15-859: Information Theory and Applications in TCS **Carnegie Mellon University**

Lecture 16: Monotone Formula Lower Bounds via Graph Entropy

March 26, 2013

Lecturer: Mahdi Cheraghchi *Scribe: Shashank Singh*

1 Recap

- **Graph Entropy:** Given $G = (V, E)$, we define $H(G) = \min(X; Y)$ over joint distributions (X, Y), where $X \in V$ is uniformy random and $X \in Y \subseteq V$. We showed Graph Entropy obeys the following:
  - **Sub-additivity:** $H(G_1 \cup G_2) \leq H(G_1) + G(G_2)$.
  - **Monotonicity:** If $G_1 \subseteq G_2$, then $H(G_1) \leq G(G_2)$.
  - **Disjoint Union:** If $G_1, \ldots, G_k$ are the connected components if G, then $H(G) = \sum_i \frac{|V(G_i)|}{|V(G)|} H(G_i)$.
- Last time, we applied Graph Entropy to lower bound the size of
  - a covering of a graph by bipartite graphs
  - a perfect family of hash functions

2 Monotone Formula Lower Bounds via Graph Entropy

Today we examine an application of Graph Entropy to Circuit Complexity.

2.1 Monotone Boolean Functions

Definition 1 *A boolean function is one mapping* $\{0, 1\}^n \to \{0, 1\}$.

Remark 2 *We can equivalently consider boolean functions as mapping* $\mathcal{P}([n]) \to \{0, 1\}$*, using the obvious bijection between* $\{0, 1\}^n$ *and* $\mathcal{P}([n])$*. Boolean functions are represented by (not necessarily unique) boolean formulae or trees in which leaves variables and internal nodes are logical connectives. We use these representations interchangeably.*

Definition 3 *A boolean function* $f : \mathcal{P}([N]) \to \{0, 1\}$ *is monotone if* $S \subseteq T \in \mathcal{P}([n])$ *implies* $f(S) \leq f(T)$*. Furthermore, if* f *is a monotone boolean function, then the min-terms of* f *of size* i *are*

$$(f)_i \stackrel{\triangle}{=} \{S \in \mathcal{P}([n]) : |S| = i, f(S) = 1, \text{ and } \forall T \subseteq S, f(T) = 0\}, \quad (f) \stackrel{\triangle}{=} \bigcup_{i=1}^{n} (f)_i.$$

Furthermore, a boolean formula is monotone if it contains only AND and OR connectives.

Example 4 The following are monotone boolean functions:

1. OR: $x \vee y = 0 \Leftrightarrow x = y = 0$. The min-terms of OR are $(OR)_1 = \{\{0\}, \{1\}\}$, $(OR)_i = \emptyset$ for $i \neq 1$.
2. AND: $x \wedge y = 1 \Leftrightarrow x = y = 1$. The min-terms of AND are $(AND)_2 = \{\{0, 1\}\}$, $(AND)_i = \emptyset$ for $i \neq 2$.
3. MAJ$_3$: $MAJ_3(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_1 \wedge x_3) \vee (x_2 \wedge x_3)$.

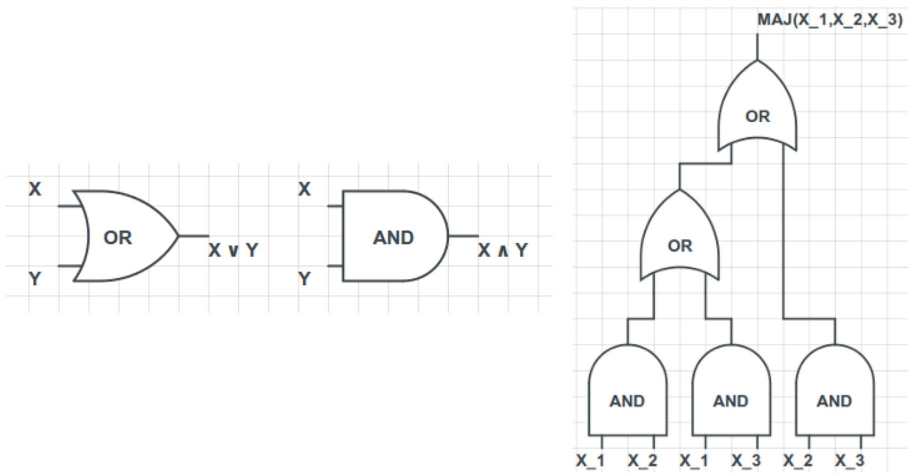

Figure 1: Tree representations of AND, OR, and MAJ_3

Proposition 5 *A boolean function is monotone iff it can be represented by a monotone boolean formula.*

Proof: Clearly, monotone boolean formulae compute monotone boolean functions.

Let f be a monotone boolean function. Then, $W \subseteq \mathcal{P}([n])$, $F(W) = 1$ if and only if $\exists S \in (f)$ with $S \subseteq W$. It follows that f is defined uniquely by (f) as follows:

$$f(x_1, \dots, x_n) = \bigvee_{S \in (f)} \left(\bigwedge_{j \in S} x_j \right), \quad \forall (x_1, \dots, x_n) \in \{0,1\}^n.$$

Note that this formula, called the Disjunctive Normal Form (DNF) of f, is also represented by a binary tree, since many-input logic gates can be simulated by (linearly many) two-input gates. ■

2.2 Size of a Boolean Function and Threshold Functions

Definition 6 *The size* $\mathrm{size}(\phi)$ *of a formula ϕ is the number of nodes in the tree representation of ϕ. The size of a boolean function f is*

$$\mathrm{size}(f) \triangleq \min_{\phi \text{ computing } f} \mathrm{size}(\phi).$$

That is, $\mathrm{size}(f)$ *is number of nodes in the smallest tree computing f.*

Definition 7 *For $k \in [n]$, the threshold function $Th_k^n : \mathcal{P}([n]) \to \{0,1\}$ is defined for $S \in \mathcal{P}([n])$ by*

$$Th_k^n(S) \triangleq \begin{cases} 1 & \text{if } |S| \geq k \\ 0 & \text{else} \end{cases}.$$

Example 8 Threshold functions generalize AND, OR, and MAJ:

$$\begin{aligned} AND &= Th^n_n & \text{size}(AND) &= 2n-1 \\ OR &= Th^n_1 & \text{size}(OR) &= 2n-1 \\ MAJ &= Th^n_{\lceil n/2 \rceil} \end{aligned}$$

It can be shown that MAJ is the 'most complex' threshold, in that it maximizes size(Th^n_k) over k.

2.3 Bounding the Size of Threshold Functions

Consider the problem of bounding size(Th^n_k). For general k, the bound size$(Th^n_k) \in O(n^{5.3})$ due to (Valiant, 1984) is known, based on a probabilistic construction which we do not give here.

We analyze the case $k = 2$, for which the following upper bound is easy to demonstrate:

Claim 9 size$(Th^n_2) \in O(n^2)$.

Proof:

$$(Th^n_2)_2 = \{\{i,j\} \in \mathcal{P}([n]) : i \neq j\}.$$

Furthermore, $\forall i \neq 2$, $(Th^n_2)_i = \emptyset$. Thus, the DNF of Th^n_2 is

$$Th^n_2(x_1, \ldots, x_n) = \bigvee_{\substack{\{i,j\} \in \mathcal{P}([n]) \\ i \neq j}} x_i \wedge x_j,$$

which (since size(AND), size$(OR) \in O(n)$) indicates size$(Th^n_2) \in O(n^2)$. ■

Remark 10 *Consider the following Divide and Conquer construction:*

Divide the input string $x = (x_1, \ldots, x_n) \in \{0,1\}^n$ into $y = (x_1, \ldots, x_{\lceil n/2 \rceil})$ and $z = (x_{\lfloor n/2 \rfloor}, \ldots, x_n)$. Then, we have the recursive formula

$$Th^n_2(x) = Th^{\lceil n/2 \rceil}_2(y) \vee Th^{\lfloor n/2 \rfloor}_2(z) \vee (Th^{\lceil n/2 \rceil}_1(y) \wedge Th^{\lfloor n/2 \rfloor}_1(z)).$$

This recurrence gives an upper bound: defining $S_n = \text{size}(Th^n_2)$, the recurrence gives

$$S_n \leq 2S_{n-1} + O(n),$$

since clearly size$(Th^n_1) \in O(n)$. *The solution of this standard recurrence (think mergesort) is*

$$S_n \leq (2n + \lceil \log n \rceil + 1)(\lceil \log n \rceil) \quad \text{and so} \quad S_n \in O(n \log n).$$

Exercise: Refine this bound to size$(Th^n_2) \leq 2n\lceil \log n \rceil - 1$. The lower bound we now give shows this is tight.

We now apply Graph Entropy to prove the lower bound size$(Th^n_2) \geq 2\lceil n \log n \rceil - 1$, following (Newman, Ragde, and Wigderson 1990). In order to use graph entropy we're going define a graph G_f for a boolean function f. Consider defining the following:

Definition 11

$$G_f \triangleq (V, E), \quad \text{where} \quad V \triangleq [n], \quad \text{and} \quad E \triangleq (f)_2.$$

Note that n is from Th^n_2 and is not necessarily the number of variables in f.

Example 12 $G_{Th^n_2} = K_n$. For a single variable x_i, G_{x_i} is the empty graph on n vertices.

It helps now to have a few lemmas about how graph entropy evolves with AND and OR operations.

Lemma 13 *Suppose $f = g \vee h$. Then, $G_f \subseteq G_g \cup G_h$, and hence $H(G_f) \leq H(G_g) + H(G_h)$.*

Proof: Suppose $e = \{i, j\} \in E(G_f)$. Then, $1 = f(e) = g(e) \vee h(e)$; without loss of generality, $g(e) = 1$. By construction of G_f, $e \in (f)_2$, so $f(\{i\}) = f(\{j\}) = 0$. Then, $g(\{i\}) = g(\{j\}) = 0$, so $e \in (g)_2 = E(G_g)$. ■

It would be nice if we also had this property for AND, but it doesn't hold, as the following example shows:

Example 14 Suppose $g(x_1, x_2) = x_1, h(x_1, x_2) = x_2$. Then, $\{1, 2\} \in E(G_f)$, but $\{1, 2\} \notin E(G_g), E(G_h)$.

Thus, we need a weaker statement:

$$G_{g\wedge h} \subseteq G_g \cup G_h \cup T_{g,h}.$$

Lemma 15 *$T_{g,h}$ is the subgraph of G_f induced by edges in*

$$(g)_1 \triangle (h)_1 \stackrel{\triangle}{=} ((g)_1 - (h)_1) \times ((h)_1 - (g)_1).$$

Proof: Let $e = \{i, j\}$, and let $f', g', h' : \{0, 1\}^2 \to \{0, 1\}$ denote the restrictions of f, g, h, respectively, to e (since the formulae are monotone, we can think of this as setting the other coordinates to 0). Then, we have

$$\left.\begin{array}{r} f'(x_i, x_j) = x_i \wedge x_j \\ g'(x_i, x_j) \neq x_i \wedge x_j \\ h'(x_i, x_j) \neq x_i \wedge x_j \\ f' = g' \wedge h' \\ \text{by inspection} \left\{ \begin{array}{l} g'(x_1, x_2) \neq x_i \vee x_j \\ h'(x_1, x_2) \neq x_i \vee x_j \end{array}\right. \end{array}\right\} \Rightarrow \text{ possible cases are } \left\{ \begin{array}{ll} g' = x_i, & h' = x_j \\ g' = x_j, & h' = x_i \end{array}\right.$$

(here, we make the simplifying assumption that f, g, h are non-constant functions; these cases can be analyzed separately) which in turn implies that $e \in ((g)_1 - (h)_1) \times ((h)_1 - (g)_1) \stackrel{\triangle}{=} (g)_1 \triangle (h)_1$. ■

Remark 16 *Since $(g)_1 - (h)_1$ and $(h)_1 - (g)_1$ are disjoint, $T_{g,h}$ is bipartite. We showed in a previous lecture that this implies $H(T_{g,h}) \leq 1$. Using subadditivity and the facts*

$$\begin{array}{rl} H(G_{g\vee h}) & \leq H(G_g) + H(G_h) \\ H(G_{g\wedge h}) & \leq H(G_g) + H(G_h) + 1 \\ H(G_{x_i}) & = 0 \\ H(G_{Th_2^n}) = H(K_n) & = \log n, \end{array}$$

we see that any monotone formula for Th_2^n has at least $\lceil \log n \rceil$ AND gates, and hence $\mathrm{size}(Th_2^n) \geq \lceil \log n \rceil$.

We can get an even tighter lower bound by tightening the upper bound on $H(T_{g,h})$:

Observe that, while $V(T_{g,h}) = [n]$, $E(T_{g,h}) \subseteq (g)_1 \triangle (h)_1$, which implies, by the disjoint union property,

$$H(T_{g,h}) \leq \frac{|(g)_1 \triangle (h)_1|}{n}.$$

Let's define a potential function:

Definition 17

$$\mu(f) \stackrel{\triangle}{=} H(G_f) + \frac{|(f)_1|}{n}.$$

Claim 18 *For both $f = g \vee h$ and $f = g \wedge h$, $\mu(f) \leq \mu(g) + \mu(h)$.*

Proof: Case 1: $f = g \vee h$. Assuming no gate computes a constant function,

$$(f)_1 = \{i : f(\{i\}) = 1\} = (g)_1 \cup (h)_1.$$

Thus,

$$\mu(f) = H(G_f) + \frac{|(f)_1|}{n} \leq H(G_h) + H(G_h) + \frac{|(g)_1| + |(h)_1|}{n} = \mu(h) + \mu(g).$$

Case 2: $f = g \wedge h$. This time, $(f)_1 = (g)_1 \cap (h)_1$. Thus,

$$\begin{aligned}
\mu(f) = H(G_f) + \frac{|(f)_1|}{n} &\leq H(G_h) + H(G_h) + H(T_{g,h}) + \frac{|(g)_1 \cap (h)_1|}{n} \\
&\leq H(G_h) + H(G_h) + \frac{|(g)_1 \triangle (h)_1|}{n} + \frac{|(g)_1 \cap (h)_1|}{n} \\
&= H(G_h) + H(G_h) + \frac{|(g)_1| + |(h)_1|}{n} = \mu(g) + \mu(h).
\end{aligned}$$

■

Note that each leaf has $\mu(x_i) = \frac{1}{n}$ and the root has $\mu(Th_2^n) = \log n$. Hence, by subadditivity and the preceding claim, there must be at least $\lceil n \log n \rceil$ leaves. Since each gate has two inputs and one output, it follows that there are at least $\lceil n \log n \rceil - 1$ internal nodes, for a total lower bound of

$$\text{size}(Th_2^n) \geq 2\lceil n \log n \rceil - 1.$$