5. Depth Reduction for Algebraic Formulas

Saturday, September 2, 2023 2:56 PM

We will prove: (Depth reduction for formulas) [Brent '73]

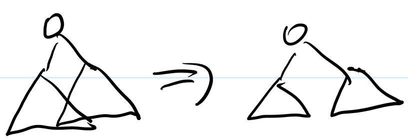
Thm 1: If f is computed by an algebraic formula of size s, then it is computed by an algebraic formula of depth $O(\log s)$ (and hence of size $\mathrm{poly}(s)$).

Def: $VNC^k = \{f : f \in \mathbb{F}[x_1, \dots, x_n]$ is computed by a circuit of size $\mathrm{poly}(n)$ and depth $O(\log^k n)$, and $\deg(f) \le \mathrm{poly}(n)\}$.

Cor: $VF = VNC^1$

Pf: By Thm 1, $f \in VF$ is computed by a formula (and hence a circuit) of depth $O(\log n)$ and size $\mathrm{poly}(n)$. So $VF \subseteq VNC^1$.

Conversely, for $f \in VNC^1$, we may turn a circuit of depth $O(\log n)$ into a formula of depth $O(\log n)$, and increase its size by at most a factor of $2^{O(\log n)} = \mathrm{poly}(n)$. So $VNC^1 \subseteq VF$.



$\square$

proved later

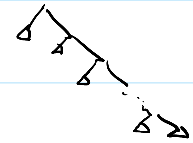
Thm 2 (Depth reduction for circuits) [Valiant-Skyum-Berkowitz-Rackoff '83]
If f is computed by a circuit of size s and $\deg(f) = d$, then f is computed by a circuit of size $\mathrm{poly}(s, d)$ and depth $O(\log d \cdot (\log d + \log s))$.

Cor: $VP = VNC^2$.

Depth reduction for formulas.

The underlying graph of a formula is a tree.

Its depth is $\gg \log(\text{size})$ if it is "highly unbalanced":

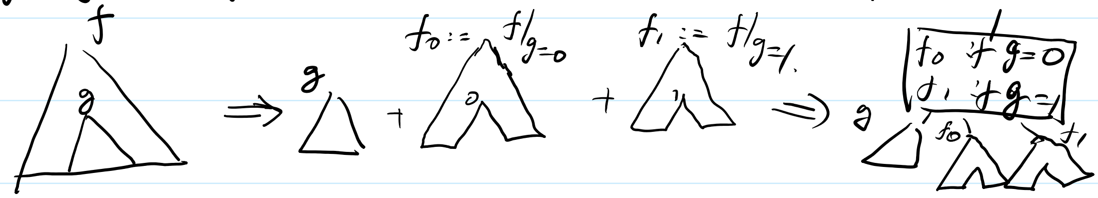


Idea: balancing the tree

Same idea was originally used for Boolean formulas [Spira '71, Khrapchenko '60s]

Same idea was originally used for Boolean formulas [Spira '71, Khrapchenko '60s]

Boolean case



Then recurse.

In the algebraic setting, we can't use 0/1, but a similar idea works.

Observe that u is an affine function in v, since v is used only once in the formula computing u. So we may decompose u into the "linear part" and the "constant part."

Pf of Thm 1: We argue that we can balance the formula T_f computing f such that the subtrees of the output gate have size $\le c\cdot s$ for some $c<1$, where $s = \mathrm{size}(T_f)$. Then recurse.

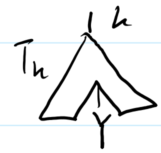
Claim: We may choose a gate $g \in T_f$ such that the subtree rooted at g (T_g) has size $\in [\frac{1}{3}s, \frac{2}{3}s]$.

T_f: 

Pf of the claim: Choose g s.t. $\mathrm{size}(T_g) \ge \frac{1}{3}s$.

If $\mathrm{size}(T_g) > \frac{2}{3}s$, then for at least one children g' of g the subtree $T_{g'}$ at g' has size $\ge \frac{1}{3}s$. Replace g by g' and repeat. $\square$

(Abuse the notation and use $f, g \in \mathbb{F}[x_1, \dots, x_n]$ to denote both the gates and their corresponding polynomials.)

Let $h \in \mathbb{F}[x_1, \dots, x_n, y]$ be the polynomial computed by T_h: 

← also a formula.

Then $f = h(x_1, \dots, x_n, g(x_1, \dots, x_n))$

$\frac{1}{3}s \le \mathrm{size}(T_h), \mathrm{size}(T_g) \le \frac{2}{3}s$ by the choice of g.

As T_h is a formula, h is an affine function in Y.

i.e., $h = h_1 Y + h_0$ for some $h_0, h_1 \in \mathbb{F}[x_1, \dots, x_n]$

$\frac{2}{3}s$

i.e., $h = h_1 \cdot Y + h_0$ for some $h_0, h_1 \in \mathbb{F}[X_1, \dots, X_n]$

Note $h_0 = h(X_1, \dots, X_n, 0)$, $\Rightarrow$ h_0 is computed by a formula of size $\le \frac{2}{3}s$

$h_1 = \frac{\partial h}{\partial Y}$.

Consider a gate ℓ in T_h. If $\ell = Y$, then $\frac{\partial \ell}{\partial Y} = 1$.

If Y is not in the subtree T_ℓ rooted at ℓ, then $\frac{\partial \ell}{\partial Y} = 0$

If $\ell = \ell_1 + \ell_2$, $Y \in T_{\ell_1}$, then $\frac{\partial \ell}{\partial Y} = \frac{\partial \ell_1}{\partial Y}$.

If $\ell = \ell_1 \cdot \ell_2$, $Y \in T_{\ell_1}$, then $\frac{\partial \ell}{\partial Y} = \frac{\partial \ell_1}{\partial Y} \cdot \ell_2$.

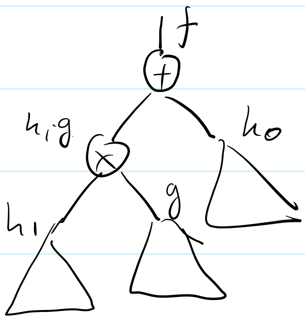
As pointed out by Pooya, we can simply compute h_1 as $h(X_1, \dots, X_n, 1) - h_0$

$= h(X_1, \dots, X_n, 1) - h(X_1, \dots, X_n, 0)$

each part of size $\le \frac{2}{3}s$.

By induction, h_1 is computed by a formula of size $\le \text{size}(T_h) \le \frac{2}{3}s$ in $X_1, \dots, X_n$.

As $f = h(X_1, \dots, X_n, g)$ and $h = h_1 Y + h_0$, $f = h_1 \cdot g + h_0$.



formula sizes for g, h_0, h_1 are $\le \frac{2}{3}s$ $\square$

Removing division gates in formulas.

Thm Suppose f is computed by a formula of size s with division gates, and $d = \deg(f)$. If $|\mathbb{F}| > s(d+1)$, then f can be computed by a formula of size $\text{poly}(s, d)$ without division gates.

In general, f can be computed by a formula of size $(sd)^{O(\log \log_{|\mathbb{F}|}(sd))}$.

Pf: Proceed as before. Compute f as $\text{Hom}_{\le d}(g(1+t+\dots t^d))$ where $f = h/g$ and $t = 1-g$.

Use depth reduction. Depth for $g(1+t+\dots t^d)$ is $O(\log s + \log d)$

Previously, $\text{Hom}_0, \dots \text{Hom}_d$ of $g(1+t+\dots+t^d)$ are computed by:

$\text{Hom}_i(\ell_1 + \ell_2) = \text{Hom}_i(\ell_1) + \text{Hom}_i(\ell_2)$ and $\text{Hom}_i(\ell_1 \cdot \ell_2) = \sum \text{Hom}_j(\ell_1)\text{Hom}_{i-j}(\ell_2)$

Previously, $\mathrm{Hom}_0, \dots, \mathrm{Hom}_d$ of $g(1+t+\cdots+t^d)$ are computed by:

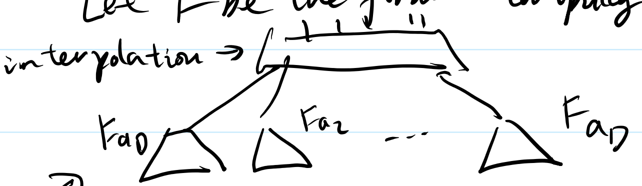
$\mathrm{Hom}_i(l_1+l_2) = \mathrm{Hom}_i(l_1) + \mathrm{Hom}_i(l_2)$ and $\underbrace{\mathrm{Hom}_i(l_1 l_2) = \sum_{0\le j\le i} \mathrm{Hom}_j(l_1)\,\mathrm{Hom}_{i-j}(l_2)}$

Increases the depth by a factor of $O(\log d)$

$\Rightarrow$ size $(sd)^{O(\log d)}$.

Instead, we compute $\mathrm{Hom}_i(g(1+t+\cdots+t^d))$ using interpolation:

Let F be the formula computing $g(1+t+\cdots+t^d)$. $F_a = F(aX_1, \dots, aX_n)$



, $a_0, \dots, a_D \in \mathbb{F}$ are distinct.

where D is an upper bound for $\deg(g(1+t+\cdots+t^d))$

$D \le s + sd = s(d+1)$.

Need $|\mathbb{F}| \ge D+1$.

May not be homogeneous, but this is fine

If $|\mathbb{F}| < D+1$, can work over an extension field $\mathbb{K}$ with degree $[\mathbb{K}:\mathbb{F}] \le \log_{|\mathbb{F}|} D = O(\log_{\mathbb{F}}(sd))$

Simulating $+$ and $\times$ of $\mathbb{K}$ over $\mathbb{F}$. $\times$ becomes $k\times k$ matrices where $k = [\mathbb{K}:\mathbb{F}]$

$\Downarrow$

increases the depth by a factor of $O(\log k)$

$= O(\log\log_{\mathbb{F}}(sd))$.

$\Rightarrow$ size $= (sd)^{O(\log\log_{\mathbb{F}}(sd))}$. $\square$