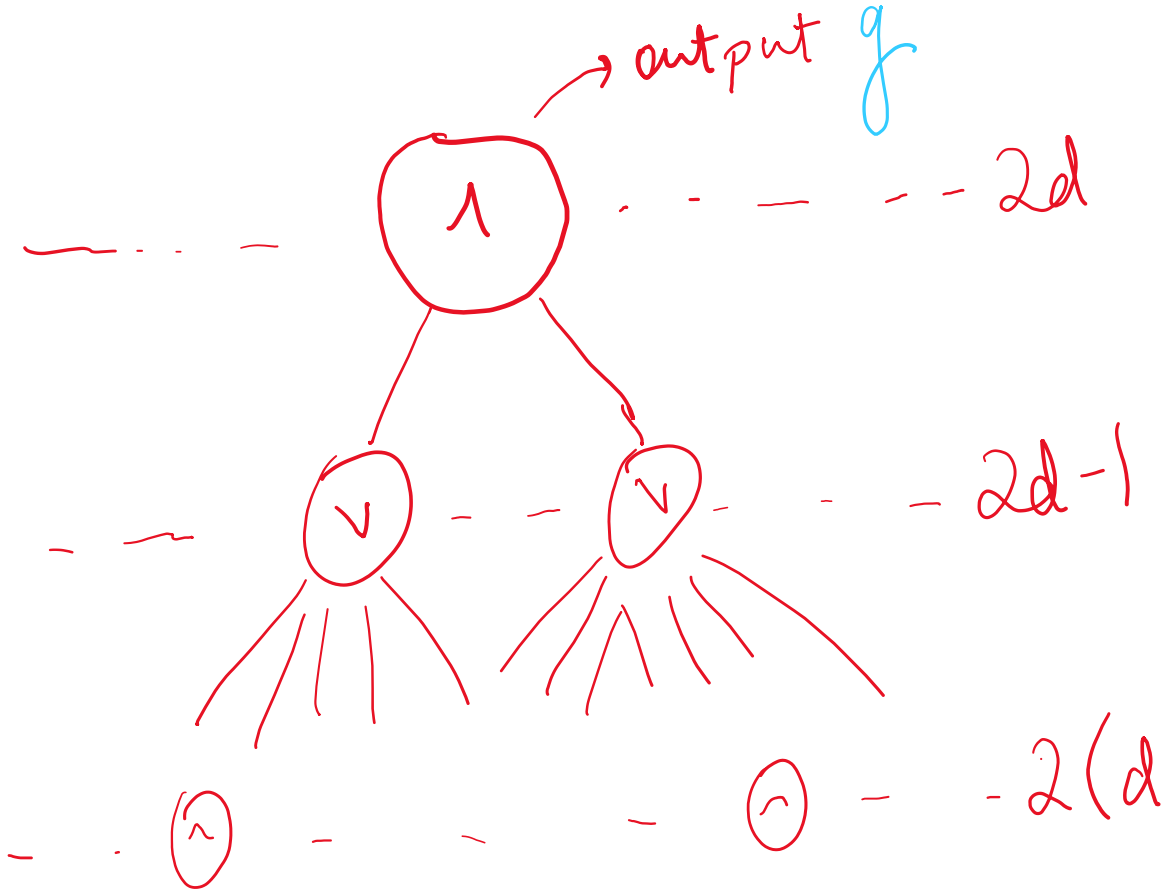
Simply the complement of g contingent on the fact that $P_{k-1} = c$

Know P_{d-1}

Idea is: along with count(c, k), also keep track of.

$CC(f|c) = \neg f$ given that $P_{k-1} = c$

What else is needed?



Simply the complement of g contingent on the fact that $P_{k-1} = c$

Know P_{d-1}

Idea is: along with count(c, k), also keep track of $CC(f|c) = \neg f$ given that $P_{k-1} = c$ (for all gates f of C and all $1 \leq c \leq s$)

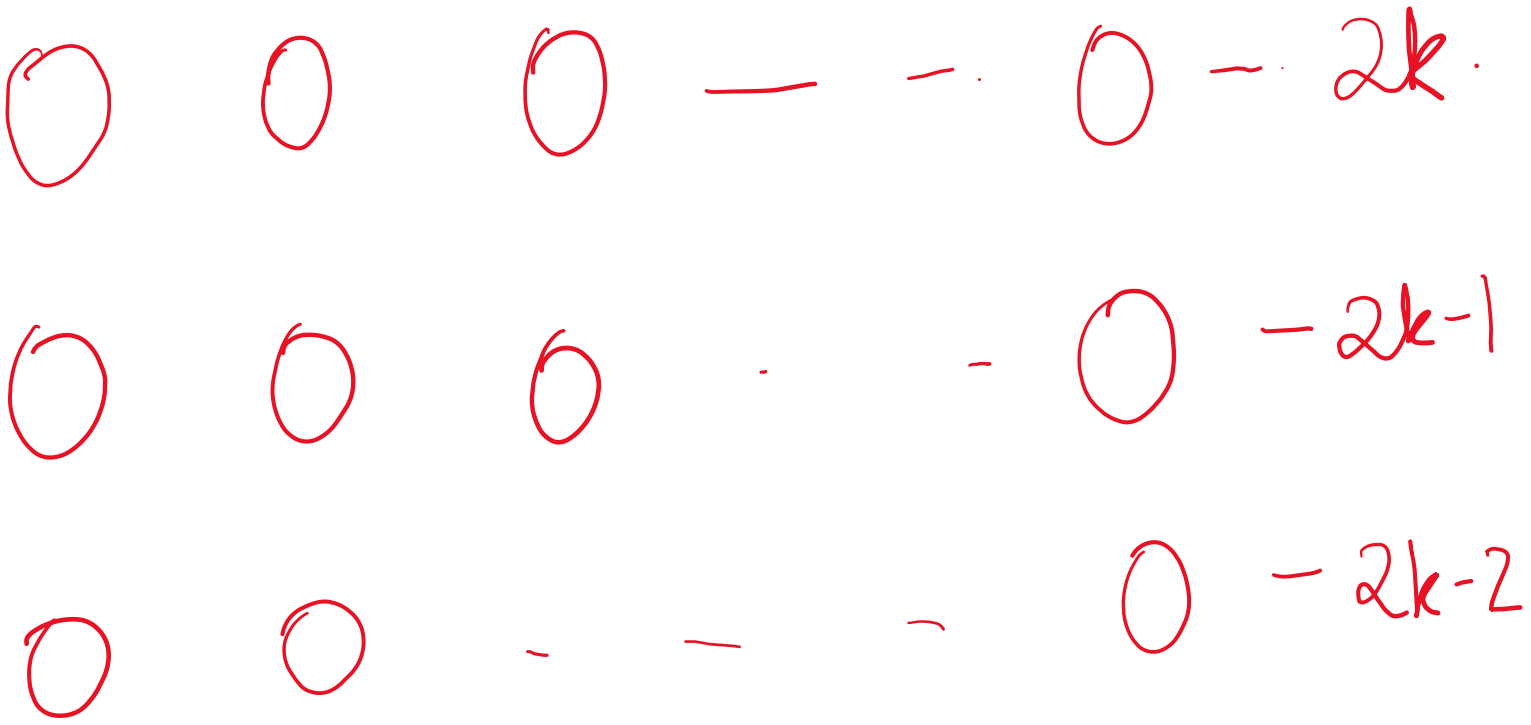
Okay, but how do you compute count(c, k), CC(flc) inductively?

Okay, but how do you compute count(c, k), CC(f|c) inductively?

→ $k=0$ ✓ ($P_0 = n$) i.e. $\text{count}(c, 0) = 1$ iff $c = n$.

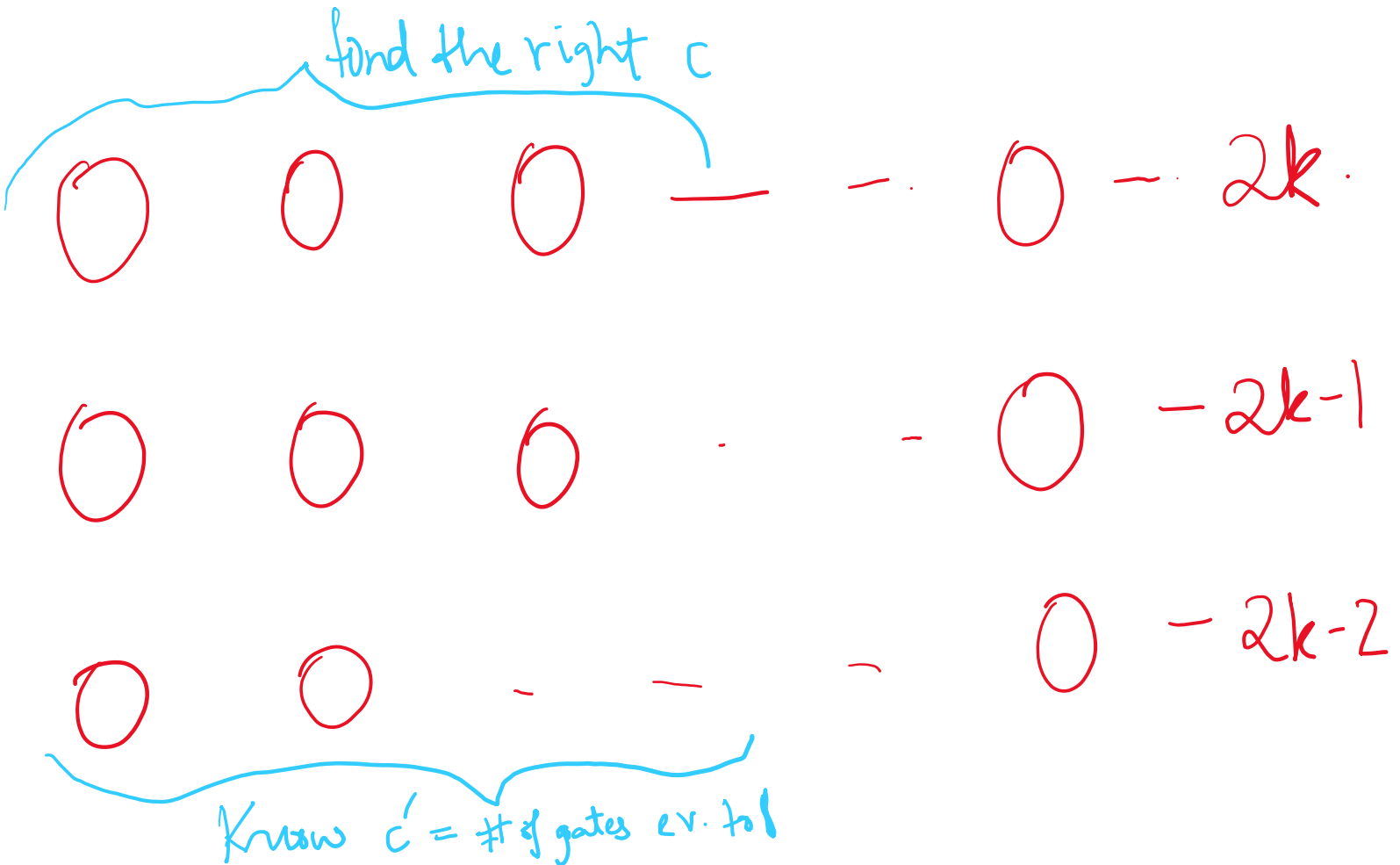
Okay, but how do you compute count(c, k), CC(f|c) inductively?

→ k=0 ✓ ($P_0 = n$) i.e. count(c, 0) = 1 iff c = n.



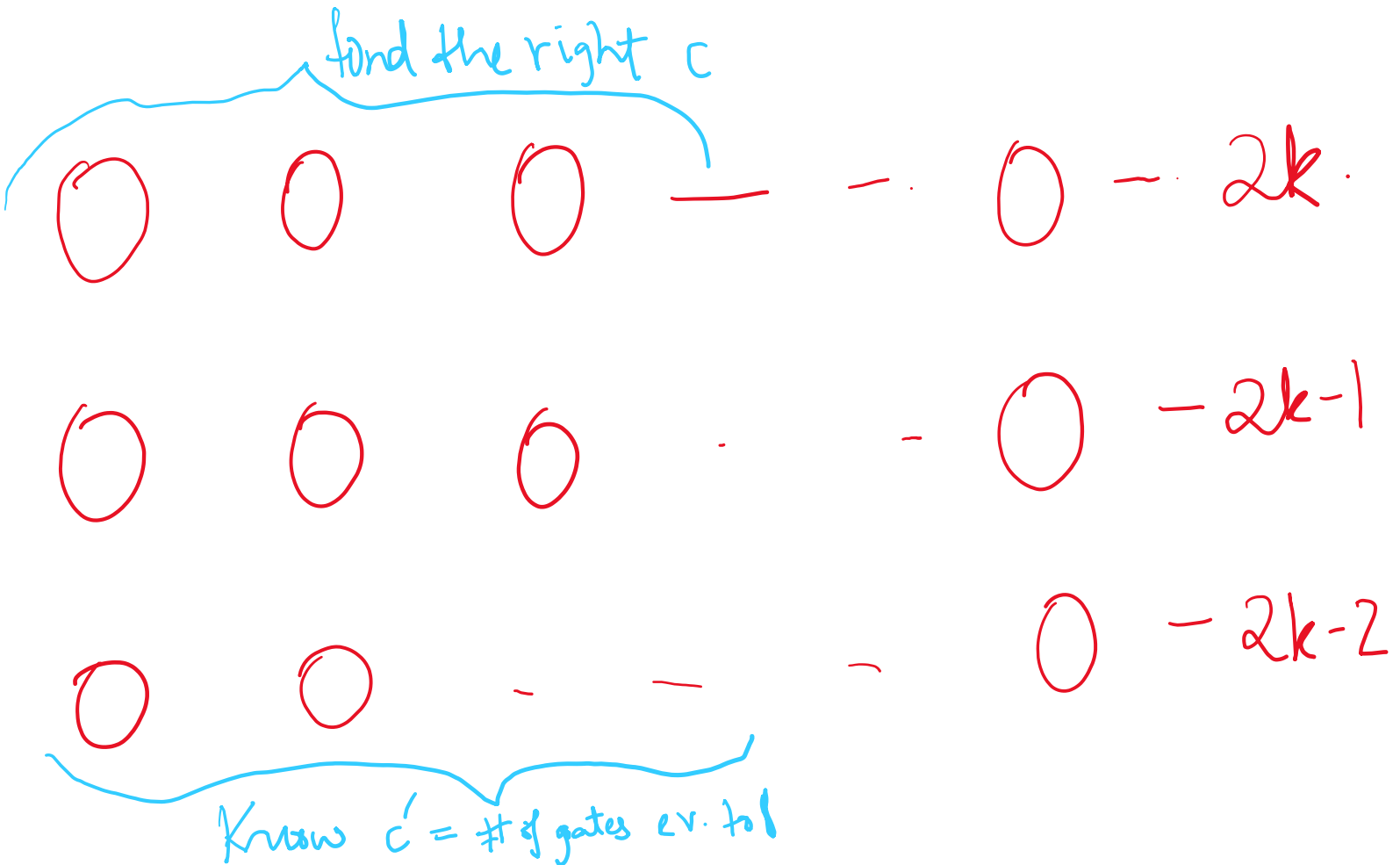
Okay, but how do you compute count(c, k), CC(f|c) inductively?

- $k=0$ ✓ $(P_0 = n)$ i.e. count(c, 0) = 1 iff $c = n$.



Okay, but how do you compute count(c, k), CC(f|c) inductively?

- $k=0$ ✓ $(P_0 = n)$ i.e. count(c, 0) = 1 iff $c = n$.



c is the unique $\in \{1, \ldots, s\}$ for which $\geq c$ gates are 1 on layer 2k AND $\geq s-c$ complement gates are 1 on layer 2k.

$$\text{count}(c, k) = \bigvee_{c'=0}^{8} \text{AND}(\text{count}(c', k-1),$$

$$\text{count}(c, k) = \bigvee_{c'=0}^{8} \text{AND}(\text{count}(c', k-1),$$

$$Th_c(\text{all gates on layer } 2k);$$

$$\text{Count}(c,k) = \bigvee_{c'=0}^{8} \text{AND}(\text{count}(c', k-1),$$

$$Th_c(\text{all gates on layer } 2k),$$

$$Th_{s-c}(\{C_c(f|c') \text{ for all gates } f \text{ on layer } 2k\})$$

$$\text{Count}(c,k) = \bigvee_{c'=0}^{s} \text{AND}(\text{count}(c', k-1),$$

$$Th_c(\text{all gates on layer } 2k),$$

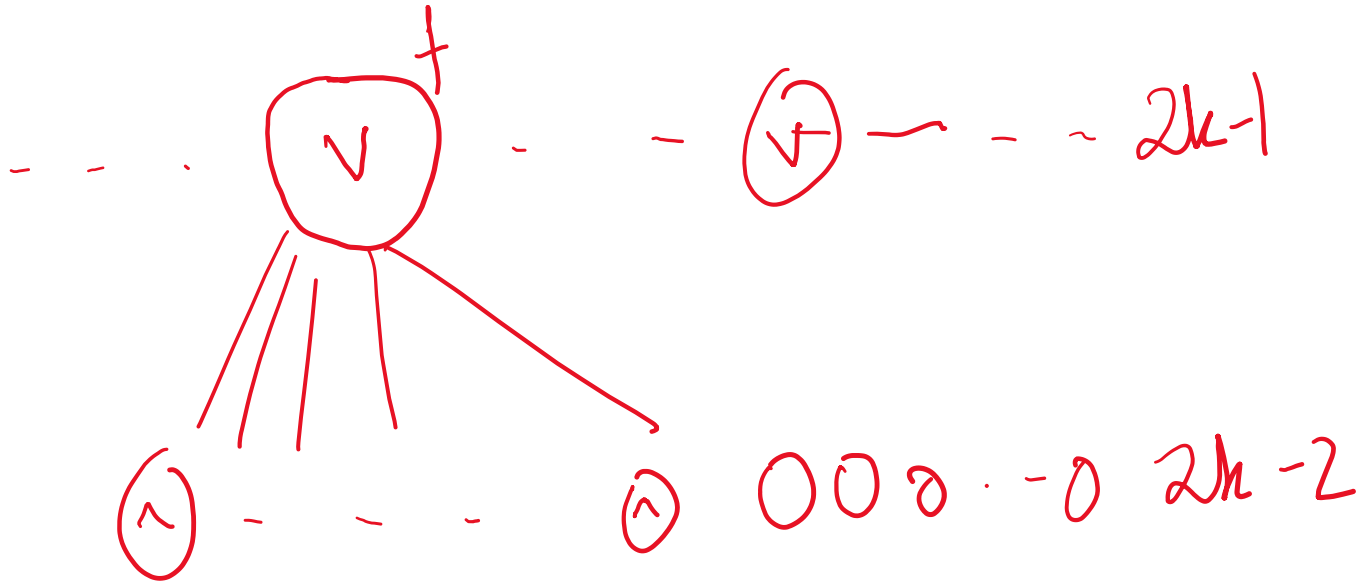
$$Th_{s-c}(\{C_c(f|c') \text{ for all gates } f \text{ on layer } 2k\})$$

Fact: Threshold gates $\in NC^1 \subseteq SAC^1$.

But how do we compute $CC(f|c)$?

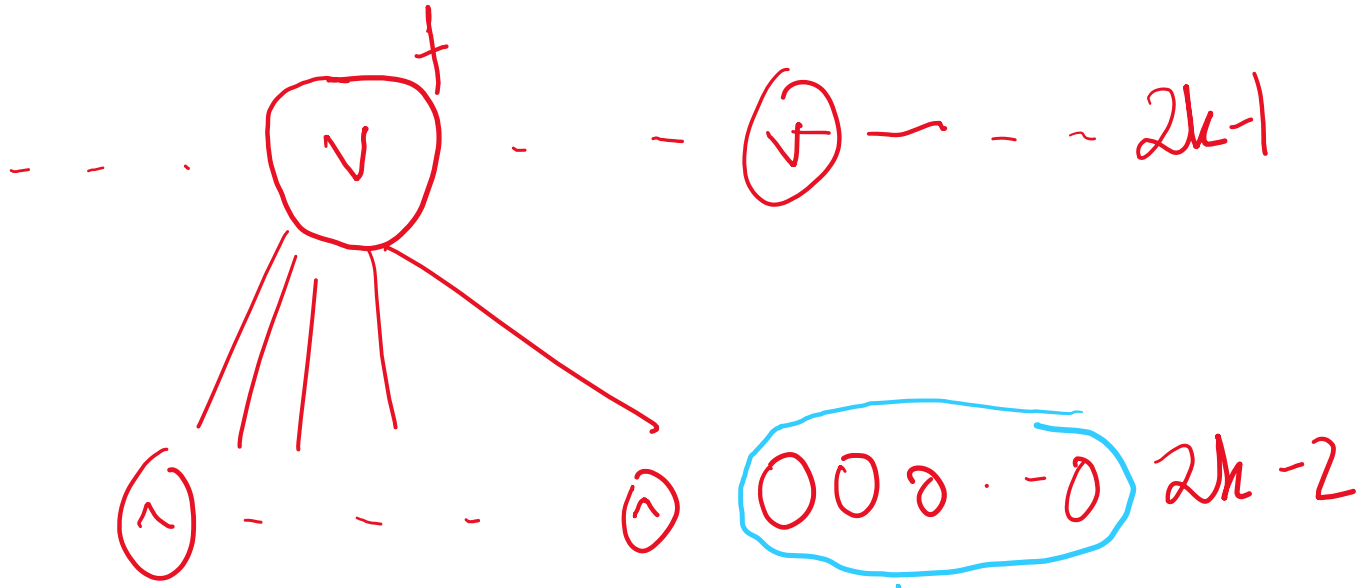
But how do we compute $CC(f|c)$?

f is OR.



But how do we compute $CC(f|c)$?

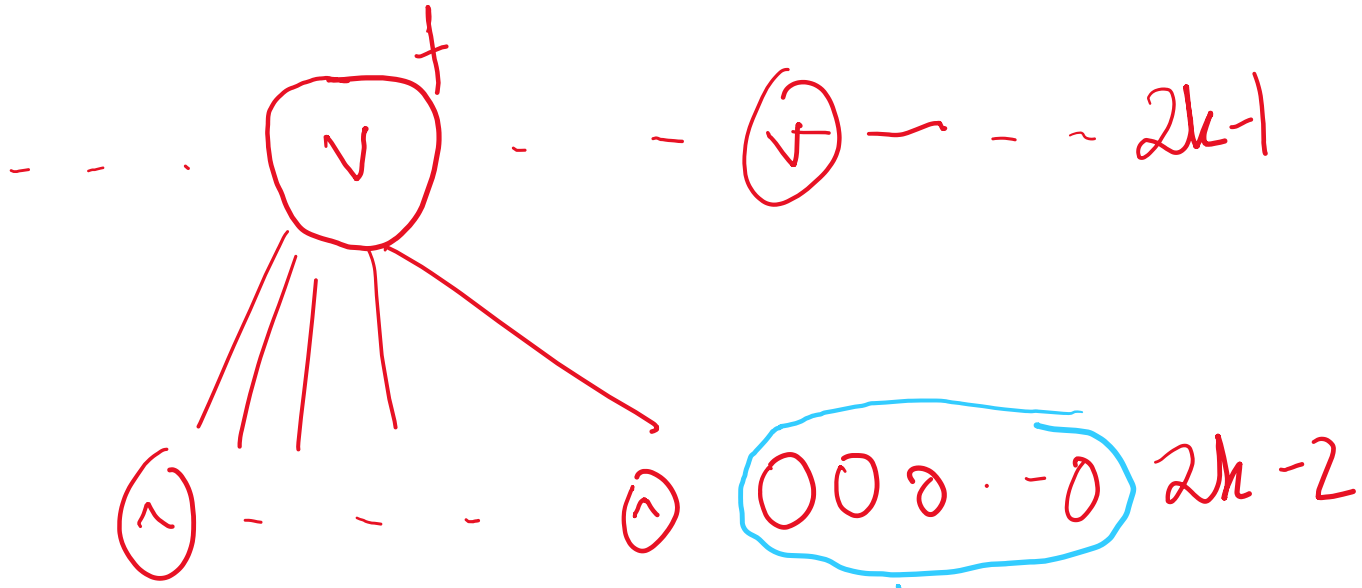
f is OR.



Then $CC(f|c) = Th_c$(these gates)

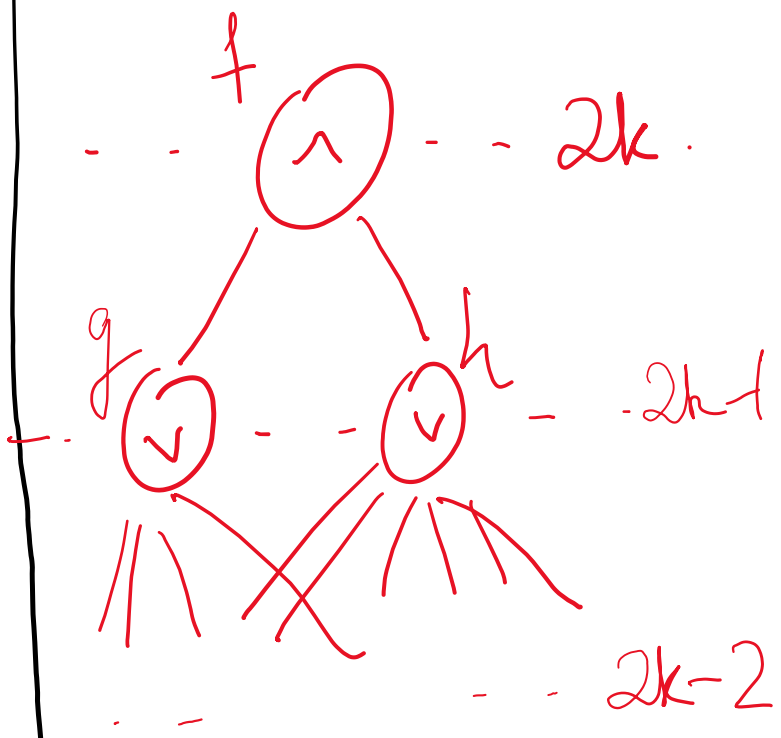
But how do we compute $CC(f|c)$?

f is OR.



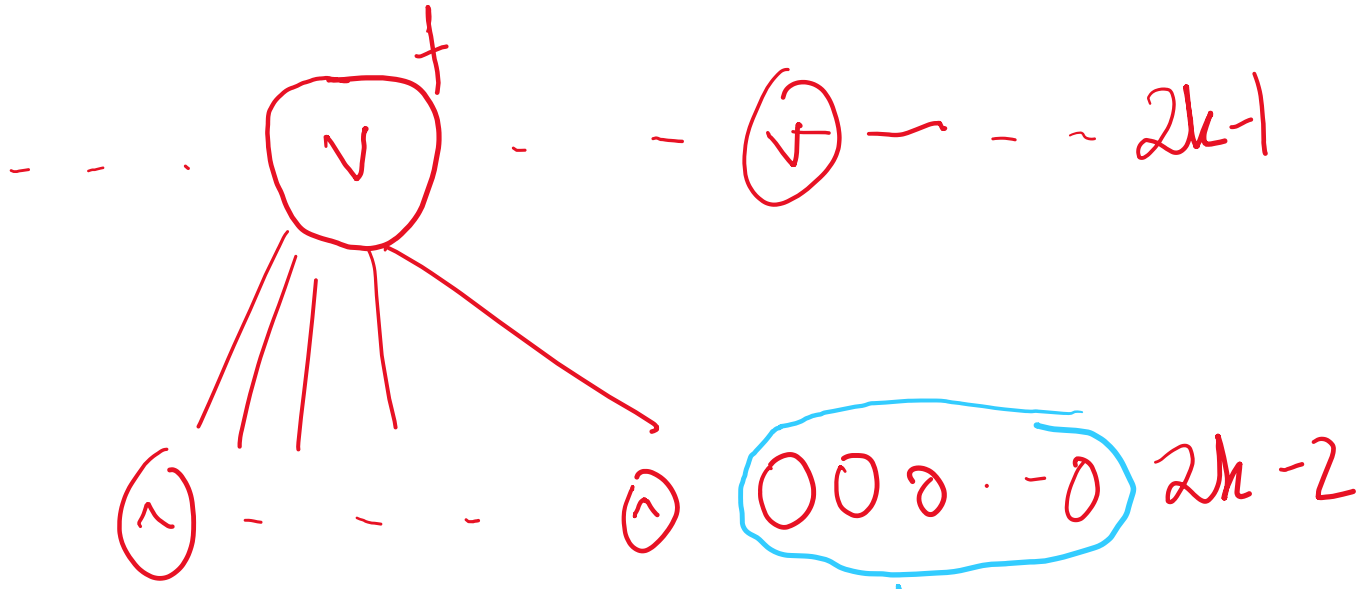
Then $CC(f|c) = Th_c$(these gates)

f is AND



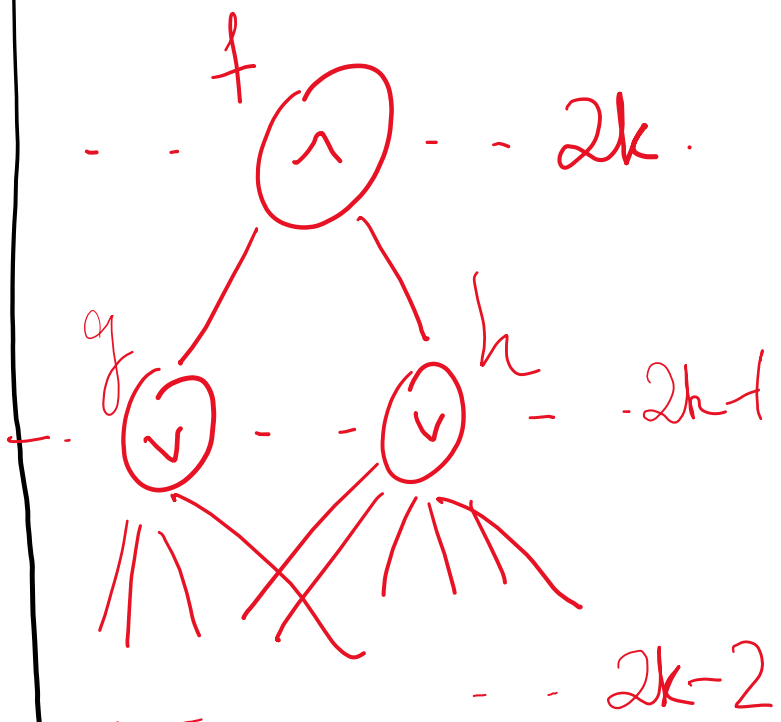
But how do we compute $CC(f|c)$?

f is OR.



Then $CC(f|c) = Th_c$(these gates)

f is AND



Then $CC(f|c) = CC(g|c) \vee CC(h|c)$

Thank You!