Limitations of Lower-Bound Methods
*
for the Wire Complexity of Boolean Operators

Andrew Drucker

MIT

What's this about?

- An introduction to one area of circuit lower bounds work;

- A (partial) explanation of why progress is slow.

What's this about?

- But first: a look at the important theme of

  "joint computation"

  in complexity theory…

- Key question: when can we cleverly combine two or more computations to gain efficiency?

- Our focus: multiple computations on a shared input.

Joint computation

- First example: Sorting!

$\mathrm{SORT}(a_1, \ldots.\ a_n)$:=

$$\mathrm{Rk}_1(a_1, \ldots.\ a_n)\ ,\ \mathrm{Rk}_2(a_1, \ldots.\ a_n)\ ,\ \ldots\ ,\ \mathrm{Rk}_n(a_1, \ldots.\ a_n)$$

- n inputs, n outputs.

Joint computation

- First example: Sorting!

$\mathrm{SORT}(a_1, \ldots.\ a_n)$:=

$\mathrm{Rk}_1(a_1, \ldots.\ a_n)$, $\mathrm{Rk}_2(a_1, \ldots.\ a_n)$, … , $\mathrm{Rk}_n(a_1, \ldots.\ a_n)$

- For each $i \in [n]$, can determine $\mathrm{Rk}_i(a_1, \ldots.\ a_n)$ using $\Theta(n)$ comparisons… **[Blum et al., '73]**

- But, can compute all values with $O(n \log n)$ comparisons!

Joint computation

- Second example: Linear transformations

$L(x_1, \dots x_n) :=$

$L_1(x_1, \dots x_n), L_2(x_1, \dots x_n), \dots, L_n(x_1, \dots x_n)$

- For each i, L_i needs $\Theta(n)$ arithmetic operations to compute (individually, and in general).

- But for important examples like L = DFT, can compute L with $O(n \log n)$ operations!

Joint computation

- Third example: Matrix multiplication

$\mathrm{Mult}(A, B) := A * B$

- Each output coordinate of an n-by-n MM takes $\Theta(n)$ arithmetic operations.

- **[Strassen, others]**: can compute $A * B$ with $O(n^{3 - \varepsilon})$ operations!

Joint computation

- Third example: Matrix multiplication

Mult(A, B) := A * B

- Each output of an n by n MM takes $O(n)$ arithmetic op

- [S - ε) operations!

In each of these models/problems, efficient joint computation is the central issue!

Lower bounds

- **Main challenge:** prove for some explicit operator

$$F(x) = (f_1(x), f_2(x), \ldots ,f_n(x)),$$

and complexity measure C, that

$$C(F) \gg \mathrm{Max}_i\, C(f_i) .$$

- (Hopefully for important ones like DFT, MM, etc.!)

- “limits to computational synergies.”

What’s known?

- A brief, partial review for some natural models…

Monotone ckts: an early success story

- Before **[Razborov '85]**, no superlinear LBs for any Boolean function in the monotone circuit model.

- But for Boolean operators, interesting results were long known **[Nechiporuk '71, … , Wegener '82]:**
  - ∃ monotone $F: \{0, 1\}^n \rightarrow \{0, 1\}^n$ such that:

    $C_m(f_i) = \Theta(n)$, $\qquad C_m(F) = \Omega(n^2/\log n)$.

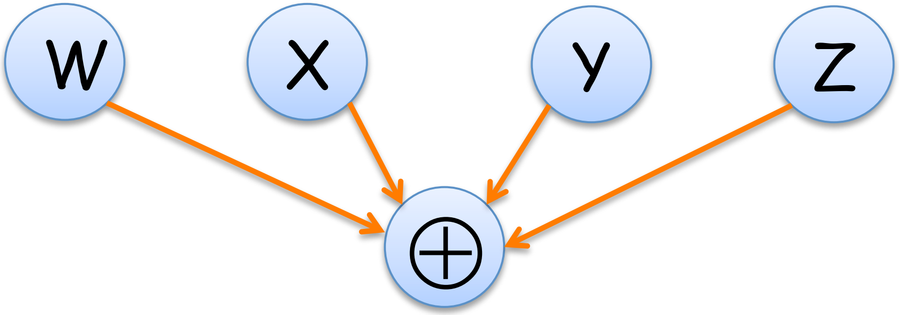
  - For Boolean matrix mult., and some other natural monotone operators, naïve approaches are ≈ <u>optimal</u> for monotone ckts!

Linear operators:
things get (much) trickier

$L(x)$: $\{0, 1\}^n \rightarrow \{0, 1\}^n$

$L \in \{0, 1\}^{n \times n}$ described by a 0/1 ($\mathbf{F}_2$) matrix.

- Natural computational model: $\mathbf{F}_2$-linear circuits.



- Natural cost measure: number of wires.

Linear operators:
things get (much) trickier

$L(x): \{0, 1\}^n \rightarrow \{0, 1\}^n$

- For random L, $L(x)$ takes $\Theta(n^2/\log n)$ wires to compute by a linear circuit. **[Lupanov '56]**

- For explicit examples, no superlinear LBs known!
  … except in constant depth.

- Bounds are quite modest, as we'll see…

Linear operators:
things get (much) trickier

$L(x): \{0, 1\}^n \rightarrow \{0, 1\}^n$

- More discouragingly (perhaps): best lower bounds known don't even exploit the

linear structure of linear circuits!

- Can get by with "generic" techniques…

- Don't even know if "non-linearity" helps!

Generic techniques

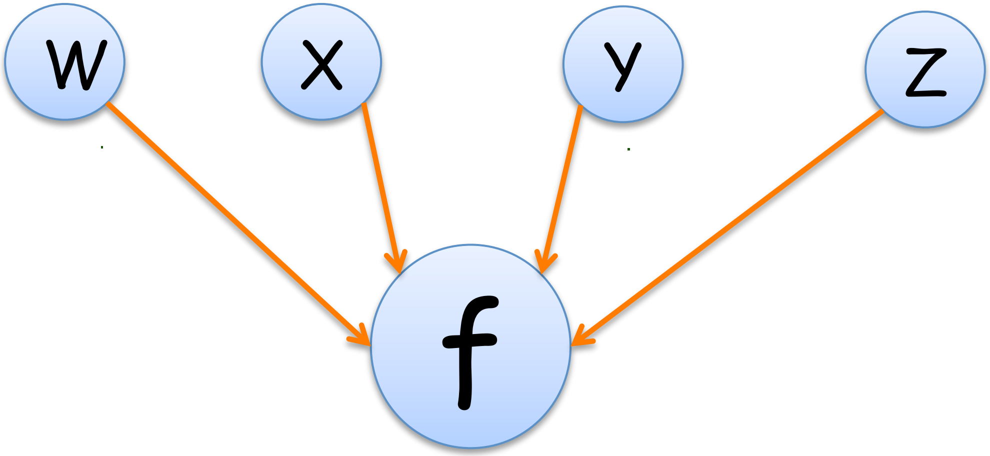
Generic techniques

- What are these “generic” circuit LB techniques?

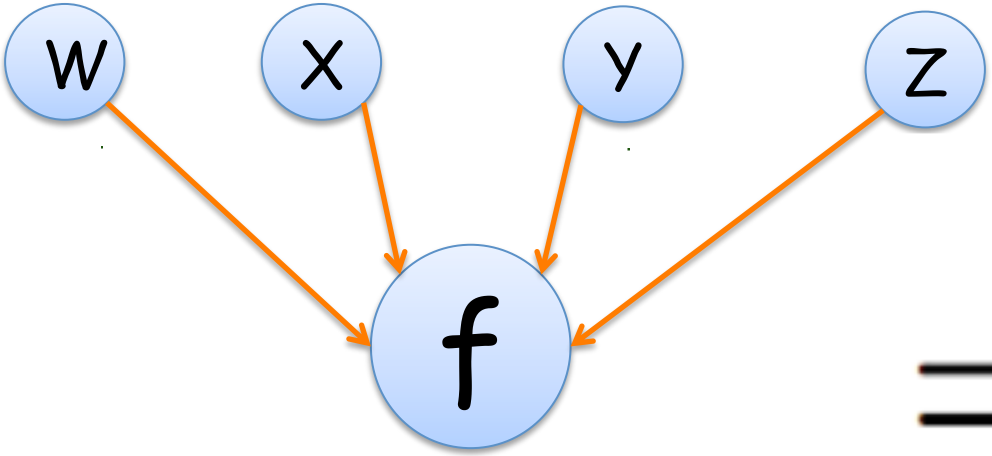
- What are their virtues and limitations?

- Next: a model of “generic circuits” used to help understand these issues. **[‘70s]**

The arbitrary-gates model



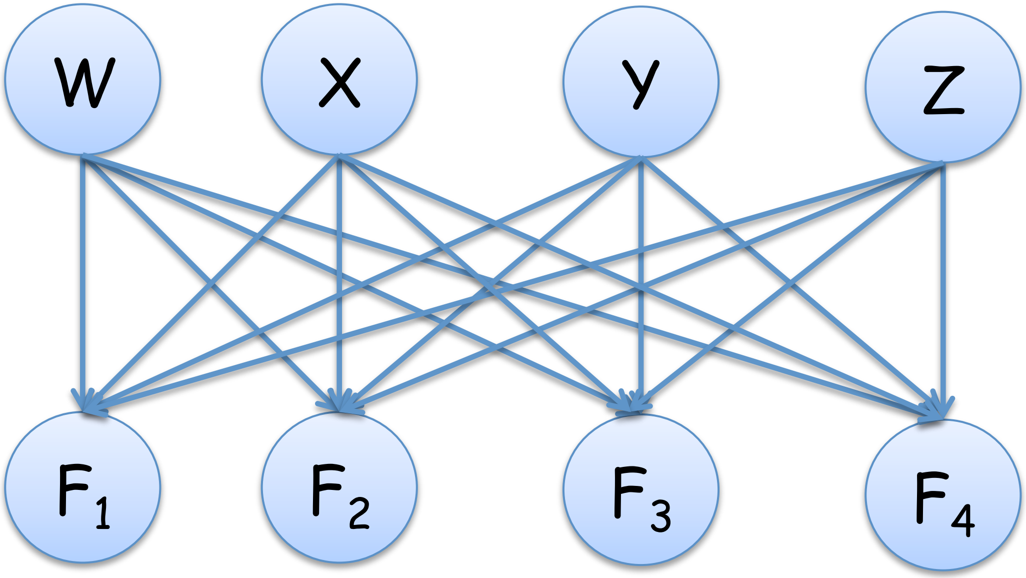
The arbitrary-gates model



=

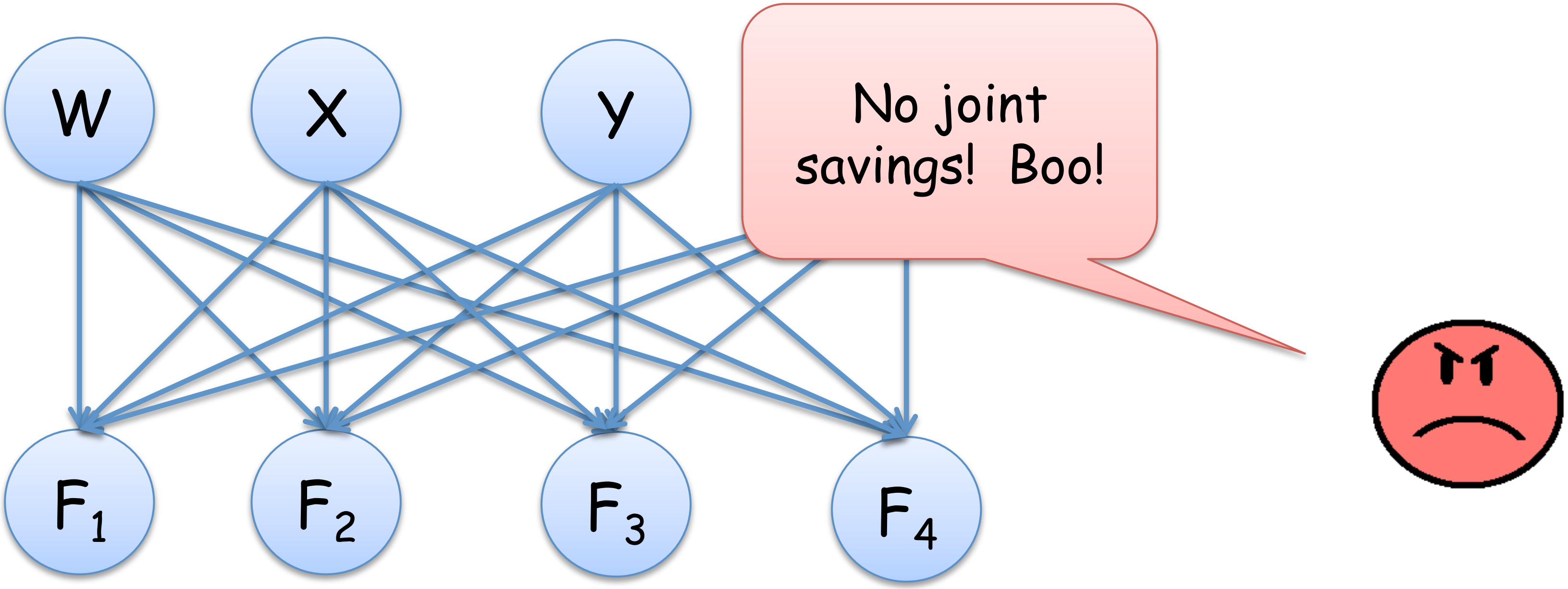
The arbitrary-gates model

- Here, any F: {0, 1}^{n} → {0, 1}^{n} can be trivially computed with n^2 gates!



The arbitrary-gates model

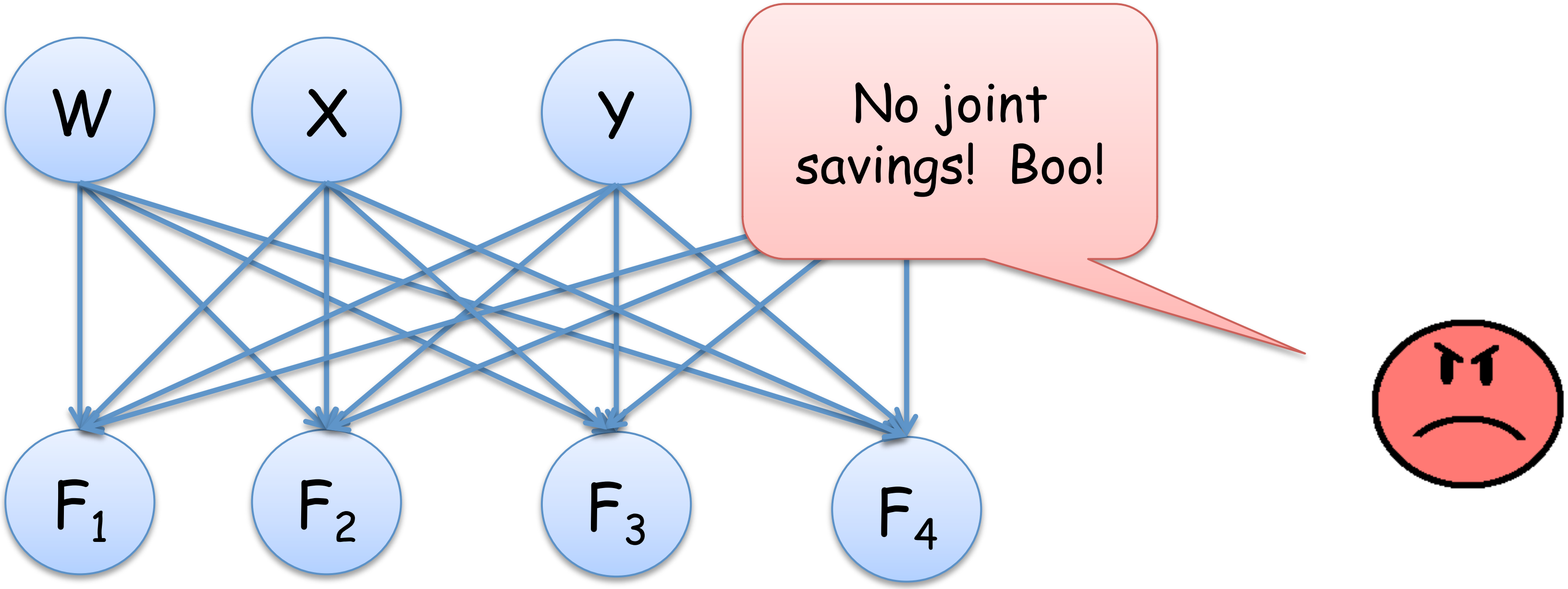
- Here, any $F: \{0, 1\}^n \rightarrow \{0, 1\}^n$ can be trivially computed with n^2 gates!



No joint savings! Boo!

The arbitrary-gates model

- Here, any $F: \{0, 1\}^n \to \{0, 1\}^n$ can be trivially computed with n^2 gates!



No joint savings! Boo!

- The arb-gates model: a “pure” setting to study efficient joint computation.

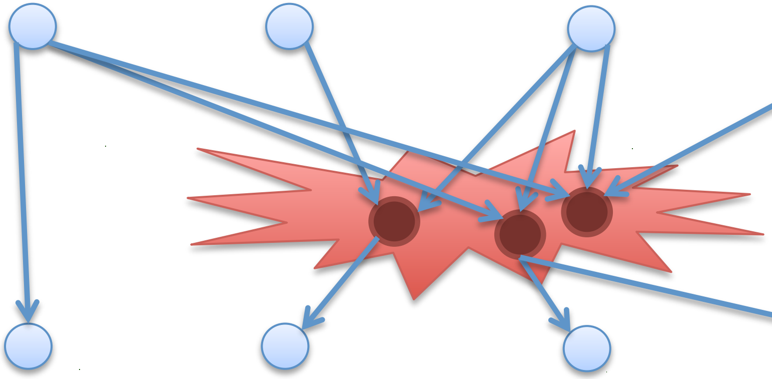
The arbitrary-gates model

- Perhaps surprisingly: we can prove some lower bounds in this model!

Connectivity arguments

- Basic idea behind most LBs in the arb-gates model:

-If the edges in *C* are too few, and the depth too low,
Graph theory → a bottleneck must appear in the circuit.
-Information "can't get through"...



Connectivity arguments

- Lower bounds are then implied for operators F whose circuits require a strong connectivity property.

- Most famous/influential: the superconcentrator property **[Valiant '75]**. Some $F: \{0, 1\}^n \rightarrow \{0, 1\}^n$ require a circuit C whose graph obeys:

For any $S, T \subseteq$ *(inputs x outputs)* with $|S| = |T|$, $\exists$ vertex-disjoint paths in C matching S with T.

Connectivity arguments

- Lower bounds are then implied for operators F whose circuits require a strong connectivity property.

- Most famous/influential: the superconcentrator property **[Valiant '75]**. Some $F: \{0, 1\}^n \rightarrow \{0, 1\}^n$ require a circuit C whose graph obeys:

> For any $S, T \subseteq$ *(inputs x outputs)* with $|S| = |T|$, $\exists$ vertex-disjoint paths in C matching S with T.

- Other, related connectivity properties can be more widely applicable for lower bounds, e.g. when F is linear…

Connectivity arguments

- Lower bounds are then implied for operators F whose circuits require a strong connectivity property.

- Most famous/influential: the superconcentrator property **[Valiant '75]**. Some $F: \{0, 1\}^n \rightarrow \{0, 1\}^n$ require a circuit C whose graph obeys:

For any $S, T \subseteq$ *(inputs x outputs)* with $|S| = |T|$, $\exists$ vertex-disjoint paths in C matching S with T.

- **[Pudlák '94; Raz-Sphilka '03; Gál et al. '12]**

Connectivity arguments

- Lower bounds are then implied for operators F whose circuits require a strong connectivity property.

- Most famous/influential: the superconcentrator property **[Valiant '75]**. Some $F: \{0, 1\}^n \rightarrow \{0, 1\}^n$ require a circuit C whose graph obeys:

For any $S, T \subseteq$ *(inputs x outputs)* with $|S| = |T|$, $\exists$ vertex-disjoint paths in C matching S with T.

- These sometimes match, but don't beat, superconcentrator LBs.

Connectivity arguments

- Virtues of the known "connectivity-based" lower bounds:
  - They apply to all reasonable Boolean circuit models.
  - They're intuitive.
- Drawbacks:
  - Quantitative bounds leave much to be desired.
  - This weakness is inherent, due to known constructions of sparse, low-depth superconcentrators (and related objects).

What do we get?

- Superconcentrator-based lower bounds: **[Dolev et al. '83; Alon, Pudlak '94; Pudlak '94; Radhakrishnan, Ta-Shma '00]**

Depth d	Bound
2	$\Omega(n \log^2 n / \log \log n)$
3	$\Omega(n \log \log n)$
4	$\Omega(n \log^* n)$
5	$\Omega(n \log^* n)$
6	$\Omega(n \log^{**} n)$
7	$\Omega(n \log^{**} n)$
.	
.	
d	$\Omega_d(n\, \lambda_d(n))$

What do we get?

- Superconcentrator-based lower bounds: **[Dolev et al. '83; Alon, Pudlak '94; Pudlak '94; Radhakrishnan, Ta-Shma '00]**

Depth d	Bound
2	$\Omega(n \log^2 n / \log \log n)$
3	$\Omega(n \log \log n)$
4	$\Omega(n \log^* n)$
5	$\Omega(n \log^* n)$
6	$\Omega(n \log^{**} n)$
7	$\Omega(n \log^{**} n)$
.	
.	
d	$\Omega_d(n \lambda_d (n))$

(Warning: competing notations…)

What do we get?

- Superconcentrator-based lower bounds: **[Dolev et al. '83; Alon, Pudlak '94; Pudlak '94; Radhakrishnan, Ta-Shma '00]**

Depth d	Bound
2	$\Omega(n \log^2 n / \log \log n)$
3	$\Omega(n \log \log n)$
4	$\Omega(n \log^* n)$
5	$\Omega(n \log^* n)$
6	$\Omega(n \log^{**} n)$
7	$\Omega(n \log^{**} n)$
.	

All shown asymptotically tight in these papers!

What do we get?

- Superconcentrator-based lower bounds: **[Dolev et al. '83; Alon, Pudlak '94; Pudlak '94; Radhakrishnan, Ta-Shma '00]**

Depth d	Bound
2	$\Omega(n \log^2 n / \log \log n)$
3	$\Omega(n \log \log n)$
4	$\Omega(n \log^* n)$
5	$\Omega(n \log^* n)$
6	$\Omega(n \log^{**} n)$
7	$\Omega(n \log^{**} n)$
.	
.	
d	$\Omega_d(n \lambda_d (n))$

(Best bounds for explicit linear operators a bit weaker)

LBs of this form proved for explicit linear <u>and</u> non-linear operators

A new dawn?

- 2008: Cherukhin gives a new lower-bound technique for arbitrary-gates circuits:
  - First asymptotic improvements over the superconcentrator-based bounds!
  - An information-theoretic, rather than connectivity-based, lower-bound criterion.

    (Proof still uses connectivity ideas, though.)
  - Invented for Cyclic Convolution operator; described as a general lower-bound technique by **[Jukna '12]**.

Cherukhin's idea

- Given $F = (f_j)$: $\{0, 1\}^n \rightarrow \{0, 1\}^n$, suppose $i \in I \subseteq [n]$.
- Let $f_{j\ [I,\ i]}$ be the restriction of f_j that sets $x_i = 1$ and zeros out $(I \setminus i)$.
- For $J \subseteq [n]$, define the operator

$$F_{I,\ J} := (f_{j\ [I,\ i]} \ \mid i \in I \ , j \in J \ ).$$

Cherukhin's idea

- Define an operator's entropy as

$$\mathrm{Ent}(F) := \log_2 (|\mathrm{range}(F)|).$$

- Cherukhin: $\mathrm{Ent}(F_{I, J})$ is a useful measure of "information flow" in F between I, J.

- "Strong Multiscale Entropy" (SME) property **[Cherukhin, Jukna]** says:
  - Roughly speaking: $\mathrm{Ent}(F_{I, J})$ is large for many pairs I, J, for many choices of a "scale" $p = |I| \approx n/|J|$.

What do we get?

Depth d	Superconc. Bound	SME Bound
2	$\Omega(n \log^2 n / \log\log n)$	$\Omega(n^{1.5})$
3	$\Omega(n \log\log n)$	$\Omega(n \log n)$
4	$\Omega(n \log^* n)$	$\Omega(n \log\log n)$
5	$\Omega(n \log^* n)$	$\Omega(n \log^* n)$
6	$\Omega(n \log^{**} n)$	$\Omega(n \log^* n)$
7	$\Omega(n \log^{**} n)$	$\Omega(n \log^{**} n)$
.		
.		
d	$\Omega_d(n \lambda_d (n))$	$\Omega_d(n \lambda_{d-1} (n))$

What do we get?

Depth d	Superconc. Bound	SME Bound
2	$\Omega(n \log^2 n / \log \log n)$	$\Omega(n^{1.5})$
3	$\Omega(n \log \log n)$	$\Omega(n \log n)$
4	$\Omega(n \log^* n)$	$\Omega(n \log \log n)$
5	$\Omega(n \log^* n)$	$\Omega(n \log^* n)$
6	$\Omega(n \log^{**} n)$	$\Omega(n \log^* n)$
7	$\Omega(n \log^{**} n)$	$\Omega(n \log^{**} n)$
.		
.		
d	$\Omega_d(n\ \lambda_d\ (n))$	$\Omega_d(n\ \lambda_{d-1}\ (n))$

What do we get?

Depth d	Superconc. Bound	SME Bound
2	$\Omega(n \log^2 n / \log \log n)$	$\Omega(n^{1.5})$
3	$\Omega(n \log \log n)$	$\Omega(n \log n)$
4	$\Omega(n \log^* n)$	$\Omega(n \log \log n)$
5	$\Omega(n \log^* n)$	$\Omega(n \log^* n)$
6	$\Omega(n \log^{**} n)$	$\Omega(n \log^* n)$
7	$\Omega(n \log^{**} n)$	$\Omega(n \log^{**} n)$
.		
.		
d		

(Note: SME property only holds for non-linear operators.)

What do we get?

Depth d	Superconc. Bound	SME Bound
2	$\Omega(n \log^2 n / \log \log n)$	$\Omega(n^{1.5})$
3	$\Omega(n \log \log n)$	$\Omega(n \log n)$
4	$\Omega(n \log^* n)$	$\Omega(n \log \log n)$
5	$\Omega(n \log^* n)$	$\Omega(n \log^* n)$
6	$\Omega(n \log^{**} n)$	$\Omega(n \log^* n)$
7	$\Omega(n \log^{**} n)$	$\Omega(n \log^{**} n)$
.		
.		
d		

Can we get a more substantial improvement in these bounds?

SME – room for improvement?

- Unlike superconcentrator method, limits of the SME criterion were unclear….
- In particular: could the SME criterion, unchanged, imply much better LBs by an improved analysis?
- Our main result: NO.

Our result

- **Theorem:** There's an explicit operator with the SME property, yet computable in depth d with

$O(n\, \lambda_{d-1}(n))$ wires

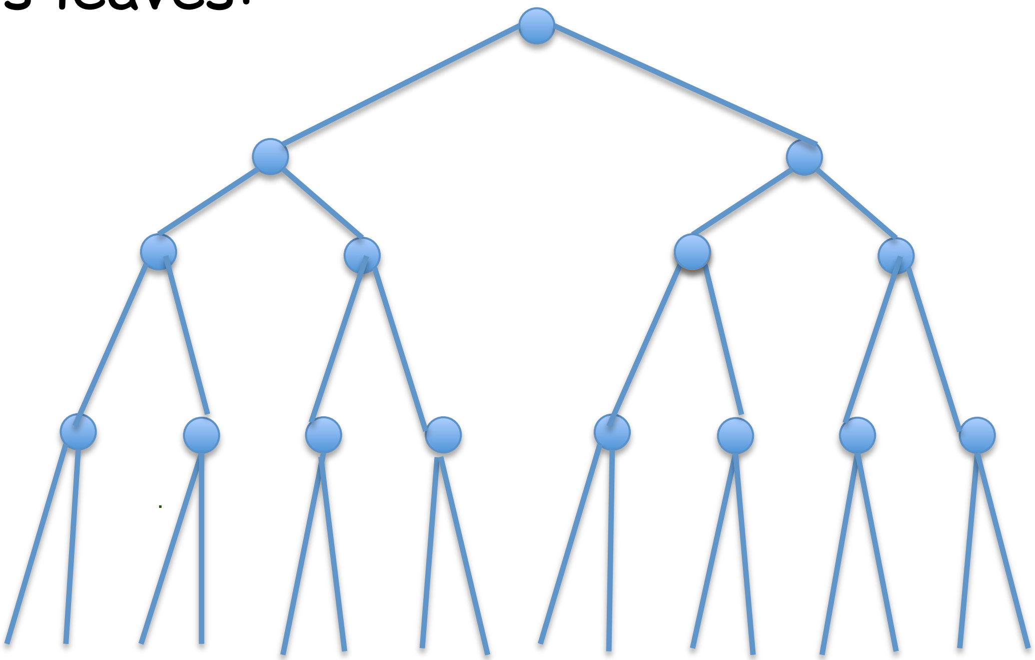
(in the arb-gates model)

(for $d = 2,3$ and for even $d \geq 6$).

Our operator:

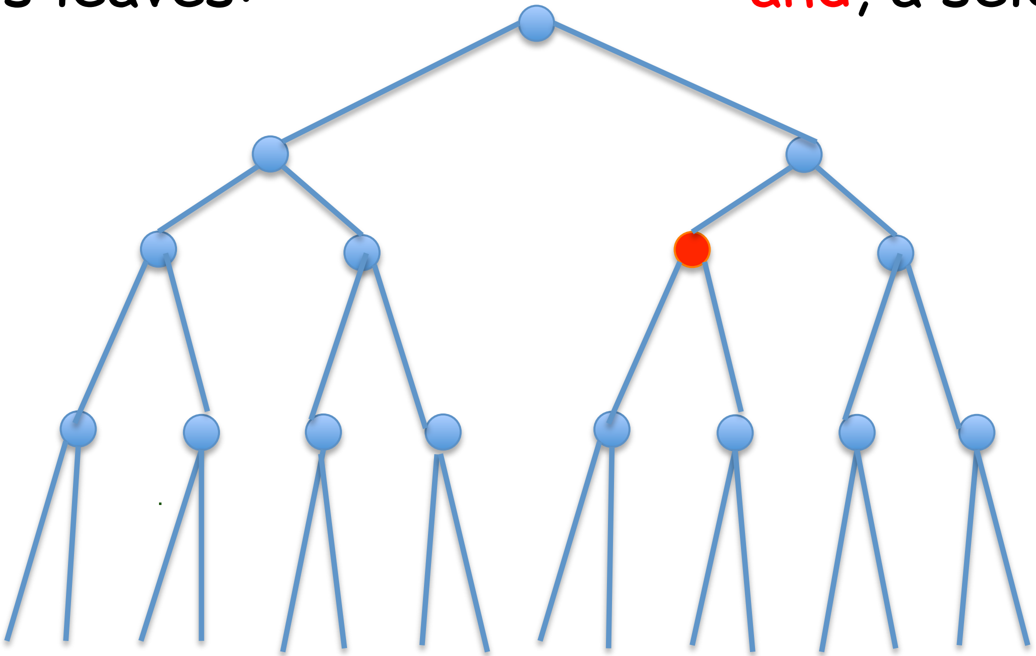
the “Subtree-Copy“ problem

- Input: a string x, regarded as labeling of a full binary tree's leaves:



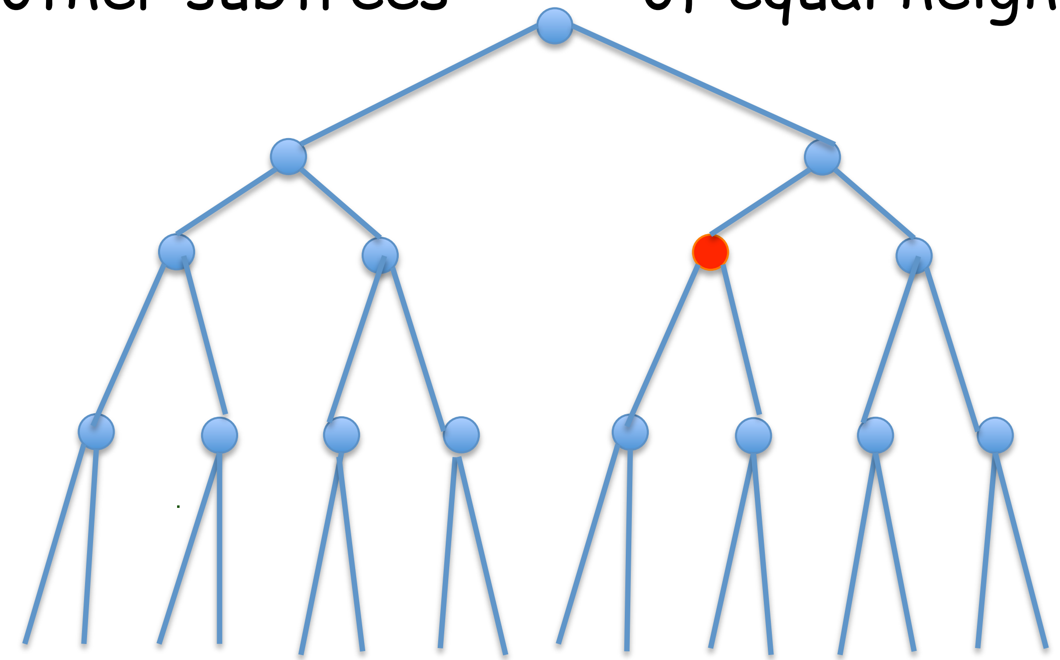
$$x = \underbrace{0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1}_{n = 2^k}$$

- Input: a string x, regarded as labeling of a full binary tree's leaves: and, a selected node v.



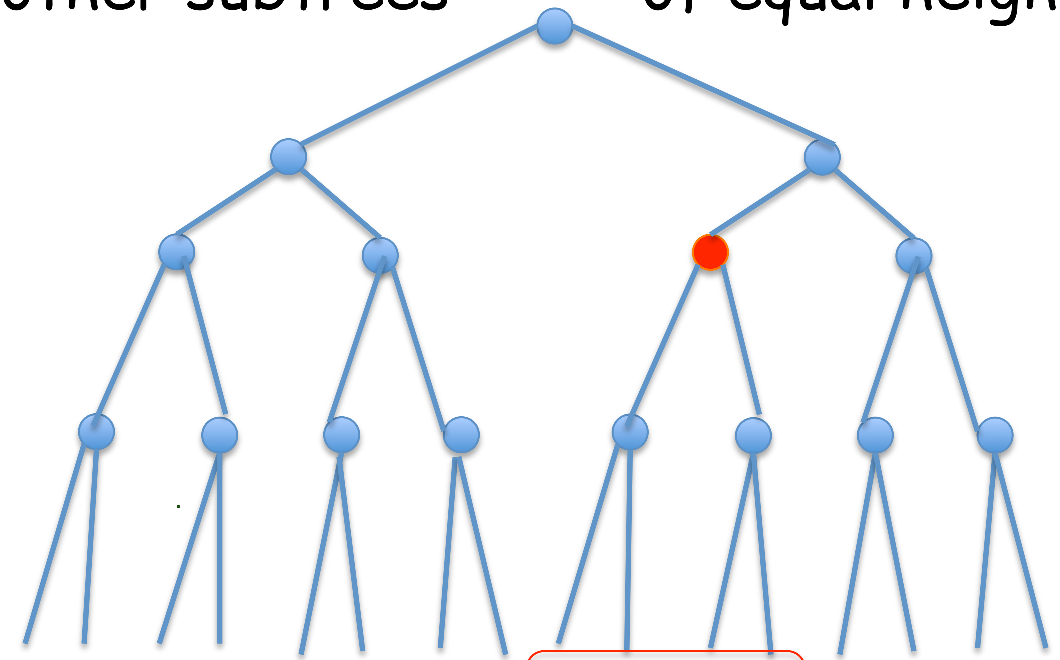
x = 0 1 1 0 0 1 0 1 1 1 0 1 0 0 0 1

- Output: a string z, obtained by copying v’s subtree to the other subtrees of equal height.



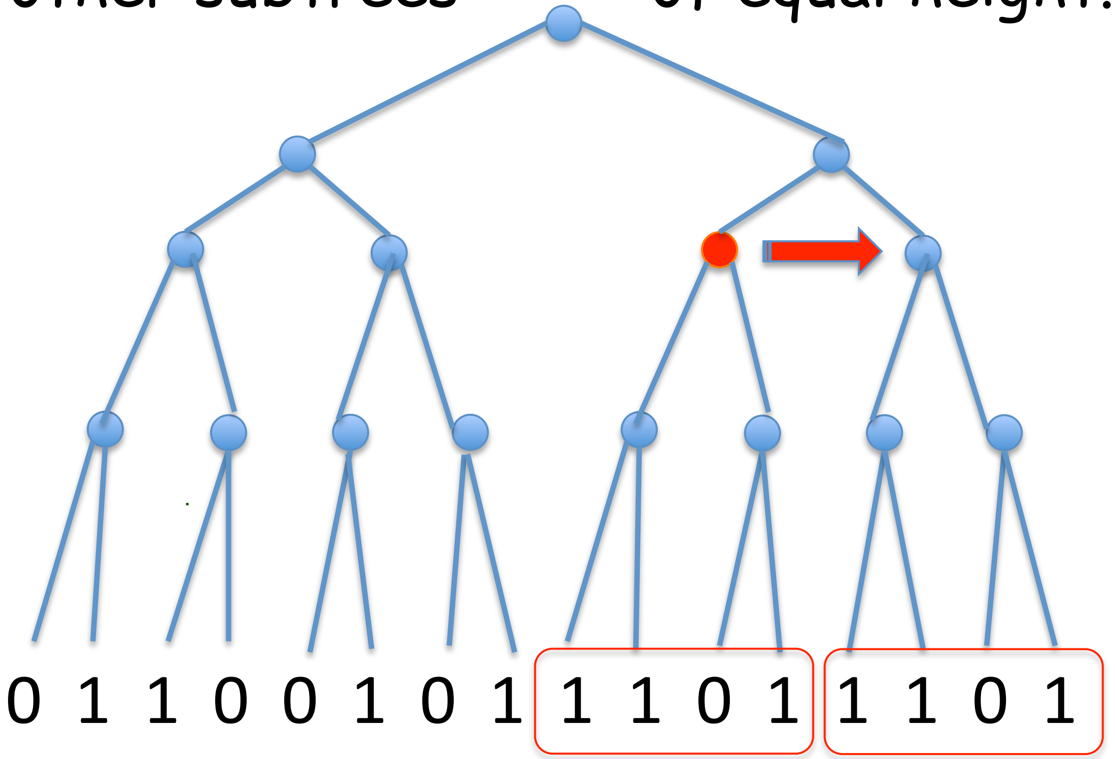
x = 0 1 1 0 0 1 0 1 1 1 0 1 0 0 0 1

- Output: a string z, obtained by copying v's subtree to the other subtrees of equal height.

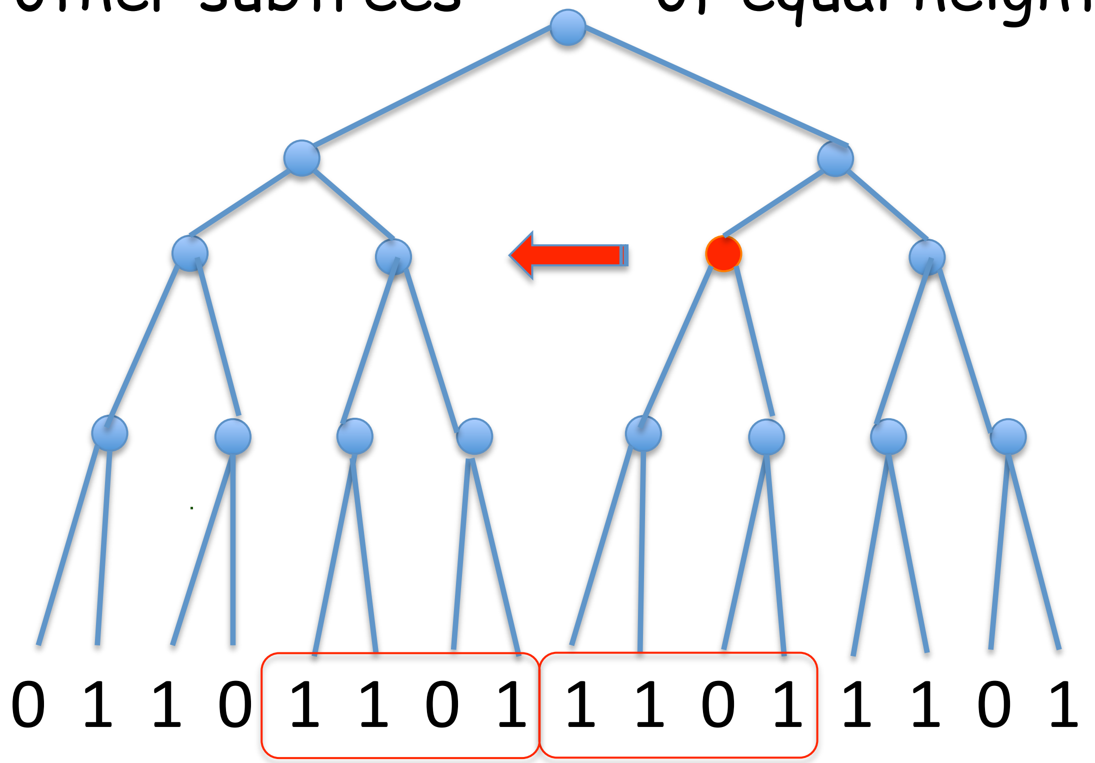


$x = 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ \boxed{1\ 1\ 0\ 1}\ 0\ 0\ 0\ 1$

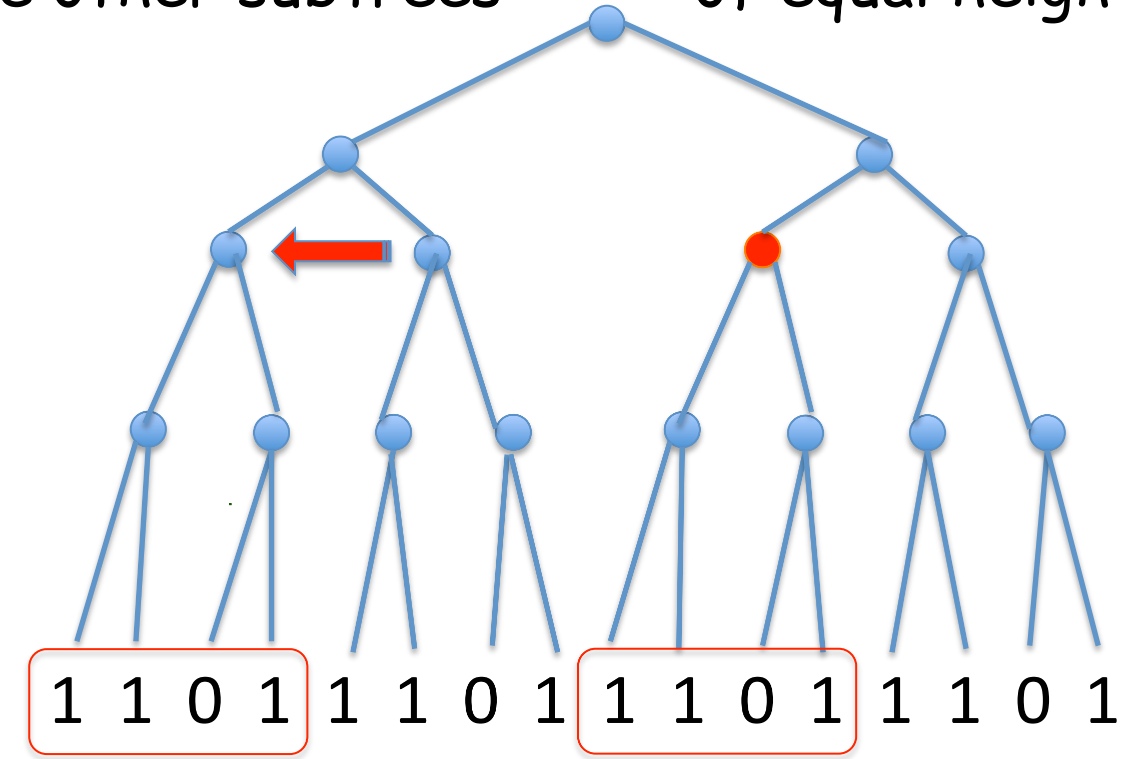
- Output: a string z, obtained by copying v's subtree to the other subtrees of equal height.



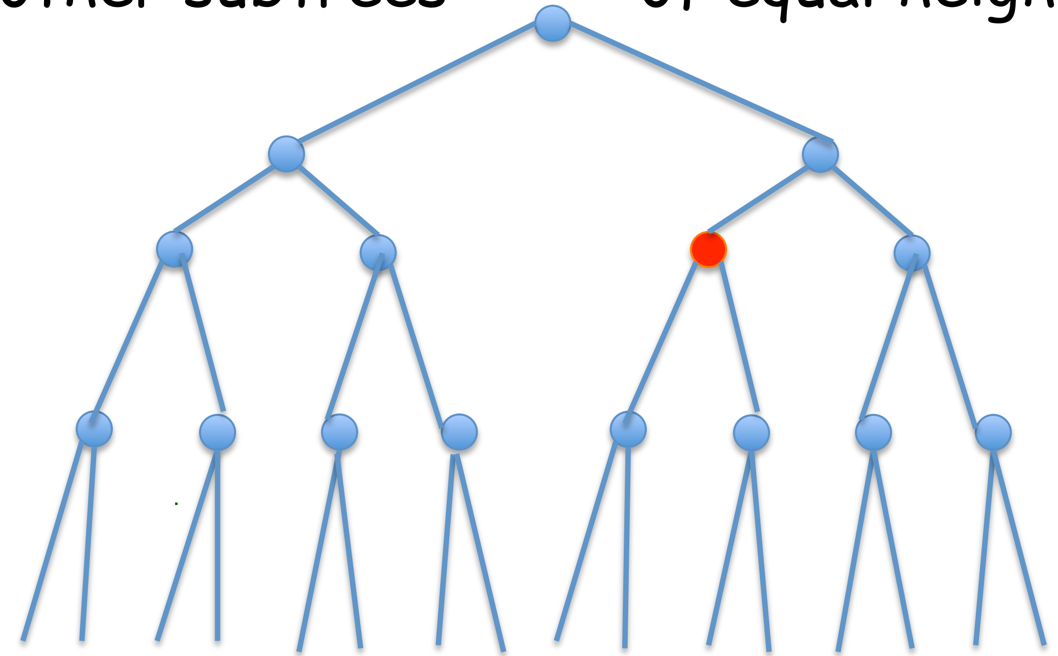
- Output: a string z, obtained by copying v's subtree to the other subtrees of equal height.



- Output: a string z, obtained by copying v's subtree to the other subtrees of equal height.



- Output: a string z, obtained by copying v's subtree to the other subtrees of equal height.



z = 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1

The basic strategy

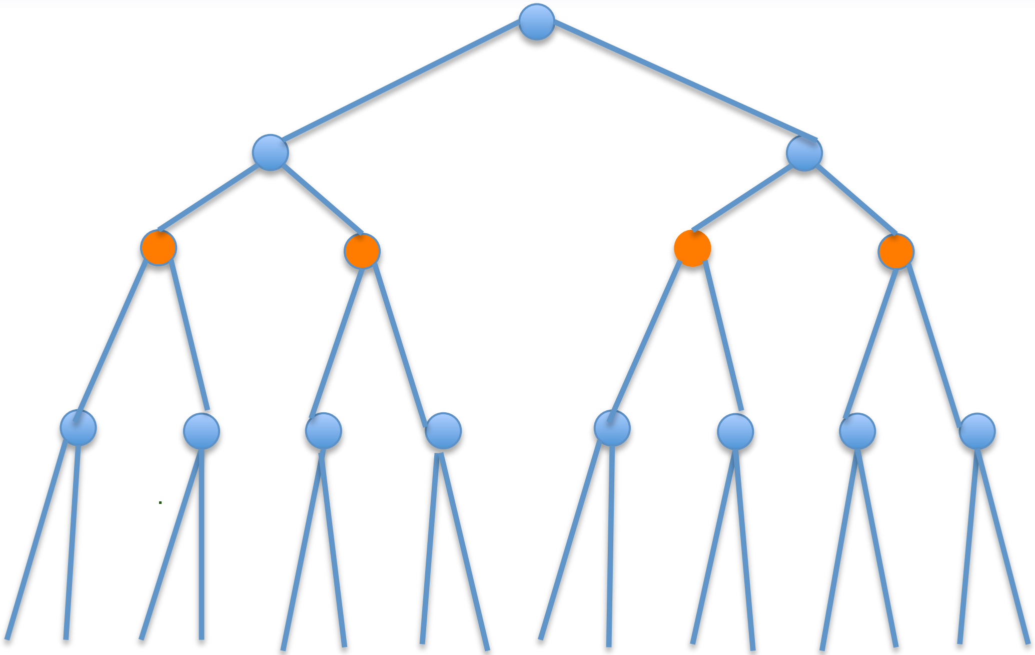
- Idea: this operator "spreads information" from all parts of x to all of z, at multiple scales;

- The node v is encoded as extra input in a way that helps ensure SME property.

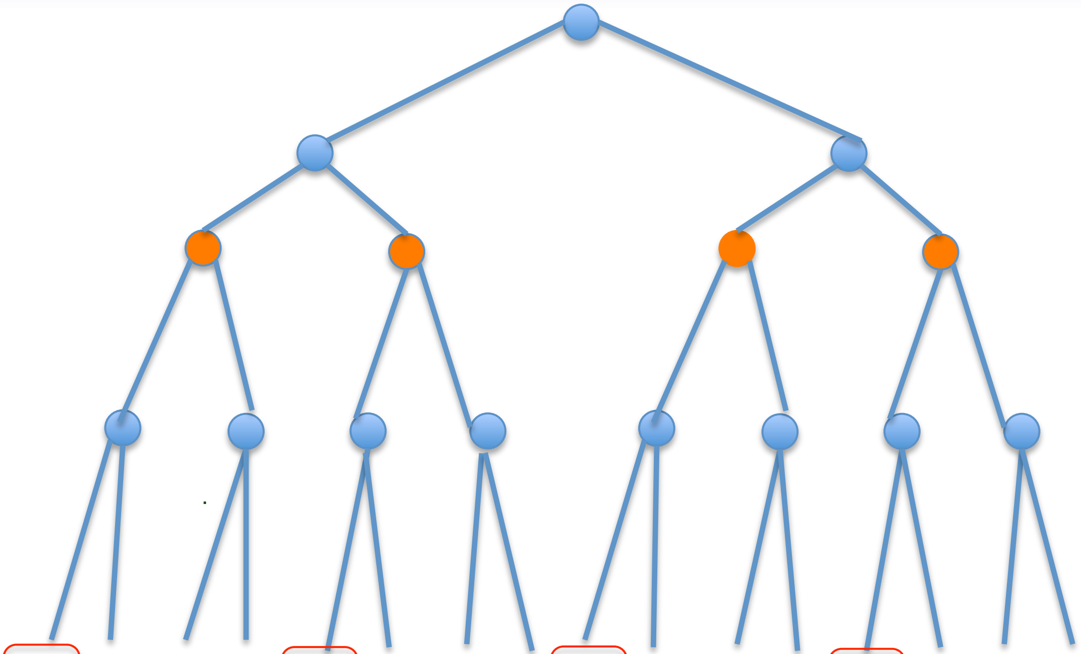
- At the same time, information flow in our tree is restricted, to make easy to implement.

The basic strategy

- Why is Subtree-Copy easy to compute?
- (Glossing many details here…)
- First, simple to compute with $O(n)$ wires, when the height of v is fixed in advance…

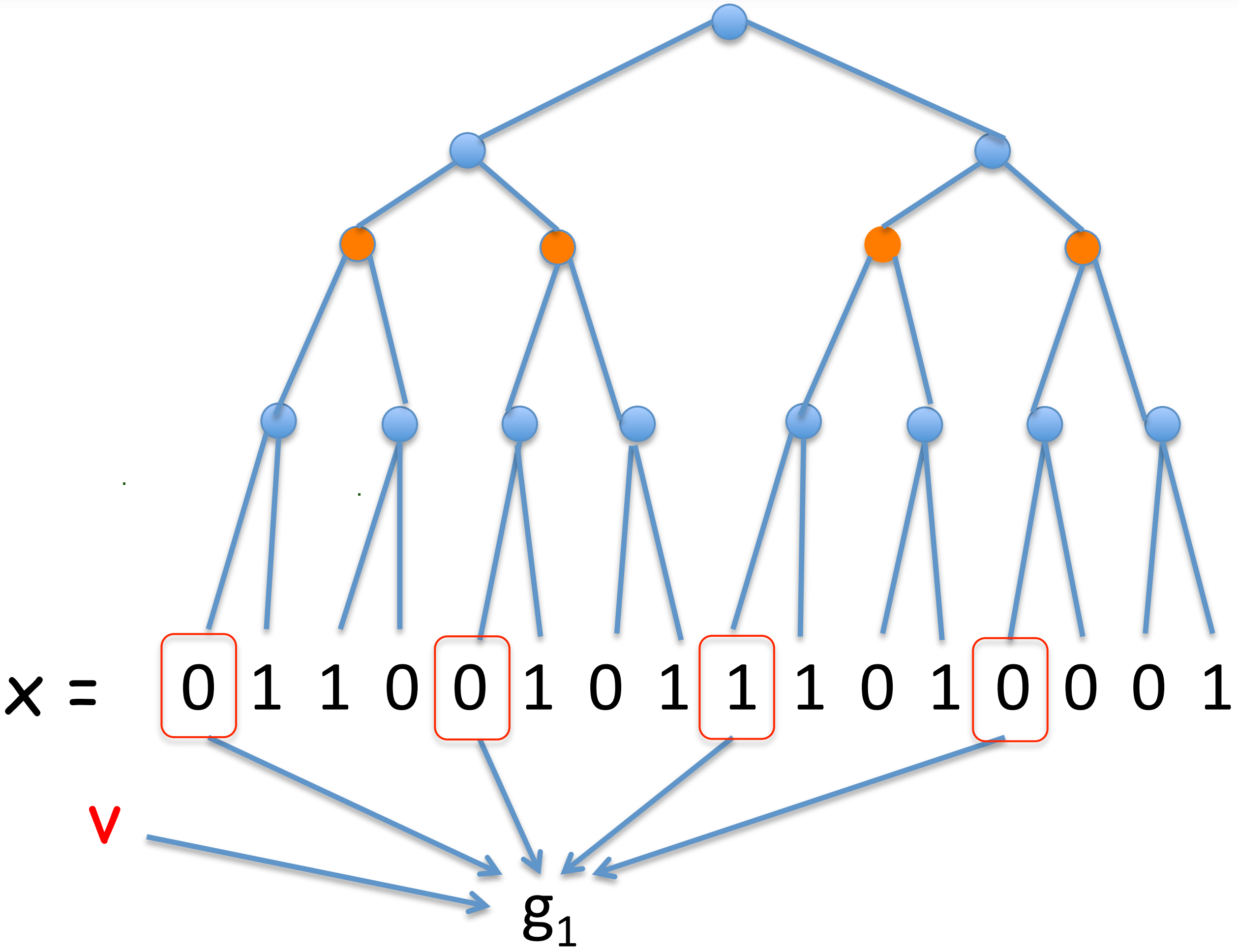


$x = \; 0\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 0\ 1$



$$x = \boxed{0}\ 1\ 1\ 0\ \boxed{0}\ 1\ 0\ 1\ \boxed{1}\ 1\ 0\ 1\ \boxed{0}\ 0\ 0\ 1$$

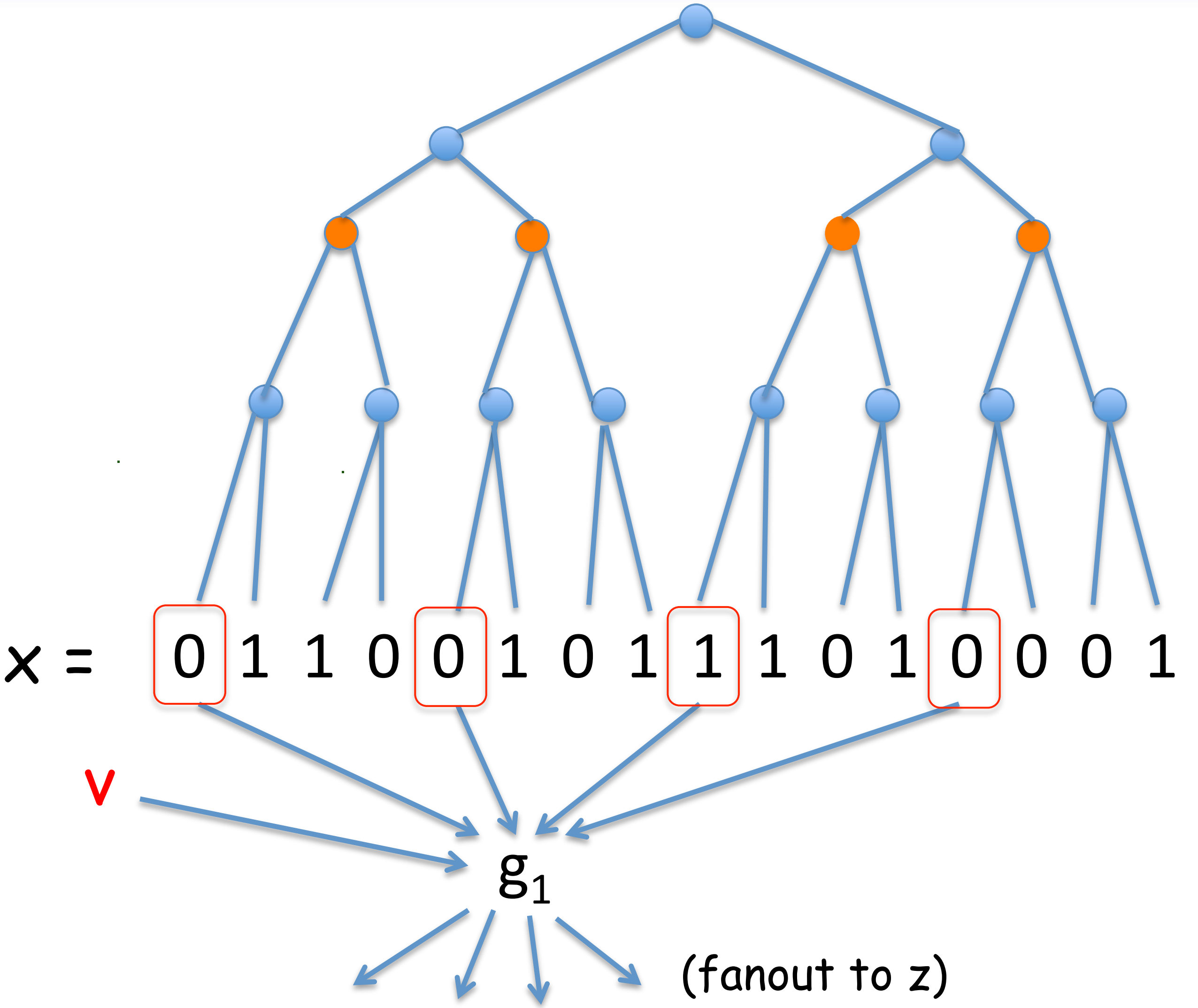
g_1



x = 0 1 1 0 0 1 0 1 1 1 0 1 0 0 0 1

v

g_1

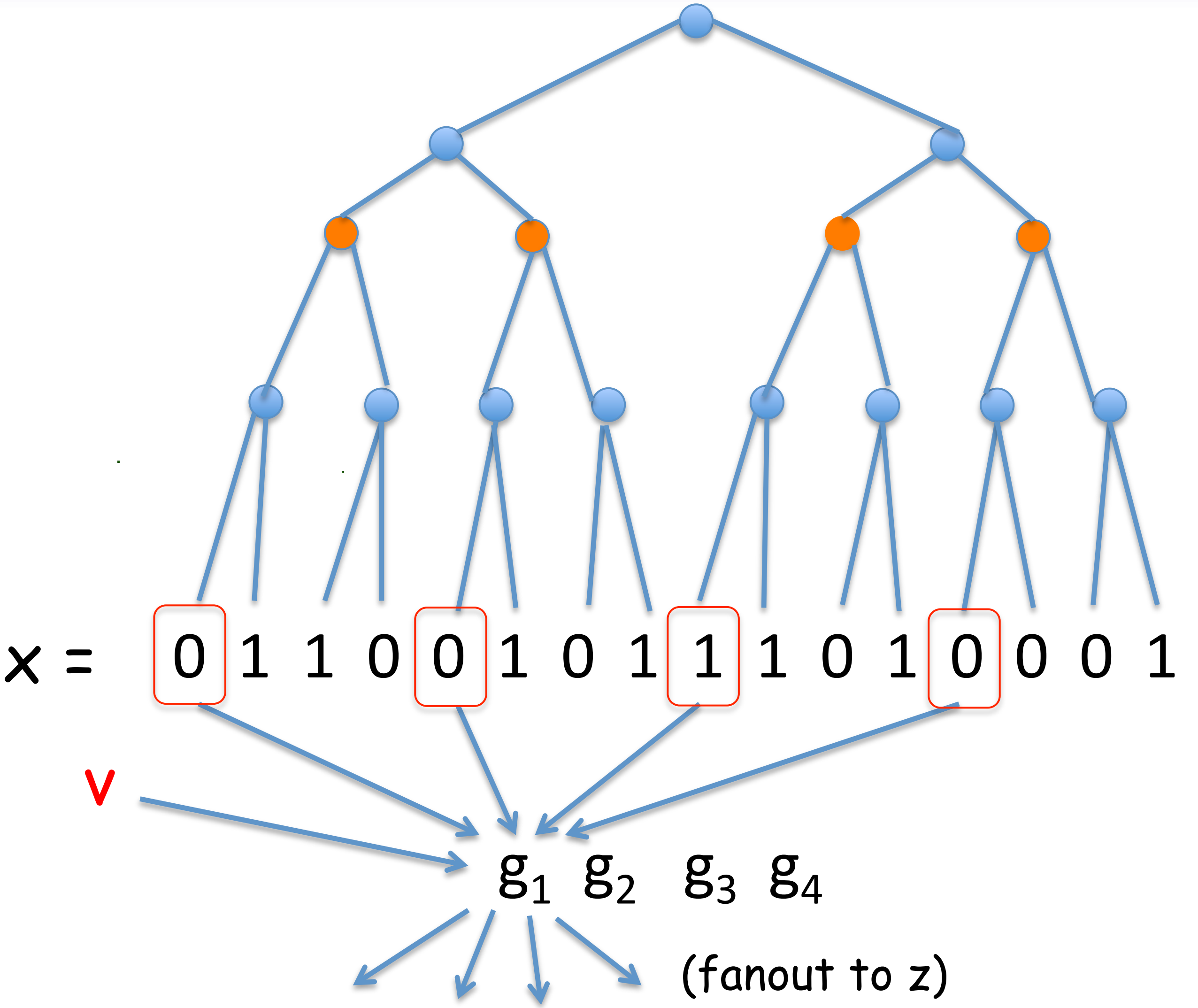


x = 0 1 1 0 0 1 0 1 1 1 0 1 0 0 0 1

v

g_1

(fanout to z)



x = 0 1 1 0 0 1 0 1 1 1 0 1 0 0 0 1

v

g_1 g_2 g_3 g_4

(fanout to z)

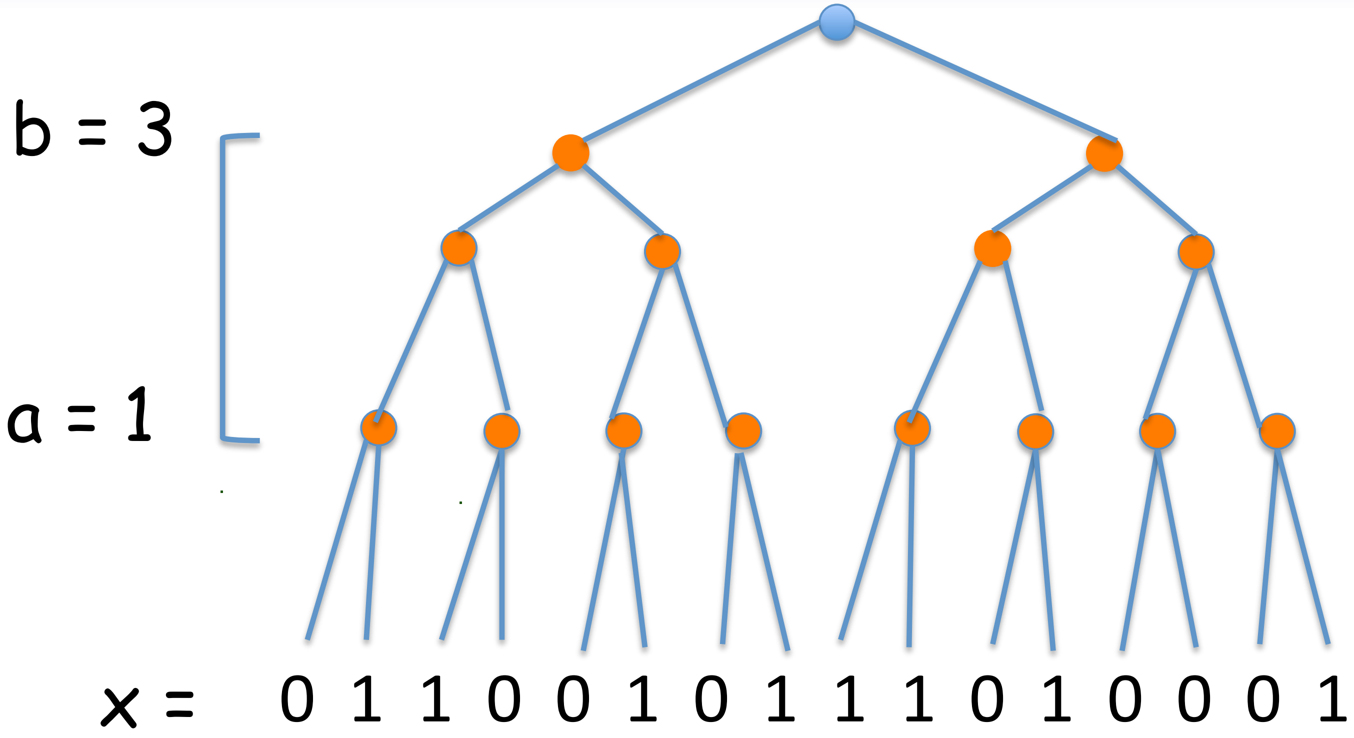
The basic strategy

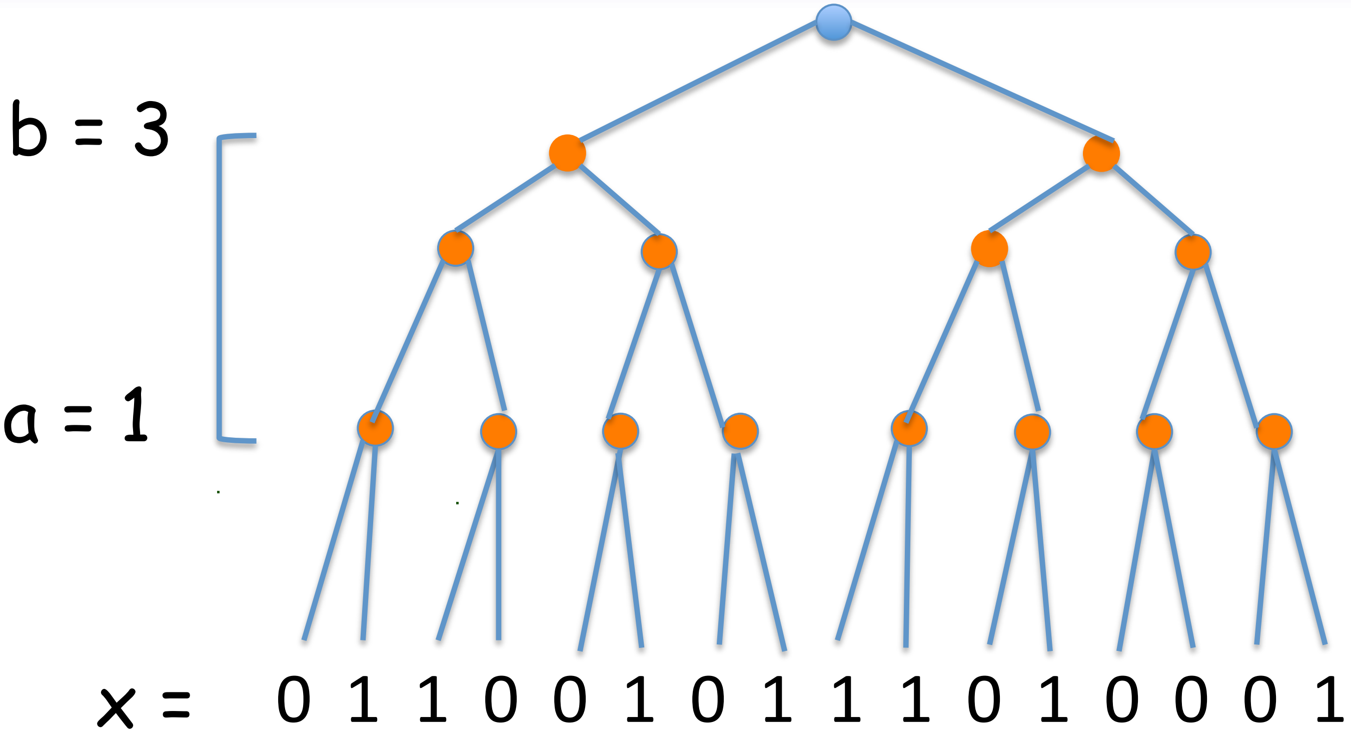
- There are only $\log n$ possible heights of v.

  Using this, can compute Subtree-Copy in depth 3 and $O(n \log n)$ wires.

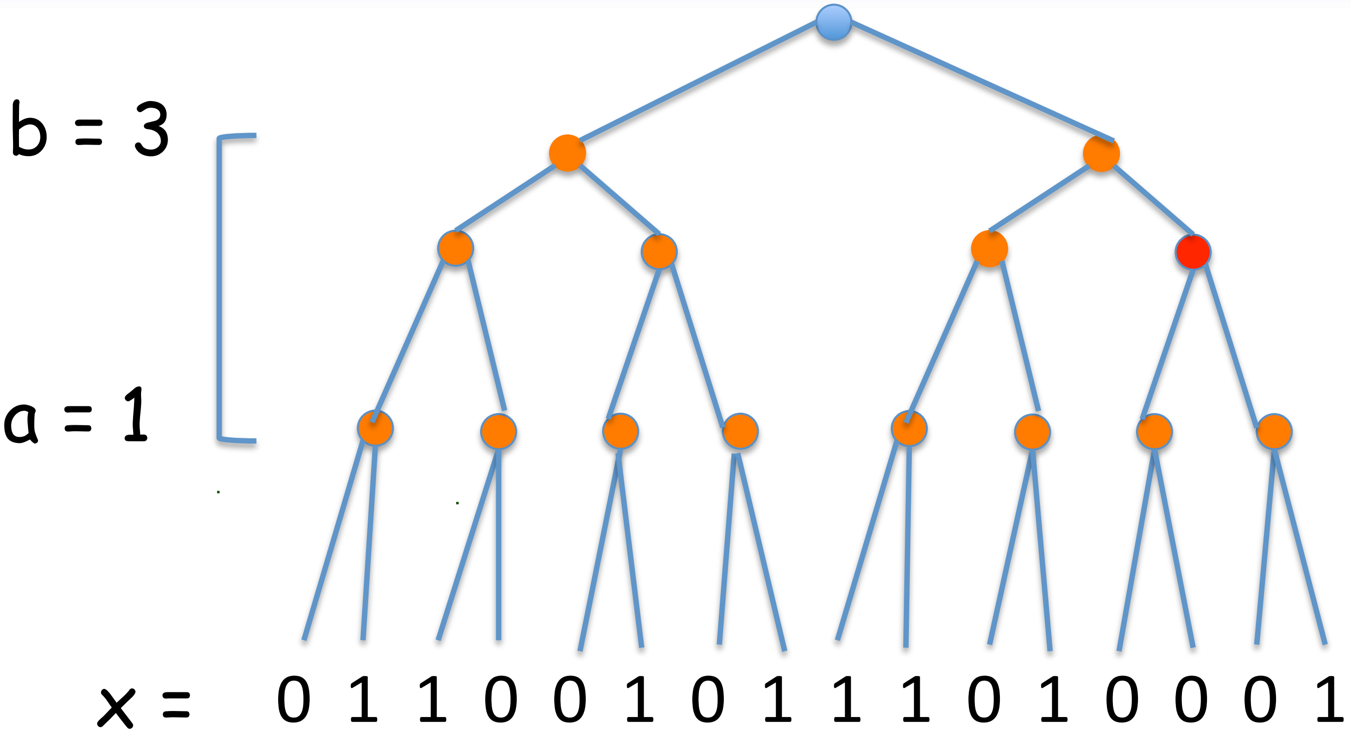
- **Next step:** an inductive construction of more-efficient circuits at higher depths…

- Consider the subproblem where v's height promised to lie in some range $[a, b] \subseteq [\log n]$.

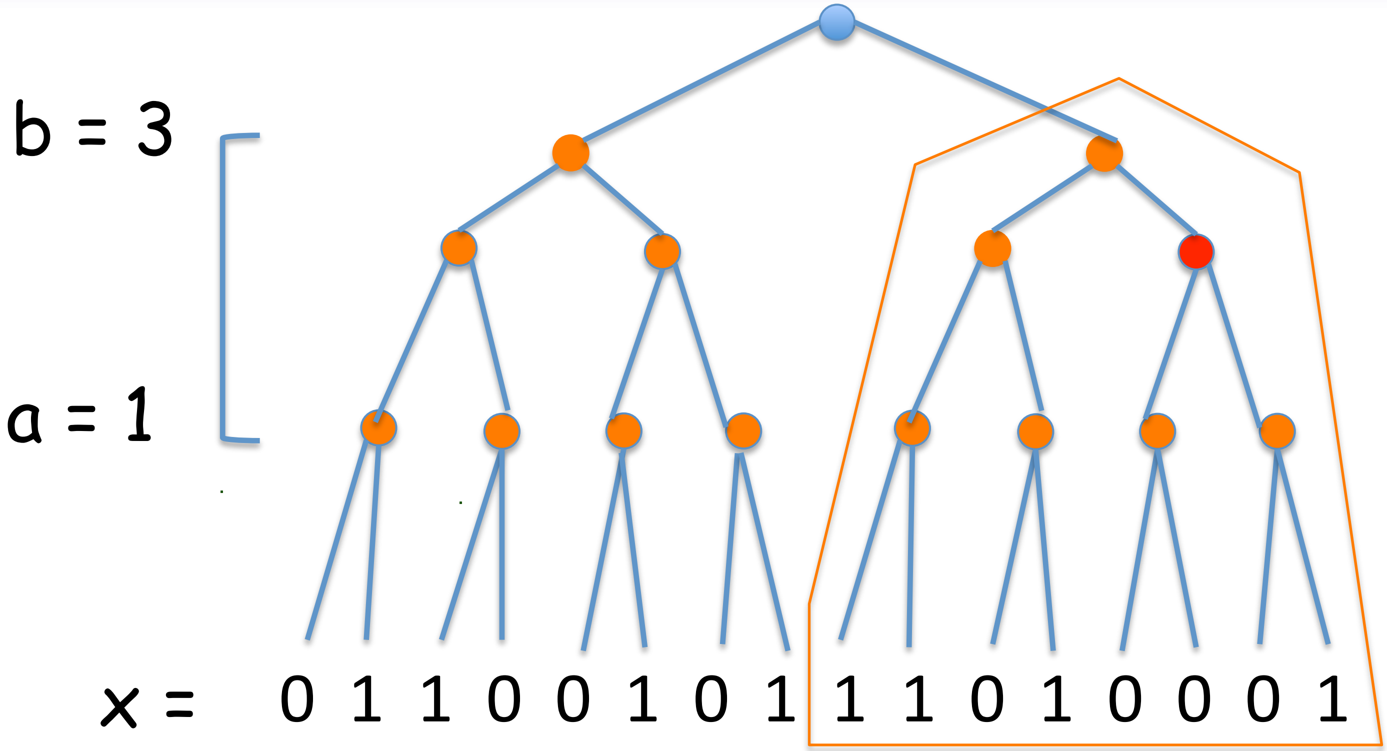

b = 3
a = 1
x = 0 1 1 0 0 1 0 1 1 1 0 1 0 0 0 1



First: "shrink the problem" by extracting the relevant subtree of height b.



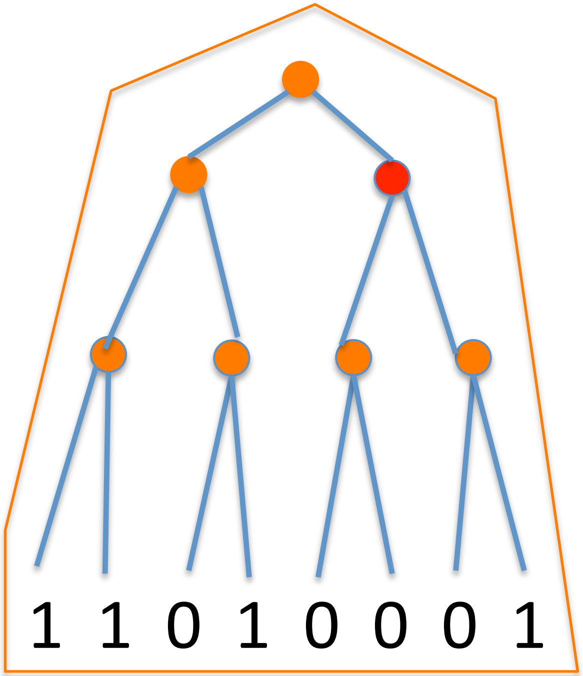
First: "shrink the problem" by extracting the relevant subtree of height b.



First: "shrink the problem" by extracting the relevant subtree of height b.

b = 3

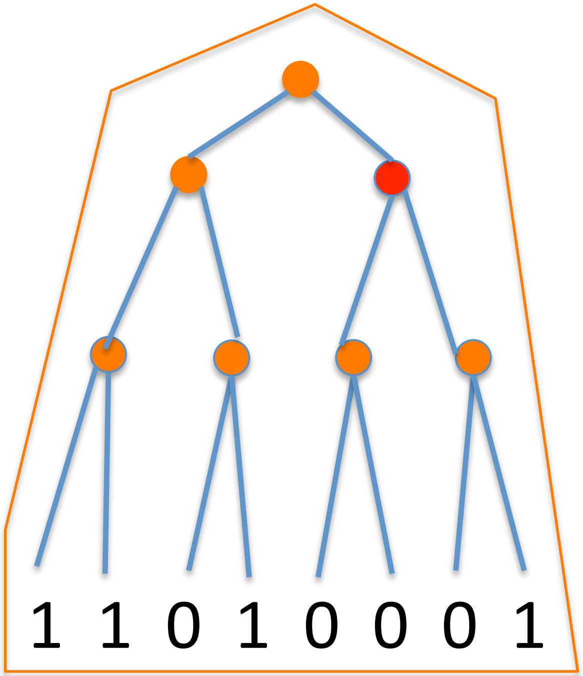
a = 1



First: "shrink the problem" by extracting the relevant subtree of height b.

$b = 3$

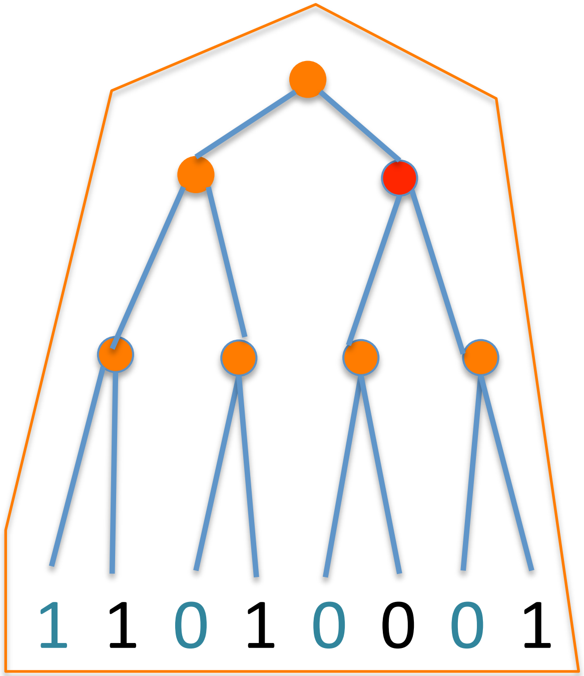
$a = 1$



Now: remainder basically "divides" into 2^a instances of Subtree-Copy, each of height $(b - a)$.

b = 3

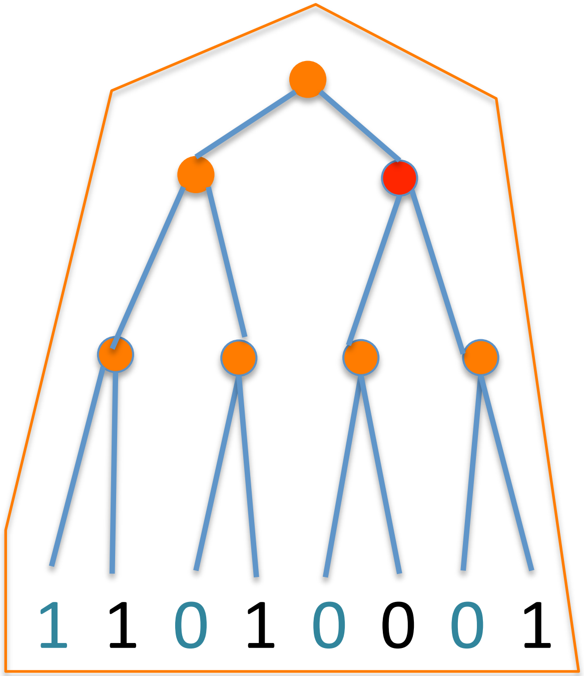
a = 1



Now: remainder basically “divides” into 2^a instances of Subtree-Copy, each of height $(b - a)$.

$b = 3$

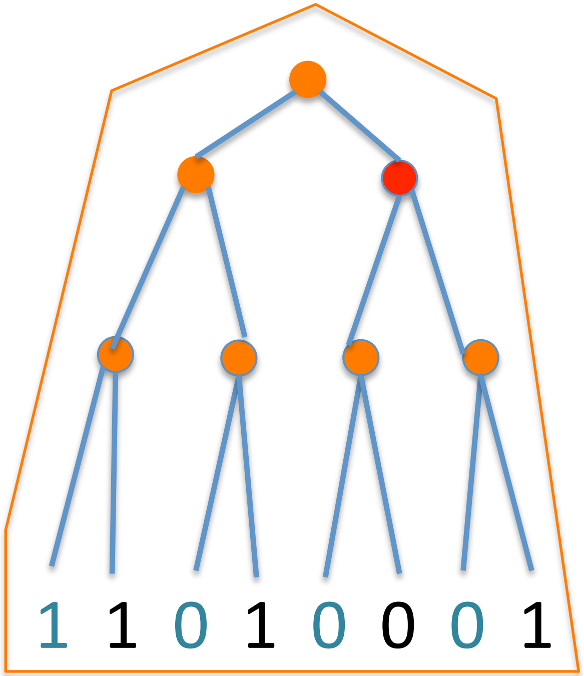
$a = 1$



- Solve these smaller instances inductively, using a lower-depth circuit!

$b = 3$

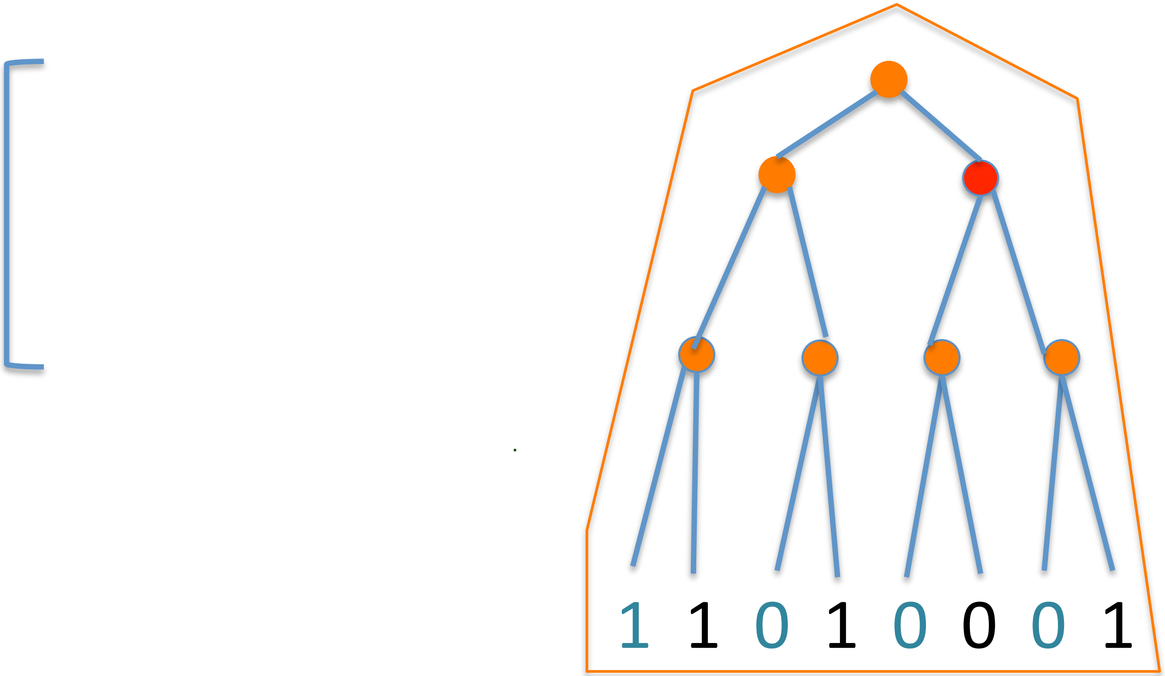
$a = 1$



- Then, "fan out" the result to the rest of **z**.

b = 3

a = 1



- Then, “fan out” the result to the rest of **z**.
- Smaller-size instances → inefficiency hurts us less.

- **Main remaining challenge:** partition the possible heights of v into "buckets" $[a_i, b_i]$, to minimize the wires in resulting circuit.

- Similar sorts of inductive optimizations have been done before, in diff't settings…

  [Dolev et al. '83],

  [Gál, Hansen, Koucký, Pudlák, Viola '12]

Other results

- We prove more results showing that previous, simpler LB criteria do not work beyond depth 2.

  One example:

- Jukna's simplified entropy criterion **[Jukna '10]**: gave elegant proof that naïve GF(2) matrix mult. is asymptotically optimal in depth 2.

- We show: this LB criterion gives no superlinear bound for depth 3.

  -Best lower bounds for d > 2 are connectivity-based

  [Raz, Shpilka '03]

Open questions

- New LB techniques that escape the limitations of known ones?
- Natural proofs-type barriers for LBs in the arbitrary gates, or linear circuits model? **[Aleknovich '03]**
- Draw more connections between the theory of individual Boolean function complexity, and that of joint complexity? **[Baur-Strassen '83; Vassilevska Williams, Williams '10]**

Thanks!