6.S897 Algebra and Computation April 1, 2015

Lecture 15

Lecturer: Madhu Sudan *Scribe: Tianren Liu*

Today's Topic

- Homogenization
- Division Removal
- Partial Derivatives
- Width Reduction
- Depth Reduction
- Determinant

Homogenization (Low intermediate degree) Consider arithmetic circuits on field $\mathbb{F}$ with operations $\{+, \cdot\}$. If we need an arithmetic circuit computing a low degree polynomial, does it helps to have intermediate computations of high degree?

Theorem 1. *If an arithmetic circuit C compute a polynomial f that,* $\deg(f) \leq d$, $\text{size}(C) \leq s$, *then there exists a circuit C' computing f that all intermediate polynomial of C' is of degree $\leq d$, and* $\text{size}(C') = O(s \cdot d)$, $\text{depth}(C') = O(\log s) \cdot \text{depth}(C)$.

To prove this theorem, we introduce definition of homogeneous i-th degree component of a polynomial. Define $\text{Hom}_i(f) =$ homogeneous i-th degree part of f. Formally, for polynomial $f = \sum_{\mathbf{e}} c_{\mathbf{e}} x^{\mathbf{e}}$, define

$$\text{Hom}_i(f) = \sum_{\mathbf{e}: \sum_j e_j = i} c_{\mathbf{e}} x^{\mathbf{e}}.$$

The Theorem 1 follows from the following lemma as $f = \sum_{i \leq d} \text{Hom}_i(f)$.

Lemma 2 (Homogenization Lemma)**.** *If f has circuit C of size s, then $\{\text{Hom}_i(f)\}_{i \leq d}$ can be computed by a circuit of size $O(s \cdot d)$, depth $O(\text{depth}(C) \cdot \log s)$.*

Proof The new circuit will compute the homogeneous i-th degree component of each intermediate polynomial in C ($i \leq d$). By simple induction, if $f = f_1 + f_2$,

$$\text{Hom}_i(f) = \text{Hom}_i(f_1) + \text{Hom}_i(f_2),$$

if $f = f_1 \cdot f_2$,

$$\text{Hom}_i(f) = \sum_{j \leq i} \text{Hom}_j(f_1) \cdot \text{Hom}_{i-j}(f_2).$$

So for each intermediate polynomial, given the homogeneous components of lower layers, its homogeneous components can be computed in $O(d)$ space and $O(\log s)$ extra depth. ■

Division Removal Division gates is not necessary for an arithmetic circuit. If an arithmetic circuit computing a low degree polynomial with division gates, then we could remove the divsision without blowing up size or depth.

Lemma 3. *If $f \in \mathbb{F}[x_1, \dots, x_n]$ cna be computed by a depth Δ, size s circuit over $\{+, \cdot, \div\}$ and $\deg(f) = d$. Then f can be computed by a circuit of depth $O(\log s) \cdot \Delta$ and size $\mathrm{poly}(s, d)$ over $\{+, \cdot\}$.*

Proof Notice that each intermediate computation is a rational function $f_i = g_i/h_i$. Which $O(1)$ loss in size and depth, we can compute the numerator and denominator of each node seperately.

$$f = f_1 + f_2 \implies \frac{g}{h} = \frac{g_1}{h_1} + \frac{g_2}{h_2} = \frac{g_1 h_2 + g_2 h_1}{h_1 h_2},$$
$$f = f_1 \cdot f_2 \implies \frac{g}{h} = \frac{g_1}{h_1} \frac{g_2}{h_2} = \frac{g_1 g_2}{h_1 h_2}.$$

This would produce an arithmetic circuit computing g, h over $\{+, \cdot\}$. such that $f = \frac{g}{h}$

Assume w.l.o.g. $h(0) = 1$

$$f = \frac{g}{h} = \frac{g}{1 - (1 - h)} = \sum_{i \geq 0} g(1 - h)^i$$

$$\mathrm{Hom}_j(f) = \mathrm{Hom}_j\left(\sum_{i \geq 0} g(1 - h)^i\right) = \mathrm{Hom}_j\left(\sum_{i \geq 0}^{j} g(1 - h)^i\right)$$

By Homogenization Lemma, the homogeneous components of g, h can be computed by a depth $O(\log s) \cdot \Delta$ size $\mathrm{poly}(s, d)$ circuit over $\{+, \cdot\}$. ■

Partial Derivatives [Baur Strassen]

Theorem 4. *Given circuit computing $f(x_1, \dots, x_n) \in \mathbb{F}[x_1, \dots, x_n]$ of size s. There exists circuit of size $O(s)$ computing*

$$\left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n}\right).$$

As a corollary if computing $\phi : \mathbb{F}^n \to \mathbb{F}^m$ need size s, then

$$\hat{\phi} : \mathbb{F}^n \times \mathbb{F}^m \to \mathbb{F} \qquad \hat{\phi}(\mathbf{x}, \mathbf{y}) = \sum y_i \phi(\mathbf{x})$$

needs a $\Omega(s)$ size circuit to compute.

Proof It's prove by induction. Instead of a naive approach computing the partial derivative of each gate wrt to input, we compute the partial derivative of output wrt to each gate.

A circuit can be formalized as a straight line program,

$$\begin{aligned} & x_1 \\ & \quad \vdots \\ & x_n \\ & y_1 \leftarrow x_1 + x_2 \\ & \quad \vdots \\ & y_s \leftarrow y_{s-1} \cdot y_{s-2} \end{aligned}$$

A circuit can also be viewed a series of substitutions.

$$\begin{aligned}
\Psi_s(x_1,\dots,x_n,y_1,\dots,y_s) &= y_s \\
\Psi_s(x_1,\dots,x_n,y_1,\dots,y_{s-1}) &= \Psi_s|_{y_s\leftarrow y_{s-1}\cdot y_{s-2}} \\
&\vdots \\
\Psi_0(x_1,\dots,x_n) &= \Psi_1|_{y_1\leftarrow x_1+x_2}
\end{aligned}$$

Use induction, Ψ_s is the base case, in which the partial derivatives of Ψ_s can be trivially computed in $O(1)$ size

$$\frac{\partial \Psi_s}{\partial x_j} = 0 \qquad \frac{\partial \Psi_s}{\partial y_j} = \delta_{js}$$

Assume for some index i, we have a circuit computing

$$\frac{\partial \Psi_i}{\partial x_j}, \frac{\partial \Psi_i}{\partial y_j}.$$

Let $y_i \leftarrow y_l + y_k$ or $y_i \leftarrow y_l \cdot y_k$, we want to compute

$$\frac{\partial \Psi_{i-1}}{\partial x_j}, \frac{\partial \Psi_{i-1}}{\partial y_j}$$

As $\Psi_{i-1}(\dots,y_{i-1}) = \Psi_i(\dots,y_{i-1},y_i(y_l,y_k))$, the partial derivatives of $\{x_j\},\{y_j\}$ besides y_l, y_k are the same

$$\begin{aligned}
\frac{\partial \Psi_{i-1}}{\partial x_j}(x_1,\dots,x_n,y_1,\dots,y_{i-1}) &= \frac{\partial \Psi_i}{\partial x_j}(x_1,\dots,x_n,y_1,\dots,y_{i-1},y_i(y_l,y_k)) \\
\frac{\partial \Psi_{i-1}}{\partial y_j}(x_1,\dots,x_n,y_1,\dots,y_{i-1}) &= \frac{\partial \Psi_i}{\partial y_j}(x_1,\dots,x_n,y_1,\dots,y_{i-1},y_i(y_l,y_k)) \qquad j \notin \{l,k\}
\end{aligned}$$

for y_l, y_k

$$\begin{aligned}
&\frac{\partial \Psi_{i-1}}{\partial y_l}(x_1,\dots,x_n,y_1,\dots,y_{i-1}) \\
&= \frac{\partial \Psi_i}{\partial y_l}(x_1,\dots,x_n,y_1,\dots,y_{i-1},y_i(y_l,y_k)) + \frac{\partial \Psi_i}{\partial y_i}(x_1,\dots,x_n,y_1,\dots,y_{i-1},y_i(y_l,y_k))\frac{\partial y_i(y_l,y_k)}{\partial y_l}.
\end{aligned}$$

If $y_i \leftarrow y_l + y_k$, $\frac{\partial y_i(y_l,y_k)}{\partial y_l} = 1$; if $y_i \leftarrow y_l \cdot y_k$, $\frac{\partial y_i(y_l,y_k)}{\partial y_l} = y_k$.

The circuit computing the derivatives of Ψ_{i-1} has $O(1)$ more gates than the circuit computing derivatives of Ψ_i. Therefore, there exists circuit of size $O(s)$ computing the derivatives of $f = \Psi_0$. ■

Width Reduction When the memory is limited, consider a register machine model of computation. Memory is a set of registers $M = \{R_1,\dots,R_m\}$. Unlikely previous model where all intermediate result is stored and can be later used, the machine could only remember m intermediate results. The arithmetic computation can be considered as a straight line program

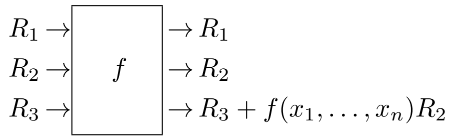
$$\begin{aligned}
&R_1 \leftarrow X_1 + \gamma X_2 \\
&R_2 \leftarrow \alpha R_1 + \beta X_5 \\
&R_1 \leftarrow \dots \\
&\quad\vdots
\end{aligned}$$

Theorem 5 (Barrington)**.** *If boolean ϕ has formula size s implies ϕ can be computed with $\log_2 s$ bits of memory in size s^2.*

Theorem 6 (Ben-Or-Clere)**.** *If polynomial f has formula size s, then f can be computed by 3-register machine in size s^2.*

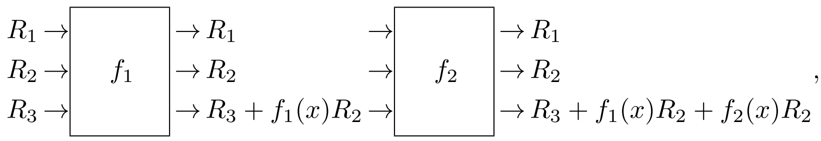
Proof If $f \leftarrow f_1 \cdot f_2$, then $\text{size}(f) = \text{size}(f_1) + \text{size}(f_2)$. First we should applies ... to balance the formula, so that the formula, viewed as a binary tree, is balanced. This would introduce an $O(1)$-factor on the size (NEED VERIFY).

In Ben-Or-Clere, we are looking for a computation sequence that

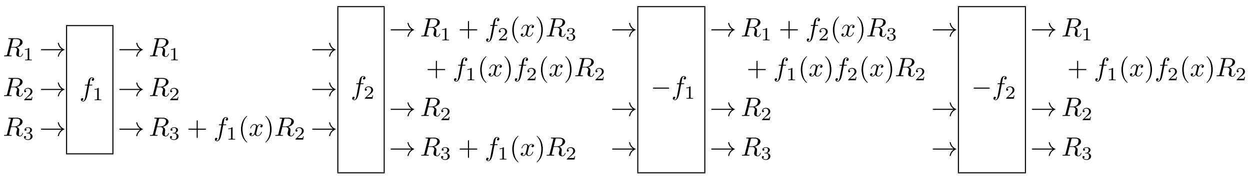


The sequence takes the initial values stored in the registers as a part of the inputs. If the registers is initialized as $R_2 = 1, R_3 = 0$, then such sequence will compute $f(x)$.

Assuming we've found such computation sequence for f_1 and f_2, to compute $f = f_1 + f_2$,



to compute $f = f_1 f_2$,



In either case, $\text{size}_{\text{3-Reg}}(f) \leq 2\,\text{size}_{\text{3-Reg}}(f_1) + 2\,\text{size}_{\text{3-Reg}}(f_2)$. ■

Depth Reduction (If we have a depth reduction method,) consider boolean circuit and operations $\{+, \cdot\}$ (which is complete). Then we would have a general method to reduce depth of boolean circuit. (Which is unlikely.)

Theorem 7. *f computed by size s circuit, $\deg(f) = d \implies f$ can be computed in size $\text{poly}(s, d)$ depth $(\log s)(\log d)$*

Remark: Then $s = \text{poly}(n)$, size s boolean circuit is $P/_{\text{poly}}$ class, size $\text{poly}(s, d)$ depth $(\log s)(\log d)$ boolean circuit is like $\mathcal{C}NC_2$ class. The reason why we didn't prove $P/_{\text{poly}} \subseteq \mathcal{C}NC_2$ is when we transfer a boolean circuit to a boolean formula, the degree of output may blow up.

Proof Let $f_v(x_1, \ldots, x_n)$ be function computed by gate v, $\partial_{v,w}(x_1, \ldots, x_n)$ be partial derivative of gate v wrt gate w.

Set w as a variable, gives $\tilde{f}_v(x_1, \ldots, x_n, w)$, then

$$\partial_{v,w}(x_1, \ldots, x_n) = \frac{\partial \tilde{f}_v}{\partial w}(x_1, \ldots, x_n, f_w)$$

In i-th stage, compute

- all f_w that $\deg(f_w) \in \{2^i, \ldots, 2^{i+1} - 1\}$
- all $\partial_{v,w}$ that $\deg(\partial_{v,w}) \in \{2^i, \ldots, 2^{i+1} - 1\}$

(from previous stage) in $\log s$ depth. ■