Circuit Complexity and Computational Complexity

Lecture Notes for Math 267AB
University of California, San Diego
January–June 1992

Instructor: Sam Buss

Notes by: Frank Baeuerle
Sam Buss
Bob Carragher
Matt Clegg
Goran Gogić
Elias Koutsoupias
John Lillis
Dave Robinson
Jinghui Yang

These lecture notes were written for a topics course in the Mathematics Department at the University of California, San Diego during the winter and spring quarters of 1992. Each student wrote lecture notes for between two and five one and one half hour lectures.

I'd like to thank the students for a superb job of writing up lecture notes.

January 9-14. Bob Carragher's notes on
Introduction to circuit complexity.
Theorems of Shannon and Lupanov giving upper and lower bounds of circuit complexity of almost all Boolean functions.

January 14-21. Matt Clegg's notes on
Spira's theorem relating depth and formula size
Krapchenko's lower bound on formula size over AND/OR/NOT

January 23-28. Frank Baeuerle's notes on
Neciporuk's theorem
Simulation of time bounded TM's by circuits

January 30 - February 4. John Lillis's notes on
NC/AC hierarchy
Circuit value problem
Depth restricted circuits for space bounded Turing machines
Addition is in NC^1.

February 6-13. Goran Gogić's notes on
Circuits for parallel prefix vector addition and for symmetric functions.

February 18-20. Dave Robinson's notes on
Schnorr's 2n-3 lower bound on circuit size
Blum's 3n-o(n) lower bound on circuit size

February 25-27. Jinghui Yang's notes on
Subbotovskaya's theorem
Andreev's $n^{2.5}$ lower bound on formula size with AND/OR/NOT.

March 3-5. Elias Koutsoupias's notes on
Hastad's switching lemma.

March 10-12. Sam Buss's notes on
Applications of Hastad's Lemma.

April 7-14. Frank Baeuerle's notes on
Constant depth reducibilities (Chandra-Stockmeyer-Vishkin)
Smolensky's theorem (beginning part)

April 16-23. Goran Gogić's notes on
Smolensky's Theorem (concluding part)
Probabalistic Circuits
Valiant-Vazirani Theorem (beginning part)

April 28 - May 7. Jinghui Yang's notes on
Vazirani-Valiant (concluding part)
Theorems of Toda and Allender-Hertrampf

May 12-19. Dave Robinson's notes on
Uniform circuit families, unbounded and bounded fanin

May 26 - June 2. Elias Koutsoupias's notes on
Conclusion of uniformity for AC^0
Linear-time Hierarchy
Multiplication is in constant-alternation linear time
Log depth circuits for division (Beame-Cook-Hoover)

June 4. Sam Buss's notes on
Log depth circuits for division (concluding part)

January - June. Unified list of homework assignments.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 7 January 1991 **Topic: Organizational Meeting**
Instructor: Sam Buss **Notes by: Robert J. Carragher**

Instructor Information	
Sam Buss	
Office:	AP&M 6210
Phone:	x4-6455
Email:	sbuss@ucsd.edu
	sbuss@cs
	sbuss@math
Emergencies	792-9674 (home)

There is a tentative room change to AP&M 7218. The change will be confirmed later this week via email.

The coverage of this course includes the following topics:

Circuit Size, Circuit Depth, Formula Size } Circuit complexity using AND, OR, and NOT gates

Relevant papers and results to be discussed include those by

Hastad
Razborov
Smolensky
Yao
Toda

Note that the course is flexible, depending upon the interests of the students. For example, a preliminary survey revealed that necessary is a review of many theorems basic for a good understanding of the above results.

Next Thursday's lecture will cover the Shannon and Lupanov theorem which states that most boolean functions over n inputs require circuits of size $\frac{2^n}{n}$.

In the meantime, there are several helpful references as well, none of which need be purchased. (Most are unavailable for purchasing in any case.) If a reference cannot be found at UCSD, then an interlibrary loan may work, although it usually takes two weeks to process.

Complexity of Computing,		QA267.S28, S&E	UCSD
John Savage		QA267.S28, Library	UCSB
		QA267.S28, Rivera	UCR
		QA267.S28, Phy Sci	UCR
		QA267.S28, Phys Sci	UCD
		QA267.S281, Moffitt	UCB
	2 copies	QA267.S281, Astr/Math	UCB
Complexity of Boolean Networks,		QA76.9.M35.D86, S&E	UCSD
Paul Dunne		QA76.B7 no.29, Science	UCSC
		QA76.9.M35.D86, Library	UCSB
		QA76.9.M35.D86, Rivera	UCR
		QA267.7.D86, Engr/Math	UCLA
		QA76.9.M35.D85, Phys Sci	UCD
		QA76.9.M35.D861, Astr/Math	UCB
Complexity of Boolean Functions,		**Note: None at UCSD**	
Ingo Wegener		QA10.3.W431, Astr/Math	UCB
		QA10.3.W44, Phys Sci	UCD
		QA10.3.W44, Main Lib	UCI
		QA10.3.W44, Rivera	UCR
		QA10.3.W44, Library	UCSB
Handbook of Theoretical Computer Science*		CSE Theory Lab	
		QA76.H279, Astr/Math	UCB
		QA76.H279, Engin	UCB
		QA76.H279, Phys Sci	UCD
		QA76.H279, Main Lib	UCI
		QA76.H279, Library	UCSB
		QA76.H279, Science	UCSC

Forthcoming, unnamed book by *Clote* and *Kranakis*

*Only the Boppana/Sipser section will be covered, although a perusal of the Karp/Ramachandran section may prove useful.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 9 January 1991 **Topic: Circuit Definitions and Shannon's Theorem**
Instructor: Sam Buss **Notes by: Robert J. Carragher**

Definition 1 *A Boolean function f is a mapping, $f : \{0,1\}^n \to \{0,1\}$. By convention, 1 means* TRUE *and 0 means* FALSE.

Example 1 $\wedge$ and $\vee$ are binary Boolean functions. (Note that $\cdot$ and $+$ are used as alternatives, respectively, for $\wedge$ and $\vee$.)

$\neg$ is a unary Boolean function. (An alternative for $\neg x$ is $\overline{x}$.)

0 and 1 are zero-ary (constant) Boolean functions.

□

Definition 2 *Let Ω be a set of Boolean functions (e.g. $\Omega = \{\wedge, \vee, \neg\}$). An* **$\Omega$-circuit** *is a directed, acyclic, labelled graph such that for every node v, EITHER v has indegree 0 and is labelled by a variable $x_1, \ldots, x_n$ OR v is labelled with a function in Ω of arity m and has m incoming, ordered edges*[1]*. Nodes of the former type are called* **inputs** *and nodes of the latter type are called* **gates**.

Note that each node in the circuit computes a Boolean function.

The circuit computes Boolean functions $f_1, \ldots, f_k$ if there are k nodes in the circuit computing these functions.

Sometimes circuits are given as straight-line programs or as alternating AND-OR circuits with unbounded fan in.

Definition 3 *Ω is* **complete** *if and only if for any Boolean function f there is an Ω-circuit that computes f.*

[1] The ordering of the edges is optional for symmetric functions.

Definition 4 *A circuit is a* formula *if and only if every node in the circuit has outdegree at most 1 and the circuit is connected.*

Definition 5 *The* depth *of a circuit is the length of the longest, directed path in the circuit.*

The following function definitions concern complexity measures for classes of circuits implementing functions over a given set of Boolean functions. Let $g(C)$ be the number of gates in circuit C.

$$C_\Omega(f) = \min\{g(C) : C \text{ computes } f \text{ and is an } \Omega\text{-circuit}\}$$
Circuit Complexity of f

$$L_\Omega(f) = \min\{g(C) : C \text{ computes } f \text{ and is an } \Omega\text{-formula}\}$$
Formula Complexity of f

$$D_\Omega(f) = \min\{\text{depth of } C : C \text{ computes } f \text{ and is an } \Omega\text{-circuit}\}$$
Depth Complexity of f

As a special case of $\Omega = \{\wedge, \vee, \neg\}$,

$$C^*_{\{\wedge,\vee,\neg\}}(f) = \min\{\#\text{ of } \wedge \text{ and } \vee \text{ gates in } C : C \text{ computes } f \text{ and is an } \Omega\text{-circuit}\}$$
Circuit Complexity of f

Example 2 Let $f = x_1 \oplus x_2$, the 2-variable parity function. Implemented as

$$(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$$

2-variable parity requires 5 gates. Thus,

$$C_{\{\wedge,\vee,\neg\}}(\oplus) \leq 5.$$

□

HW 1: Show $C_{\{\wedge,\vee,\neg\}}(\oplus) \leq 4$.

HW 2: Prove $C_{\{\wedge,\vee,\neg\}}(\oplus) = 4$.

Observation 1 *There are exactly 2^{2^n} Boolean functions on n inputs.*

Proof. There are 2^n possible input values. Each value may be assigned the value 0 or 1 by the function. This gives us 2^{2^n} possible functions.

Q.E.D.

Theorem 1 *Let $n \geq 1$. Let Ω be any complete set of connectives. There exists a circuit of size $2^{2^n} - n$ that computes all 2^{2^n} Boolean functions of n variables.*

Proof. Note that we get all functions of the form $f(x_1, \ldots, x_n) = x_i, 1 \leq i \leq n$ for free since these are given as inputs for the circuit. To build the rest of the circuit, we iteratively add new functions to the circuit. Each new node implements a different function (no function need be repeated). All functions will be computed since Ω is complete.

Q.E.D.

Claim 2 *The above theorem is not true for formulas.*

The proof is part of **HW 3***. (The * means Sam didn't know the answer.) The remainder of HW 3* is to give a lower bound on the size of a formula computing all 2^{2^n} Boolean functions of n variables.

Now we turn to the first interesting theorem of the class. In 1942, Riordan and Shannon provided a lower bound on the average formula complexity. Then, in 1949, Shannon provided a counting proof to the following theorem on average circuit complexity:

Theorem 3 (Shannon, 1949) *For n sufficiently large, almost all Boolean functions on $x_1, \ldots, x_n$ require circuit size at least $2^n/n(r-1)$, where the circuit is over any finite, complete Ω consisting of functions of arity at most r.*

Proof. We will prove Shannon's Theorem for $\Omega = \{\wedge, \vee, \neg\}$. The proof for arbitrary Ω with $r \geq 2$ is analogous.

Let $F(s, n)$ be the number of Boolean functions (n-ary) computable by circuits of size exactly s with no two nodes computing the same function. A circuit of size s

- has s gates with up to $(s + n - 1)^r$ possible inputs ($r = 2$ in this case),
- has gates that can be $\wedge, \vee$, or $\neg$ (3 possibilities),
- computes $s + n$ functions, and
- has $s!$ permutations (renaming of the gates).

Thus,

$$\begin{aligned}
F(s,n) &\leq \frac{3^s((s+n-1)^2)^s(s+n)}{s!} \\
&\leq \frac{3^s(2s)^{2s}(s+n)}{s!} \text{ since } s \geq n \\
&\leq \frac{3^s 4^s s^{2s}(s+n)}{s^s/e^s} \text{ using } s! \geq (s/e)^s \text{ for sufficiently large } n \text{ (see below)}^2 \\
&\leq (24e)^s s^s \\
&= c^s s^s \quad \text{for some constant } c
\end{aligned}$$

(A quick proof that $s! \geq (s/e)^s$.)

$$\begin{aligned}
\ln s! &= \sum_{i=1}^{s} \ln i \\
&\geq \int_1^s \ln x \, dx \\
&= x \ln x - x|_1^s \\
&= s \ln s - s - (0 - 1) \\
&= s \ln s - (s - 1)
\end{aligned}$$

[2] Stirling's formula could also be used here.

So,

$$\begin{aligned} \Rightarrow s! &\geq \exp\left(s \ln s - (s-1)\right) \\ &= e^{s \ln s}/e^{s-1} \\ &\geq s^s/e^s \end{aligned}$$

What if $s \leq 2^n/n$? (We will use log to mean $\log_2$.)

$$\begin{aligned} \log F(s,n) &\leq \log(c^s s^s) \\ &= s(\log c + \log s) \\ &\leq (2^n/n)(\log c + \log(2^n/n)) \\ &= 2^n \frac{1}{n}(\log c + n - logn) \\ &= 2^n \left(1 - \frac{\log n - \log c}{n}\right) \\ &< 2^n \left(1 - \frac{1}{n}\right) \text{ for sufficiently large } n \end{aligned}$$

Thus,

$$\begin{aligned} F(s,n) &\leq 2^{2^n(1-\frac{1}{n})} \\ &= \frac{1}{2^{2^n/n}} 2^{2^n} \end{aligned}$$

In other words, the percentage of Boolean functions requiring no more than $2^n/n$ gates is very tiny as n grows large.

Q.E.D.

Remark 1: there are no known, non-linear lower bounds for "natural" Boolean functions (e.g. Travelling Salesman Problem).

Remark 2: an obvious upper bound for the circuit complexity, $C_\Omega(f)$, over $\Omega = \{\wedge, \vee, \neg\}$ is given by the disjunctive normal form of the function, f. Thus for any Boolean function f over n variables,

$$\begin{aligned} C_\Omega(f) &\leq (n-1)2^n + (2^n - 1) + n \\ &= n2^n(1 + o(1)) \end{aligned}$$

where the first quantity in the first line represents an upper bound on the number of conjuncts in the formula, the second quantity represents an upper bound on the number of disjuncts, and the third quantity represents the negating of the input variables.

Definition 6 *Let $x^1 \equiv x$ and $x^{-1} \equiv \neg x$. A full minterm of $x_1, \ldots, x_n$ is a conjunct of the form $x_1^{\pm 1}, \ldots, x_n^{\pm 1}$.*

Theorem 4 *There exists a circuit of size $2^n(1 + o(1))$ which computes all full minterms of $x_1, \ldots, x_n$.*

Proof. Let $f_C(n)$ be the minimum size of a circuit, C, that computes all full minterms of $x_1, \ldots, x_n$. We will implement the above C_Ω using a divide-and-conquer strategy which will give us $f_C(n) = 2^n(1 + o(1))$. The circuit is formed as follows

- construct a circuit for all full minterms of the variables $x_1, \ldots, x_{\lceil n/2 \rceil}$;
- construct a circuit for all full minterms of the variables $x_{\lceil n/2 \rceil + 1}, \ldots, x_n$;
- construct 2^n many conjunctions of the minterms from the above two steps

This gives us the recursive formula

$$f(n) \leq 2^n + f(\lceil n/2 \rceil) + f(\lfloor n/2 \rfloor), \quad n \geq 4.$$

For $n \leq 4$, we have $f(1) = 1, f(2) \leq 6, f(3) \leq 15$, and $f(4) \leq 28$.

To complete the proof, we need to show that $f(n) = 2^n(1 + o(1))$. This is given by induction on n (sketched below). Our hypothesis is $f(n) \leq 2^n(1 + \epsilon_n)$, where $\epsilon_n < 1$ and $\epsilon_n \to 0$ as $n \to \infty$. By inspection, $\epsilon_n < 1$ for $n = 1, 2, 3, 4$. For $n > 4$,

$$\begin{aligned} f(n) &\leq 2^n + f(\lceil n/2 \rceil) + f(\lfloor n/2 \rfloor) \\ &\leq 2^n + 2f(\lceil n/2 \rceil) \\ &\leq 2^n + 2 \cdot 2^{\lceil n/2 \rceil}(1 + \epsilon_{\lceil n/2 \rceil}) \\ &= 2^n \left(1 + \frac{1 + \epsilon_{\lceil n/2 \rceil}}{2^{n - \lceil n/2 \rceil - 1}}\right) \\ &= 2^n(1 + \epsilon_n) \end{aligned}$$

So $\epsilon_n \to 0$ as $n \to \infty$, where

$$\epsilon_n = \frac{1 + \epsilon_{\lceil n/2 \rceil}}{2^{n - \lceil n/2 \rceil - 1}} < \frac{1}{2^{n - \lceil n/2 \rceil - 2}} \leq 1$$

since $n > 4$.

Q.E.D.

HW 4*: Let $g(1) = 1$ and $g(n) = 2^n + g(\lceil n/2 \rceil) + g(\lfloor n/2 \rfloor)$. Is $g(n)$ the size of the smallest circuit computing all the full minterms of $x_1, \ldots, x_n$?

Remark 3: Theorem 4 can be used to improve Remark 2 to show that $C_\Omega(f)(x_1, \ldots, x_n) \leq 2^n(1 + o(1))$, where $\Omega = \{\wedge, \vee, \neg\}$.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 14 January 1991 **Topic: Upper Bounds on Circuit Size**
Instructor: Sam Buss **Notes by: Robert J. Carragher**

Note: The current "solution" to **HW 3***, due to Matthew Clegg, is that the size of the formula is at least

$$2^{2^n}\left(1+\frac{1}{2^n}\right)-\mathcal{O}\left(n^2\right).$$

Theorem 5 *(Not proved in class.) Any* $f:\{0,1\}^n\to\{0,1\}$ *has*

$$L_{\{\wedge,\vee,\neg\}}(f)\leq 2\cdot\frac{2^n}{\log n}(1+o(1)).$$

This gives an upper bound by multiplying by 2^{2^n} to **HW 3*** that is the best that Sam knows.

References for today's lecture:

- Shannon, "The Synthesis of Two-Terminal Switching Circuits." *Bell System Technical Journal*, v. 28 (1949):59-98.
- Lupanov, "On a Method of Circuit Synthesis (translated title)." *Izvestia VUZ (Radiofizika)*, v. 1 (1958):120-140. (Not known whether an English translation exists.)

Recall Shannon's theorem from the previous class. Also, recall the theorems about a circuit computing all 2^n full minterms and a circuit computing all 2^{2^n} functions. We will now prove a converse for the upper bound on circuits.

Theorem 6 *a.* **(Shannon)** *Let* $f:\{0,1\}^n\to\{0,1\}$. *Then*

$$C_{\{\wedge,\vee,\neg\}}(f)\leq 2\cdot\frac{2^n}{n}+o\left(\frac{2^n}{n}\right).$$

b. *Let* $f_1, \ldots, f_m : \{0,1\}^n \to \{0,1\}$. *Then there is a circuit computing all the* f_i*'s such that the circuit size is at most*

$$(m+1)\frac{2^n}{n} + o\left(\frac{2^n}{n}\right).$$

HW 5: Improve part (b) using m instead of $m+1$. Hint, use the proof method of Lupanov's theorem below (theorem 7).

HW 6: Show a tight lower bound on a circuit computing $f_1, \ldots, f_m$ for almost all $f_1, \ldots, f_m$.

Proof.

a. Let $k = \lceil \log(n - 2\log n) \rceil$. The input variables will be divided into $x_1, \ldots, x_{n-k}$ and $x_{n-k+1}, \ldots, x_n$. We will build the circuit for f in three stages, using the large group to select functions over the small group.

1. First, build the circuit for all 2^{2^k} functions on $x_{n-k+1}, \ldots, x_n$ of size $2^{2^k} - k$. We have that
$$2^{2^k} \leq 2 \cdot 2^{n-2\log n} = 2 \cdot \frac{2^n}{n^2} = o\left(\frac{2^n}{n}\right).$$

2. Build a circuit of size $2^{n-k}(1+o(1))$ computing all full minterms on $x_1, \ldots, x_{n-k}$. We have that
$$\begin{aligned} 2^{n-k}(1+o(1)) &\leq \frac{2^n}{n-2\log n}(1+o(1)) \\ &= \frac{2^n}{n\left(1-\frac{2\log n}{n}\right)}(1+o(1)) \\ &= \frac{2^n}{n}\frac{(1+o(1))}{(1-o(1))} \\ &= \frac{2^n}{n}(1+o(1)) \end{aligned}$$

3. If σ is a full minterm over $x_1, \ldots, x_{n-k}$ then there is a function, f_σ, on $x_{n-k+1}, \ldots, x_n$ such that for all truth values for $x_1, \ldots, x_n$ satisfying σ, we have
$$f(x_1, \ldots, x_n) = f_\sigma(x_{n-k+1}, \ldots, x_n).$$
With this in mind, we have
$$f(x_1, \ldots, x_n) \equiv \bigvee_\sigma \sigma(x_1, \ldots, x_{n-k}) \wedge f_\sigma(x_{n-k+1}, \ldots, x_n),$$

where σ is any full minterm on $x_1, \ldots, x_{n-k}$ and $\bigvee$ means enough binary $\vee$'s, not one unbounded $\vee$. The problem with this approach is that the circuit built from this specification is twice as large as desired. With a simple distribution of the f_σ's, there being far more σ's than f_σ's, we have

$$\bigvee_F \left(F(x_{n-k+1}, \ldots, x_n) \wedge \bigvee_{\sigma \text{ s.t. } f_\sigma = F} \sigma(x_1, \ldots, x_{n-k}) \right),$$

where F ranges over all functions on $x_{n-k+1}, \ldots, x_n$. There are $2^{2^k} = o(2^n/n)$ F's and thus $o(2^n/n)$ $\vee$'s in the outer disjunct. For a fixed F, if there are i σ's such that $f_\sigma = F$ then there are $i-1$ inner $\vee$'s and one $\wedge$. Since each σ is used exactly once, there are 2^{n-k} inner connectives, so stage 3 uses $\frac{2^n}{n}(1+o(1))$ gates

Putting all three stages together yields a circuit requiring

$$2 \cdot \frac{2^n}{n}(1+o(1))$$

gates.

b. We repeat the first two stages of the previous proof, and repeat the third stage m times, giving us a circuit of the size stated in the theorem. Note that the $o(1)$ quantity depends only upon n.

Q.E.D.

Theorem 7 (Lupanov) *Let $f : \{0,1\}^n \to \{0,1\}$. Then*

$$C_{\{\wedge,\vee,\neg\}}(f) \leq \frac{2^n}{n} + o\left(\frac{2^n}{n}\right).$$

Proof. Let $n' = n - \lceil \log n \rceil$. As in the previous proof, we split the input variables into two groups, $x_1, \ldots, x_{n'}$ and $x_{n'+1}, \ldots, x_n$. We build a circuit in stages, but this time using the small group to select functions over the big group.

a. Form a circuit computing all full minterms on $x_{n'+1}, \ldots, x_n$. This has size at most

$$2^{\lceil \log n \rceil}(1+o(1)) \leq 2 \cdot n(1+o(1)) = o\left(\frac{2^n}{n}\right).$$

b. For each full minterm σ on $x_{n'+1}, \ldots, x_n$, let f_σ be the induced Boolean function on $x_1, \ldots, x_{n'}$. There are $2^{\lceil \log n \rceil}$ f_σ's. Now we use part (b) of the previous theorem to compute the circuit computing all the f_σ's. Here, $m = 2^{\lceil \log n \rceil}$. This size is at most

$$\left(2^{\lceil \log n \rceil} + 1\right) \frac{2^{n'}}{n'}(1 + o(1)).$$

Thus, we have a circuit with size (number of gates) at most

$$\begin{aligned}
\frac{2^{n-\lceil \log n \rceil}}{n - \lceil \log n \rceil} \left(2^{\lceil \log n \rceil} + 1\right) + o\left(\frac{2^{n-\lceil \log n \rceil}}{n - \lceil \log n \rceil}\right) &= \frac{2^n \left(2^{\lceil \log n \rceil} + 1\right)}{2^{\lceil \log n \rceil}(n - \lceil \log n \rceil)} + o\left(\frac{2^n}{n}\right) \\
&= \frac{2^n}{n} \cdot \frac{\left(1 + \frac{1}{2^{\lceil \log n \rceil}}\right)}{\left(1 - \frac{\lceil \log n \rceil}{n}\right)} + o\left(\frac{2^n}{n}\right) \\
&= \frac{2^n}{n}(1 + o(1)) + o\left(\frac{2^n}{n}\right) \\
&= \frac{2^n}{n}(1 + o(1))
\end{aligned}$$

c. Finally, we form the circuit

$$\bigvee_\sigma \sigma(x_{n'+1}, \ldots, x_n) \wedge f_\sigma(x_1, \ldots, x_{n'}),$$

where σ ranges over all full minterms over $x_{n'+1}, \ldots, x_n$. There are $2^{\lceil \log n \rceil}$ many σ's, so the number of gates added in this step is at most

$$2 \cdot 2^{\lceil \log n \rceil} = o\left(\frac{2^n}{n}\right).$$

Putting the three steps together nets a circuit of the size given by the theorem.

Q.E.D.

Remark 4: (For "experts" only.) If $k = \lceil \log n \rceil$ and $s = \lceil 2 \log(n - k) \rceil$ in the above proof, then the above construction is a $k - s$ Lupanov decomposition.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: January 14 & 16, 1992 **Topic: Depth and Formula Size**
Instructor: Sam Buss **Notes by: Matthew Clegg**

We begin this section with the following simple observation about trees.

Theorem 8 *Let Ω be a set of Boolean functions (gate types) of arity at most two. If an Ω-formula has depth d, then it has size at most $2^d - 1$.*

Question 1 *What if the maximum arity is r?*

Assuming that every gate in Ω has arity at most 2, this shows that $L_\Omega(f) \leq 2^{D_\Omega(f)} - 1$. In other words,

$$D_\Omega(f) \geq \log(L_\Omega(f) + 1)$$

Our aim is to show that this inequality is asymptotically sharp, i.e, that

$$D_\Omega(f) = \mathcal{O}\left(\log(L_\Omega(f))\right)$$

Definition 7 *If $\Omega = \{0, 1, \neg, \vee, \wedge\}$, then $D^*_\Omega(f)$ is defined to be the minimum over all circuits C computing f of the maximum number of $\vee$ and $\wedge$ gates occurring in a directed path in C. In other words,*

$$D^*_\Omega(f) = \min_{C \text{ computing } f} \max\{\text{number of } \vee, \wedge \text{ gates in a directed path in } C\}$$

Note. By abuse of notation, we will write $L^*_\Omega(\varphi)$ and $D^*_\Omega(\varphi)$ to refer to the size (respectively, depth) of the particular formula φ rather than to the minimum over all formulas of the function computed by φ.

Proposition 9 *If $f : \{0,1\}^n \to \{0,1\}$ is a boolean function, $\Omega = \{\neg, \vee, \wedge\}$, and $\Omega' = \{0, 1, \neg, \vee, \wedge\}$, then*

$$D_\Omega(f) \leq \max\{2, 1 + D^*_{\Omega'}(f)\}.$$

Proof. Our approach in this proof is to show that for any $\{0, 1, \neg, \vee, \wedge\}$–formula computing f, we can find an equivalent $\{\neg, \vee, \wedge\}$–formula of the same (or smaller) depth where there is only one $\neg$ gate along any path from the root to a leaf. If we are thus given a formula φ which is optimal with respect to the measure $D^*_{\Omega'}(f)$, we can convert this formula to one which is only slightly larger with respect to the measure $D_\Omega(f)$.

Our construction proceeds in two phases. First, we push the negations down to the inputs using De Morgan's law, by replacing each subformula in φ of the form $\neg(\psi \vee \chi)$ (respectively, $\neg(\psi \wedge \chi)$) with the equivalent formula $\neg\psi \wedge \neg\chi$ (respectively, $\neg\psi \vee \neg\chi$). After repeating this procedure a number of times, we will arrive at a formula where every $\neg$ gate occurs as part of a formula of the form $\neg x_i$ or $\neg 0$ or $\neg 1$.

In the second phase, we replace subformulas containing constants by subformulas without constants. Thus, $0 \wedge \psi$ is replaced by 0, $0 \vee \psi$ is replaced by ψ and so on. After repeating this step a number of times, we will arrive at a formula which is either itself a constant or which contains no constants. In the former case, we have $D_\Omega(f) = 2$ but $D^*_{\Omega'}(f) = 0$. In the latter case, we have $D_\Omega(f) \leq D^*_{\Omega'}(f) + 1$, since any path from the root to a leaf contains at most one $\neg$ gate.

Q.E.D.

Definition 8 *If T is a tree, the* leafsize *of T is the number of leaves in T. For φ a formula, the* leafsize *of φ is the leafsize of the corresponding tree.*

Notes.

- If every node in T has at most 2 children and b of the nodes in T have exactly 2 children, then
$$\text{leafsize}\,(T) = 1 + b.$$
- If every gate in Ω has arity at most 2, then
$$L^*_\Omega(\varphi) = \text{leafsize}\,(\varphi) - 1.$$

Lemma 10 ($\frac{1}{3}$–$\frac{2}{3}$ lemma) *If T is a tree with all nodes having arity at most 2, and if $m \geq 2$ is the leafsize of T, then there is a subtree S of T with leafsize s, where*
$$\lceil \tfrac{1}{3} m \rceil \leq s \leq \lfloor \tfrac{2}{3} m \rfloor$$

Proof. Let S be a minimal subtree of T of leafsize $s \geq \lceil \frac{1}{3}m \rceil$. If $s = 1$, then $m = 2$ or $m = 3$ and we are done. Otherwise, consider the root node of S. If this node would have only one child, then this child would be the root of a smaller subtree having the same leafsize as S. Thus, the root node of S has at least two children. Let the leaf sizes of the two immediate subtrees of S be s_1 and s_2. By assumption, both s_1 and s_2 are strictly smaller than $\lceil \frac{1}{3}m \rceil$. Thus,

$$s = s_1 + s_2 \leq 2(\lceil \tfrac{1}{3}m \rceil - 1) \leq 2((\tfrac{1}{3}m + \tfrac{2}{3}) - 1) < \tfrac{2}{3}m.$$

Q.E.D.

Theorem 11 (Spira 1971)

(a) Let C be a $\{0, 1, \neg, \vee, \wedge\}$–formula of leafsize m. Then, there is an equivalent $\{0, 1, \neg, \vee, \wedge\}$–formula C' such that

$$D^*_\Omega(C') \leq 2 \log_{3/2} m = \frac{2}{\log 3 - 1} \log m \approx 3.419 \log m$$

and such that

$$\text{leafsize}\,(C') \leq m^\alpha,$$

where α satisfies $\frac{1+2^\alpha}{3^\alpha} \leq \frac{1}{2}$. (For example, taking $\alpha \geq 2.1964$ will do.)

(b) Let B_2 be the set of all binary gate types. If C is a B_2–formula of leafsize m, then there exists an equivalent $\{0, 1, \neg, \vee, \wedge\}$–formula C' such that

$$D^*_\Omega(C') \leq \frac{2}{\log 3 - 1} \log m \approx 3.419 \log m$$

and such that

$$\text{leafsize}\,(C') \leq m^\alpha,$$

where α is as in (a).

Proof.

(a) We proceed by induction on m. If $m = 1$, then C must compute one of the formulas x_i, $\neg x_i$, 0, or 1. In each case, we can find a corresponding formula C' such that $D^*_\Omega(C') = 0$ and $\text{leafsize}\,(C') = 1$.

For a larger value of m, the $\frac{1}{3}$–$\frac{2}{3}$ lemma provides us with a subformula D of C having leafsize s, where $\lceil \frac{1}{3}m \rceil \leq s \leq \lfloor \frac{2}{3}m \rfloor$. Consider the formulas C_0 and C_1 obtained from C by replacing the subformula D by the constant functions 0 and 1 respectively. For a given input to C, the subformula D will evaluate either to 0 or to 1. If D evaluates to 0, then C will have the same value as the formula $C_0 \wedge \neg D$. Similarly, if D evaluates to 1, C will have the same value as the formula $C_1 \wedge D$. Consequently, we see that

$$C \equiv (C_0 \wedge \neg D) \vee (C_1 \wedge D).$$

The constants introduced in C_0 and C_1 can be eliminated by collapsing the gates which use them. Thus, C_0 and C_1 are equivalent to formulas of leafsize at most $\lfloor \frac{2}{3}m \rfloor$. Since D was chosen according to the $\frac{1}{3}$–$\frac{2}{3}$ lemma, we know that D has leafsize at least $\lceil \frac{1}{3}m \rceil$. Now use the induction hypothesis to obtain formulas D', C_0' and C_1' which are respectively equivalent to D, C_0 and C_1. Then,

$$\begin{aligned} D^*_\Omega(C') &\leq 2 + \max\{D^*_\Omega(D'), D^*_\Omega(C_0'), D^*_\Omega(C_1')\} \\ &\leq 2 + 2\log_{3/2}(\tfrac{2}{3}m) \\ &= 2 + 2(\log_{3/2} m - 1) = 2\log_{3/2} m. \end{aligned}$$

Comparing leafsizes, we see that

$$\text{leafsize}\,(C') \leq 2\,\text{leafsize}\,(D') + \text{leafsize}\,(C_0') + \text{leafsize}\,(C_1').$$

By induction, we have

$$\text{leafsize}\,(C') \leq 2s^\alpha + 2(m-s)^\alpha.$$

By calculus, the maximum value occurs when $s = \frac{1}{3}m$. Thus, we have

$$\begin{aligned} \text{leafsize}\,(C') &\leq 2(\tfrac{1}{3}m)^\alpha + 2(\tfrac{2}{3}m)^\alpha \\ &\leq 2m^\alpha \left(\frac{1+2^\alpha}{3^\alpha}\right) \\ &\leq m^\alpha. \end{aligned}$$

(b) This is a straightforward modification of part (a).

Q.E.D.

Corollary 12 *Let $\Omega = \{\neg, \vee, \wedge\}$ and $f : \{0,1\}^n \to \{0,1\}$ be a boolean function. Then,*

$$D^*_\Omega(f) \leq \max\{1, \frac{2}{\log 3 - 1}\log(L^*_\Omega(f) + 1)\},$$

and

$$D_\Omega(f) \leq \max\{1, \frac{2}{\log 3 - 1}\log(L_\Omega(f) + 1)\}.$$

Corollary 13 *Let* $\Omega = \{\neg, \vee, \wedge\}$ *and* $f : \{0,1\}^n \to \{0,1\}$ *be a boolean function. Then,*

$$\log(L_\Omega(f) + 1) \leq D_\Omega(f) \leq 1 + \frac{2}{\log 3 - 1} \log(D_\Omega(f) + 1)$$

Homework⋆ Homework problems 7 and 8 sketch some improvements to the constants in Spira's theorem.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: January 21, 1992 **Topic: Krapchenko's Lower Bounds**
Instructor: Sam Buss **Notes by: Matthew Clegg**

Today we will give a quadratic lower bound on formula size for certain natural functions computed over the set of gates $\Omega = \{\neg, \vee, \wedge\}$. This will establish a lower bound of $\alpha \geq 2$ for part (b) of Spira's theorem from last time.

Definition 9 *Let σ and τ be truth assignments to $x_1, x_2, \ldots, x_n$, i.e., $\sigma, \tau : \{x_1, x_2, \ldots, x_n\} \to \{0, 1\}$. Then, σ and τ are* neighbors *if $\sigma(x_i) \neq \tau(x_i)$ for exactly one i.*

Definition 10 *If A and B are sets of truth assignments to $x_1, x_2, \ldots, x_n$, then we define*

$$N(A, B) = \{\, (\sigma, \tau) \in A \times B \mid \sigma \text{ and } \tau \text{ are neighbors} \,\}.$$

Theorem 14 (Krapchenko) *Let $f : \{0,1\}^n \to \{0,1\}$ be an n–ary boolean function. Suppose that $A \subseteq f^{-1}(0), B \subseteq f^{-1}(1)$, and that neither A nor B are empty. Then,*

$$L^*_\Omega(f) \geq \frac{|N(A,B)|^2}{|A||B|} - 1,$$

where $\Omega = \{\neg, \vee, \wedge\}$.

Proof. (Paterson) Let φ be an arbitrary Ω–formula computing f, and let s be the leafsize of φ. We will show that

$$s \geq \frac{|N(A,B)|^2}{|A||B|}.$$

We proceed by induction on the L_Ω–size of φ. If φ has no gates, then φ is x_i for some i. Then, $N(A, B) \leq A$ because each element in A has at most one neighbor in B and similarly $N(A, B) \leq B$. Hence,

$$\frac{|N(A,B)|^2}{|A||B|} \leq 1 \leq s.$$

Now suppose that φ has at least one gate. We divide our argument into cases, according to the topmost connective in φ.

$\varphi = \neg\psi$**:** In this case, ψ has fewer gates than φ but the same number s of leaves. Therefore, the result holds by induction.

$\varphi = \psi \wedge \chi$**:** Let s_ψ and s_χ be the respective leaf sizes of ψ and χ. Then, $s = s_\psi + s_\chi$. Since $B \subseteq f^{-1}(1)$, we must have $B \subseteq \psi^{-1}(1)$ and also $B \subseteq \chi^{-1}(1)$. Also, we can pick $A_\psi \subseteq \psi^{-1}(0)$ and $A_\chi \subseteq \chi^{-1}(1)$ such that $A_\psi \cup A_\chi = A$ and $A_\psi \cap A_\chi = \emptyset$. Suppose first that $A_\psi = \emptyset$. Then $A_\chi = A$, so by induction we would then have

$$\frac{|N(A,B)|^2}{|A||B|} = \frac{|N(A_\chi,B)|^2}{|A_\chi||B|} \leq s_\chi \leq s.$$

Since an identical argument holds if $A_\chi = \emptyset$, we may assume that neither A_ψ nor A_χ are empty. Now let $n_\psi = |N(A_\psi, B)|$ and $n_\chi = |N(A_\chi, B)$. Note that since $A_\psi \cap A_\chi = \emptyset$ and $A_\psi \cup A_\chi = A$, it follows that $n_\psi + n_\chi = |N(A,B)|$. Also let $a_\psi = |A_\psi|$ and $a_\chi = |A_\chi|$. By our induction hypothesis, we have

$$s_\psi \geq \frac{|N(A_\psi,B)|^2}{|A_\psi||B|} \text{ and } s_\chi \geq \frac{|N(A_\chi,B)|^2}{|A_\chi||B|}.$$

Putting these facts together, we then have

$$\begin{aligned}
s &= s_\psi + s_\chi \\
&\geq \frac{|N(A_\psi,B)|^2}{|A_\psi||B|} + \frac{|N(A_\chi,B)|^2}{|A_\chi||B|} = \frac{n_\psi^2}{a_\psi|B|} + \frac{n_\chi^2}{a_\chi|B|} \\
&= \frac{n_\psi^2 a_\chi + n_\chi^2 a_\psi}{a_\psi a_\chi |B|} \\
&= \frac{(n_\psi^2 a_\chi + n_\chi^2 a_\psi)(a_\psi + a_\chi)}{a_\psi a_\chi (a_\psi + a_\chi)|B|} \\
&= \frac{n_\psi^2 a_\psi a_\chi + n_\psi^2 a_\chi^2 + n_\chi^2 a_\psi^2 + n_\chi^2 a_\psi a_\chi}{a_\psi a_\chi (a_\psi + a_\chi)|B|} \\
&= \frac{(n_\psi^2 a_\psi a_\chi + 2n_\psi n_\chi a_\psi a_\chi + n_\chi^2 a_\psi a_\chi) + (n_\psi^2 a_\chi^2 - 2n_\psi n_\chi a_\psi a_\chi + n_\chi^2 a_\psi^2)}{a_\psi a_\chi (a_\psi + a_\chi)|B|} \\
&= \frac{a_\psi a_\chi (n_\psi + n_\chi)^2 + (n_\psi a_\chi - n_\chi a_\psi)^2}{a_\psi a_\chi (a_\psi + a_\chi)|B|} \\
&\geq \frac{a_\psi a_\chi (n_\psi + n_\chi)^2}{a_\psi a_\chi (a_\psi + a_\chi)|B|}
\end{aligned}$$

$$= \frac{(n_\psi + n_\chi)^2}{(a_\psi + a_\chi)|B|}$$
$$= \frac{|N(A,B)|^2}{|A||B|}.$$

$\varphi = \psi \vee \chi$**:** The argument in this case is essentially the same as in the previous case.

Q.E.D.

Notes.

- What's going on? Let's define
$$K(f) = \max_{A,B} \frac{|N(A,B)|^2}{|A||B|}.$$
We have shown that $K(f)$ is a *formal complexity measure*, i.e., that $K(f)$ satisfies the following conditions:
  - $K(x_i) \geq 1$
  - $K(\neg f) = K(f)$
  - $K(f \wedge g) \leq K(f) + K(g)$

  These three facts alone imply Krapchenko's theorem.

- Note that Krapchenko's theorem can only prove a quadratic lower bound on formula size. A given element of A can have at most n neighbors in B, so $|N(A,B)| \leq n|A|$ and similarly $|N(A,B)| \leq n|B|$. Consequently,
$$\frac{|N(A,B)|^2}{|A||B|} \leq \frac{(n|A|)(n|B|)}{|A||B|} = n^2.$$

We will now turn to some applications of Krapchenko's theorem.

Example 3 Let's consider the case when f is a full minterm. For example, suppose that $f = x_1 \wedge x_2 \wedge \cdots \wedge x_n$. Then, B can have at most one element, which is namely the assignment of 1 to each x_i. Changing any variable from 1 to 0 gives a possible element of A. Thus, when $|B| = 1$, we may take $|A| = n$. Krapchenko's theorem then tells us that
$$L^*_\Omega(x_1 \wedge x_2 \wedge \cdots \wedge x_n) \geq n - 1.$$
In fact, it is easy to construct a formula of exactly this size computing f.

□

Example 4 We now turn to a more substantial example. We define

$$\begin{aligned}\mathbf{Parity}(x_1, x_2, \ldots, x_n) &= \begin{cases} 1 & \text{if an odd number of the } x_i\text{'s are 1} \\ 0 & \text{if an even number of the } x_i\text{'s are 1} \end{cases} \\ &= x_1 \oplus x_2 \oplus \cdots \oplus x_n\end{aligned}$$

Let $A = \mathbf{Parity}^{-1}(0)$ and $B = \mathbf{Parity}^{-1}(1)$. For any choice of the first $n-1$ inputs, x_n can be chosen so as to guarantee that $\mathbf{Parity}(x_1, x_2, \ldots, x_n) = 0$ (respectively, $\mathbf{Parity}(x_1, x_2, \ldots, x_n) = 1$), so we see that $|A| = |B| = 2^{n-1}$. Moreover, each element of $|A|$ has exactly n neighbors in $|B|$ and vice versa, so we have $|N(A, B)| = n|A| = n|B| = n2^{n-1}$. Putting these facts together, Krapchenko's theorem shows us that

$$L^*_\Omega(\mathbf{Parity}_n) \geq \frac{(n2^{n-1})^2}{(2^{n-1})(2^{n-1})} - 1 = n^2 - 1.$$

□

In fact, this bound is sharp.

Claim 15 *For $n = 2^k$, we have $L^*_\Omega(\mathbf{Parity}_n) = n^2 - 1$.*

Proof. We proceed by induction on k. For $k = 1$, the claim is obvious. For larger k, we can write

$$\begin{aligned}\mathbf{Parity}_n(x_1, x_2, \ldots, x_n) &= (\mathbf{Parity}_{n/2}(x_1, \ldots x_{n/2}) \wedge \neg \mathbf{Parity}_{n/2}(x_{n/2+1}, \ldots, x_n)) \\ &\quad \vee (\neg \mathbf{Parity}_{n/2}(x_1, \ldots, x_{n/2}) \wedge \mathbf{Parity}_{n/2}(x_{n/2+1}, \ldots, x_n)).\end{aligned}$$

This formula uses $\mathbf{Parity}_{n/2}$ four times and includes an additional 3 binary gates. Consequently, the L^*–size of this formula is $4((n/2)^2 - 1) + 3 = (n^2 - 4) + 3 = n^2 - 1$.

Q.E.D.

Example 5 As our third application of Krapchenko's theorem, we consider threshhold circuits. Define

$$\mathbf{Threshhold}_k^n(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \text{if at least } k \text{ of the } x_i\text{'s are 1} \\ 0 & \text{otherwise} \end{cases}$$

Claim 16 *We claim that* $L_\Omega^*(\mathbf{Threshhold}_k^n) \geq k(n-k+1)$.

Proof. Let A consist of the assignments where exactly $k-1$ of the x_i's are 1, and let B consist of the assignments where exactly k of the x_i's are 1. Then, $|A| = \binom{n}{k-1}$ and $|B| = \binom{n}{k}$. For any element of A, there are $n-k+1$ of the variables x_i which can be changed from 0 to 1, giving an element of B. Consequently, $|N(A,B)| = |A|(n-k+1)$. Similarly, for any element of B, there are k of the variables x_i which can be changed from 1 to 0, giving an element of A. Thus, $|N(A,B)| = |B|k$. Applying Krapchenko's theorem, we thus have

$$L_\Omega^*(\mathbf{Threshhold}_k^n) \geq \frac{(n-k+1)|A| \cdot k|B|}{|A||B|} = k(n-k+1).$$

Q.E.D.

An important special case occurs when $k = \lceil \frac{n}{2} \rceil$, which is known as the **Majority** circuit. In this case, we have

$$L_\Omega^*(\mathbf{Majority}_n) = L_\Omega^*(\mathbf{Threshhold}_{\lceil \frac{n}{2} \rceil}^n) \geq \lceil \frac{n}{2} \rceil \lceil \frac{n}{2} \rceil - 1 \geq \frac{n^2}{4} - 1.$$

It is known that **Majority** has polynomial size $\{\neg, \vee, \wedge\}$–formulas.

□

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 23 January 1992 **Topic: Neciporuk's Theorem**
Instructor: Sam Buss **Notes by: Frank Baeuerle**

Let B_2 be the set of all binary gate types.

Definition 11 *Let $f(x_1, \ldots, x_n)$ be a Boolean function. Let $Y \subset \{x_1, \ldots, x_n\}$, $|Y| = m$ and let $\sigma : Y \to \{0,1\}$ be a truthassignment to variables in Y. Then define a (n-m) -ary function $f_{|_\sigma}$ via*

$$f_{|_\sigma}(z_1, \ldots, z_{n-m}) = f(a_1, \ldots, a_n) \text{ where}$$

$$a_i = \begin{cases} \sigma(x_i) & \text{if } x_i \in Y \\ z_j & \text{otherwise, where } x_j \text{ is the } i^{th} \text{ element of } \{x_1, \ldots, x_n\} - Y. \end{cases}$$

$f_{|_\sigma}$ is called an induced subfunction of f.

Let $X = \{x_1, \ldots, x_n\}$.

Definition 12 *$N_f(Y)$ = "the number of Y- subfunctions of f " = $|\{f_{|_\sigma} : \sigma : (X - Y) \to \{0,1\}\}|$*

Example: If $\emptyset \subsetneq Y \subsetneq X$ then $N_{Parity}(Y) = 2$

Definition 13 *If C is a formula, $occur(Y, C)$ = # of times Y variables occur in C.*

Theorem 17 *(Neciporuk, 1966)*
Let $f : \{0,1\}^n \to \{0,1\}$. *Let* $\emptyset \subset Y \subset X$. *Then the number of occurrences of* Y *variables in any* B_2 *- formula computing* f *is* $\geq \log_5 N_f(Y)$ *provided* $N_f(Y) > 2$.

Proof. (due to Paterson) Obvious ways to bound $N_f(Y)$:
$N_f(Y) \leq 2^{|X-Y|}$ since there are only $2^{|X-Y|}$ many truthassignments σ: $(X - Y) \to$ {0,1}
$N_f(Y) \leq 2^{2^{|Y|}}$ since there exist at most $2^{2^{|Y|}}$ functions $Y \to \{0,1\}$
Neciporuk's bound: $N_f(Y) \leq 5^{occur(Y,C)}$ where C is a B_2- formula computing f.
Suppose occur(Y,C) = 1 and suppose we have a circuit computing the following formula:

$$(((x_i \oplus x_1) \wedge x_2) \vee x_3)$$

where $x_i \in Y$ and $x_1, x_2, x_3 \notin Y$. It can easily be seen that the only induced functions here are $x_i, \overline{x_i}$, 0 and 1.
Moral: If C is a circuit in which $X - Y$ variables can be set so as to induce g, then there exists a circuit $\hat{C}$ computing $\bar{g}$, 0, 1 with the same number of Y-variables.
Thus we may as well make X very large (keeping Y fixed) and assume formulas do not contain multiple occurrences of the same $X - Y$ variables.

Definition 14 $M_f(Y) = |\{f_{|\sigma}, \bar{f}_{|\sigma} \mid \sigma : (X - Y) \to \{0,1\}\} - \{$*constant functions 0 and 1*$\}|$

It suffices to show $occur(Y, C) \geq \log_5(2M_f(Y) + 1)$
since then $occur(Y, C) \geq \log_5(M_f(Y) + 2) \geq \log_5(N_f(Y))$ where C computes f.

Now the proof is by induction on the size of a B_2-formula C.
Base Case (1): $occur(Y, C) = 0$
Then $M_f(Y) = 0$, so $\log_5(2M_f(Y) + 1) = 0$
Base Case (2): C is just x_i where $x_i \in Y$. Then $M_f(Y) = 2$, so $\log_5(2M_f(Y) + 1) = \log_5(5) = 1$.
Induction Step (1): C has top most gate of arity 1
Then C is $\odot(D)$ and $occur(Y, C) = occur(Y, D)$. Let D compute g.

Claim 18 $M_f(Y) \leq M_g(Y)$

$$\odot \text{ computes } \begin{cases} \text{identity function} \\ \text{negation} \end{cases} \Big\} M_f(Y) = M_g(Y)$$
$$\begin{cases} \text{constant 0} \\ \text{constant 1} \end{cases} \Big\} M_f(Y) = 0$$

Induction Step (2): C has top most gate of arity 2
Then C is $\odot(D, E)$. Let D, E compute g, h respectively.

Claim 19 $M_f(Y) \leq 2M_g(Y) \cdot M_h(Y) + M_g(Y) + M_h(Y)$

What can $g_{|_\sigma}$ and $h_{|_\sigma}$ be? And what do they contribute to the $M_f(Y)$ count?

a. $g_{|_\sigma}$ and $h_{|_\sigma}$ can be constant, so $f_{|_\sigma}$ is also constant. Therefore there is no contribution to $M_f(Y)$.

b. $g_{|_\sigma}$ is constant but $h_{|_\sigma}$ is not constant. In this case $f_{|_\sigma}$ is either $\bar{h}_{|_\sigma}$ or $h_{|_\sigma}$ or 0 or 1. So this contributes $\leq M_h(Y)$ to $M_f(Y)$.

c. Similarly where $g_{|_\sigma}$ is not constant but $h_{|_\sigma}$ is. This contributes $\leq M_g(Y)$ to $M_f(Y)$.

d. Both $g_{|_\sigma}$ and $h_{|_\sigma}$ are not constant. There are $\leq M_g(Y) \cdot M_h(Y)$ many ways to have $g_{|_\sigma}$ and $h_{|_\sigma}$ be non-constant functions. We need to double this to put in negations as well. Thus $g_{|_\sigma} \odot h_{|_\sigma}$ and $\overline{g_{|_\sigma} \odot h_{|_\sigma}}$ contribute $\leq 2M_h(Y) \cdot M_g(Y)$ to $M_f(Y)$.

Altogether $g_{|_\sigma} \odot h_{|_\sigma}$ and $\overline{g_{|_\sigma} \odot h_{|_\sigma}}$ contribute $\leq 2 \cdot M_h(Y) \cdot M_g(Y) + M_g(Y) + M_h(Y)$ to $M_f(Y)$.
This proves the Claim!
This bound can be shown to be tight.

Now we prove the theorem as follows:

$$\begin{aligned} 2 \cdot M_f(Y) + 1 &\leq 4 \cdot M_h(Y) \cdot M_g(Y) + 2 \cdot M_g(Y) + 2 \cdot M_n(Y) + 1 \\ &= (2 \cdot M_g(Y) + 1) \cdot (2 \cdot M_g(Y) + 1) \end{aligned}$$

So we have

$$\log_5(2 \cdot M_f(Y) + 1) \leq \log_5(2 \cdot M_h(Y) + 1) + \log_5(2 \cdot M_g(Y) + 1)$$

$$\begin{aligned} &\leq occur(Y, D) + occur(Y, E) \\ &= occur(Y, C) \end{aligned}$$

Q.E.D.

Method of applying Neciporuk's Theorem:
Given a Boolean function f:
Pick $Y_1, \ldots, Y_m$ disjoint subsets of $\{x_1, \ldots, x_n\}$. If C is a B_2-formula for f, then the

$$\begin{aligned} leafsize \text{ of } C &\geq \sum_{i=1}^{m} occur(Y_i, C) \\ &\geq \sum_{i=1}^{m} \log_5 N_f(Y_i) \end{aligned}$$

If we're lucky and Y_i has size $\approx \log n$, so $m \approx \frac{n}{\log n}$ and each $N_f(Y) \approx 2^n$, so $\log_5 N_f(Y_i) = \Omega(n)$.
Best case results are $\Omega(\frac{n^2}{\log n})$.

Examples: Clique, Matching, Determinant.

Application: DUP_n "duplicate"
Let $n = (m+1) \cdot (\lceil \log m \rceil + 1)$ and $b = \lceil \log m \rceil + 1$.
Break $x_1, \ldots, x_n$ into (n+1) blocks of size b, i.e. $x_1, \ldots, x_b, \ldots, x_{i \cdot b+1}, \ldots, x_{(i+1) \cdot b}, \ldots$ and define

$$DUP_n(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } \exists i \neq j \text{ s.t. } x_{i \cdot b+k} = x_{j \cdot b+k} \text{ for } k = 1, \ldots, b \\ 0 & \text{otherwise} \end{cases}$$

Let $Y_i = \{x_{i \cdot b+1}, \ldots, x_{(i+1) \cdot b}\}$ be the i^{th} block, $i = 0, 1, \ldots, m$.

Claim 20 $N_f(Y_i) \geq \frac{2^{2m}}{m}$ *where* $m > 2$

$$\begin{aligned} N_f(Y_i) &= \left(\begin{array}{c} 2^b \\ m \end{array} \right) + 1 \geq \left(\begin{array}{c} 2m \\ m \end{array} \right) = \frac{(2m)!}{(m!)^2} \\ &\approx \frac{\sqrt{2\pi \cdot 2m} \cdot (\frac{2m}{e})^{2m}}{(\sqrt{2\pi \cdot m})^2 \cdot (\frac{m}{e})^{2m}} \approx \frac{\sqrt{\pi m} \cdot 2^{2m} \cdot (\frac{m}{e})^{2m}}{\pi m \cdot (\frac{m}{e})^{2m}} \\ &\approx \frac{2^{2m}}{\sqrt{\pi m}} > \frac{2^{2m}}{m} \end{aligned}$$

By Neciporuk,

$$\begin{aligned} L^{\star}_{B_2}(DUP_n) &\geq \sum_i \log_5(N_{DUP_n}(Y_i) \\ &\geq (m+1) \cdot \log_5 \frac{2^{2m}}{m} \\ &= (m+1) \cdot (2m \cdot \log_5 2 - \log_5 m) \\ &\geq \frac{(m+1) \cdot m}{c} \text{for some constant } c > 1 \text{ (} c = 2 \text{ works)} \end{aligned}$$

Recall $n = (m+1)\lceil \log m \rceil + 1$. Thus obviously $\log n > \log m$ and hence

$$m + 1 = \frac{n}{\lceil \log m \rceil + 1} \geq \frac{n}{\lceil \log n \rceil + 1}$$

hence $m \geq \frac{n}{c' \log n}$ for some constant c' and therefore

$$L^{\star}_{B_2}(DUP_n) \geq \frac{n^2}{c''(\log n)^2}$$

for some constant c''.

Definition 15 *(Paul '77) Let $ISA_n(x_1, \ldots, x_n)$ ("indirect adressing") be defined as follows: $n = 2 \cdot p + k$, where $p = 2^{2^l}, k = \log p - \log\log p$. We define $ISA_n(z_1, \ldots, z_k, x_1, \ldots, x_p, y_1, \ldots, y_p)$, let $(z_1, \ldots, z_k)_2 = i$ be the number with base 2 representation $z_1, \ldots, z_k$. Let $b = \log p$ (blocksize for $\vec{x}$) and $j = (x_{i \cdot b+1}, \ldots, x_{(i+1) \cdot b})_2$. Then $ISA_n(\vec{z}, \vec{x}, \vec{y}) = y_j$ (by definition).*

Homework #10: Prove that $L^{\star}_{B_2}(ISA_n) = \Omega(\frac{n^2}{\log n})$
Hint: Let $Y_i = \{x_{i \cdot b+1}, \ldots, x_{(i+1) \cdot b}\}$ be the i^{th} block of $\vec{x}$.

Homework #11: Show $C^{\star}_{\{\neg, \vee, \wedge\}}(ISA_n) = \mathcal{O}(n)$
Hint: Remember the circuits for computing all full minterms.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 28 January 1992 **Topic: Turing Machines and Circuits**
Instructor: Sam Buss **Notes by: Frank Baeuerle**

The first half hour of lecture was spent on a joint effort by Sam and the class to solve a problem left on the blackboard.

Problem: (comp.theory) Given a string $\sigma \in \{a\# \cup b\# \cup \#\}^\star$ produce the unique string in $\{a\# \cup b\# \cup 0\#\}^\star$ which yields σ when you delete all the 0's. That is find all adjacent #'s and insert a 0 between them and insert a 0 at the beginning if the first symbol is a # sign. Question: Can this be done in size $\mathcal{O}(n)$?

Definition 16 *$f_n : \{0,1\}^n \to \{0,1\}$ is symmetric iff $f(x_1, \ldots, x_n)$ depends only on the number of $x_i's = 1$.*

We could not verify the bound but we reduced parity to this problem and in fact all symmetric functions.

Theorem 21 *Any symmetric function $f : \{0,1\}^n \to \{0,1\}$ can be reduced to the circuits solving the comp.theory problem for m = 2n +2 inputs.*

Proof. (for reducing parity only) Given a string δ over $\{0,1\}^\star$ of length n append it at the beginning with $a\#$ and at the end with a string $a\#b\#a\#\cdots b\#$ of length 2n (with a's and b's alternating). Now replace 0 with $\#\#$ and 1 with $a\#$. Now use the circuit that computes the comp.theory problem to produce the string σ corresponding to the given string and look at the $2n + 3^{rd}$ and $2n + 4^{th}$ symbol of σ. Now the number of 0's is odd iff we see $b\#$.

Q.E.D.

This argument can be modified to prove the theorem in full generality.

Definition 17 *A language is a subset* $L \in \{0,1\}^\star$

A language L determines a family $\{f_n\}_{n=0}^{\infty}$ of Boolean functions where f_n is $n-ary$ and $f_n(x_1,\ldots,x_n)=1$ iff $x_1,\ldots,x_n \in L$ and vice versa $\{f_n\}_{n=0}^{\infty}$ determines L.
Now ask what is the size of circuits computing this L ? (This puts Turing machines and circuits into the same setting !)
Example: Hamiltonian Circuit: for $n = \frac{m\cdot(m-1)}{2}$ let

$$HC_n(x_1,\ldots,x_n) = \begin{cases} 1 & \text{if G has a hamiltonian circuit} \\ 0 & \text{otherwise} \end{cases}$$

where G is the graph on m nodes with $\#x_i's$ indicating whether edges are present. (x_i says whether an edge is present)
Problem: This is not defined for all n ! To overcome this let $HC_n(\vec{x}) = 0$ for all other n.
Notation: n is the # of inputs to the circuit.

Definition 18 *We say* $\{f_n\}$ *or* L *has* Ω*-circuits of size* $s(n)$ *(or* $\mathcal{O}(s(n))$ *etc.) iff* $\forall n \exists \Omega$*-circuit* C_n *of size* $\leq s(n)$ *(*$\mathcal{O}(s(n))$ *etc.) that computes* f_n.

Definition 19 $\{f_n\}$ *and/or* L *has polynomial circuits iff it has circuits of size* $(n+1)^{\mathcal{O}(1)}$.

Definition 20 $L \in TIME(t(n))$ *iff* L *is accepted by some deterministic multitape Turing machine which always halts in* $\leq t(n)$ *steps.*

Theorem 22 *If* $L \in TIME(t(n))$ *then* L *has* $\{\neg,\vee,\wedge\}$ *circuits of size* $\mathcal{O}(t(n)^2)$.

Corollary 23 *If $L \in P$ then L has polynomial size circuits.*

Observation: If $\{HC_n\}_n$ does not have polysize circuits then $P \neq NP$.

Convention: Let the Turing Machine alphabet be $0, 1, blank$, $c_1, \ldots c_m$. Initial tape setting is the input surrounded by blanks and the tapehead at the leftmost input symbol.

Proof. (of Theorem) Idea: The circuit will have gates giving facts like " i^{th} square of q^{th} tape has symbol c_p at time j"
Fix some Turing machine M accepting L in time $t(n)$. To avoid problems with position of tapehead w.r.t. i^{th} symbol look at the relative position of the tapehead. The circuit has nodes called

- $TR(q, i, j, p)$ which are true iff c_p is in the i^{th} tape square to the right of tapehead on tape $\#q$ at time j, where $i \geq 0$.
- $TL(q, i, j, p)$ which are true iff c_p is in the i^{th} tape square to the left of tapehead on tape $\#q$ at time j, where $i > 0$.
- $S(j, p)$ which are true iff we are in state p at time j.
- $ML(q, j)$ which are true iff tapehead on tape q moves left in step j. (step j takes machine from time j to time $j + 1$)
- $MR(q, j)$ which are true iff tapehead on tape q moves right in step j.
- $W(q, j, p)$ which are true iff M writes c_p in tape q in step j.

Claim 24 *There are $\mathcal{O}(t(n)^2)$ many nodes of these types (for $M's$ execution on inputs of length n).*

Pf. q and p range over fixed finite sets. We have
$0 \leq j \leq t(n)$, where $t(n)$ is the time bound of M
$0 \leq i \leq t(n)$, since space is bounded by time. q.e.d. Claim.

$ML(q,j)$, $MR(q,j)$, $W(q,j,p)$ are computed as fixed Boolean functions of $S(j,p')$ and $TR(q',0,j,p'')$ where p' ranges over states, p'' ranges over alphabet symbols and q' ranges over tapes
$TR(q,i,j+1,p)$ is a fixed Boolean function of $ML(q,j)$, $MR(q,j)$, $TR(q,i,j,p)$, $TR(q,i-1,j,p)$, $TL(q,i+1,j,p)$ for $i>0$.
$TR(q,0,j+1,p)$ is a fixed Boolean function of $ML(q,j)$, $MR(q,j)$, $W(q,j,p')$ for all p', $TR(q,0,j,p)$, $TR(q,1,j,p)$, $TL(q,1,j,p)$.
Similarly for $TL(\cdots)$ and $S(\cdots)$.
W.l.o.g. M accepts iff $S(t(n),r)$ for some fixed state r.

$$TR(q,i,0,p) = \begin{cases} = x_i & \text{iff } p=1,\ 0 \le i < n \text{ and } q=1 \text{ (input tape)} \\ = \bar{x_i} & \text{iff } p=0,\ 0 \le i < n \text{ and } q=1 \text{ (input tape)} \\ = 1 & \text{iff } p = blank \text{ and } x,q \text{ do not satisfy either of the above} \\ = 0 & \text{otherwise} \end{cases}$$

$$TL(q,i,0,p) = \begin{cases} = 1 & \text{iff } p = blank \\ = 0 & \text{otherwise} \end{cases}$$

$S(0,p) = 1$ iff $p = 1$ (initial state)
By induction on time (on j) the circuit accurately represents the execution of M. Overall size is still $\mathcal{O}(t(n)^2)$ since each node is computed by a fixed size Boolean circuit. This is because we have $\mathcal{O}(t(n)^2)$ "named" nodes each computed by a constant size circuit.

Q.E.D.

Encoding a circuit: A circuit is encoded by a set of tuples, nodes are given integer labels. The tuples are:

- (i,j,l) node i is the l^{th} input to node j
- (i,τ) node i has type τ, τ can be $\vee, \wedge, \neg, \cdots$ or $\tau = j$ meaning x_j.

These tuples can be encoded into binary strings over $\{0,1\}^\star$ and can be manipulated by Turing machines.

Corollary 25 *If $L \in P$ then L has polynomial size circuits, and furthermore there is a Logspace Turing machine (transducer) which on input $x_1, \ldots, x_n \in \{0,1\}^\star$ outputs a circuit over $\{0, 1, \neg, \vee, \wedge\}$ which has value 1 iff $x_1, \ldots, x_n \in L$.*

Proof. Observe that the previous construction can be performed by a *Logspace* transducer. Node numbers can be

$$q \cdot 2^{4 \cdot d} + i \cdot 2^{3 \cdot d} + p \cdot 2^d + 1$$

where $d = \lceil c \cdot \log n \rceil$ where n^c bounds the run time of the Turing machine accepting L. This could be $TR(q, i, j, p)$. The *Logspace* machine also replaces input gates $x_1, \ldots, x_n$ by 0 or 1 value that were input to the logspace machine. The result of the logspace computation is a variable-free circuit. However the True/False value of the circuit can not in general be computed in *Logspace* unless $P = Logspace$.

Q.E.D.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: January 30 **Topic: Circuit Value Problem & NC/AC Hierarchy**
Instructor: Sam Buss **Notes by: John Lillis**

From last lecture, we have the following result from Ladner ['75].

Theorem 26 *Let $L \in TIME(t(n))$. Then L has $\mathcal{O}(t(n)^2)$ size circuits.*
Note. *Fischer and Pippenger have improved this to $\mathcal{O}(t(n) \cdot \log(t(n)))$.*

Corollary 27 *If $L \in P$ then $\exists$ a log-space machine which on input $x_1x_2...x_n$ outputs a variable-free circuit of polynomial size which outputs 1 if and only if $x_1x_2...x_n \in L$.*

Definition 21 *The Circuit Value Problem, CVP, is the set of encodings of circuits with output value 1. In other words, an instance of CVP is a circuit over, say, $\{\neg, \vee, \wedge\}$ and values for the input variables of the circuit. We want to answer the question: Is the output value 1?*

Theorem 28 (Ladner 1975) *CVP is complete for P under many-one log-space reductions.*

Proof.

(a) CVP is in P. The obvious algorithm (ie, bottom-up evaluation) is polynomial time.

(b) If $L \in P$ then $\exists$ a logspace function f such that $\forall\ w \in \{0,1\}^*$, $w \in L \Leftrightarrow f(w) \in CVP$. [Ie, f performs a many-one reduction]

Let M be a Turing Machine accepting L in polynomial time. Let $f(w)$ output a circuit with value 1 if and only if M accepts w. Note that this is possible by *Corollary* 1. So, $w \in L \Leftrightarrow f(w) \in CVP$.

Q.E.D.

We next discuss circuits with Unbounded Fanin.
Notation: $U_f = \{\neg$, unbounded fanin $\vee$, unbounded fanin $\wedge\}$.
Note. Often negations are "pushed" to the inputs using DeMorgan's Law. This at most doubles the number of $\wedge$ and $\vee$ gates. This is because an $\wedge$ or $\vee$ gate may feed into both a $\neg$ gate and some other gate, so we still need the output of the "old" gate after the $\neg$ has been pushed down.

Definition 22 *$\{f_n\}_n$ is in NC^k (for fixed k) if and only if functions f_n have $\{\neg, \vee, \wedge\}$-circuits simultaneously of polynomial size and depth $\mathcal{O}\left((\log n)^k\right)$.*

Definition 23 *$\{f_n\}_n$ is in AC^k if and only if functions f_n have unbounded-fanin (**UF**) circuits (again over $\{\neg, \vee, \wedge\}$) simultaneously of polynomial size and depth $\mathcal{O}\left((\log n)^k\right)$.*
Note. *AC^0 = functions with constant depth, poly-size **UF** circuits.*

Note. For AC^k and NC^k, circuits may be non-uniform.

Definition 24 *$\{f_n\}_n$ is in logspace-uniform AC^k (or NC^k) if and only if $\exists$ a logspace Turing Machine which on input 0^n, outputs a circuit C_n computing f_n such that C_n has polynomial size and $\mathcal{O}\left((\log n)^k\right)$ depth using UF (or $\{\neg, \vee, \wedge\}$ for NC^k) gates.*

Fact: $\exists$ a non-recursive language L in AC^0.

Proof. Let $g : \mathbb{N} \to \{0, 1\}$ be any non-recursive function. Define $w \in L \Leftrightarrow g(\mid w \mid) = 1$.
Now, L is in AC^0 since C_n is either just constant **0** or constant **1**.

Q.E.D.

Theorem 29 *(a) $AC^k \subseteq AC^{k+1}$ where $k \geq 0$.*

(b) $NC^k \subseteq NC^{k+1}$ where $k \geq 1$.

(c) $NC^k \subseteq AC^k$ where $k \geq 1$.

(d) $AC^k \subseteq NC^{k+1}$ where $k \geq 0$.

Proof. (a) - (c) should be obvious.
(d): In an UF circuit of size n^{c_1} and depth $c_2 \cdot (\log n)^k$, each $\wedge$ or $\vee$ gate has fanin $< n^{c_1}$. Replace such a gate by a balanced tree of fanin-2 $\wedge$ or $\vee$ gates respectively. This tree has size $< n^{c_1}$ (ie, there is one leaf for every two inputs). It also has depth $\leq \log n^{c_1} = c_1 \cdot \log n$. Doing this for all gates gives a $\{\neg, \vee, \wedge\}$-circuit of size $n^{c_1} \cdot n^{c_1} = n^{2c_1}$ and depth $c_2 \cdot (\log n)^k \cdot c_1 \cdot \log n = c_1 c_2 \cdot \log n^{k+1}$.

Q.E.D.

Theorem 30 *Let B_2 be the set of all binary gates.*
If C is a B_2-circuit of depth d and gate size s and if $c < d$, then there is an equivalent UF-circuit of depth $2 \cdot \lceil \frac{d}{c} \rceil + 1$ and of size $\leq 2sd \cdot (2^{2^c} + 1)$.

Corollary 31 *If $L \in NC^k$ then L has UF-circuits of polynomial size and depth $\mathcal{O}\left(\frac{(\log n)^k}{\log \log n}\right)$.*

Proof. Take $c = \log \log n$ and $2^{2^c} = n$.
Note. This corollary slightly improves (c) of the previous theorem.

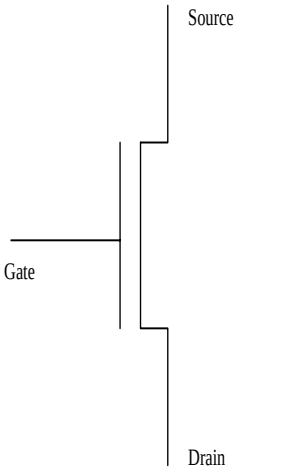
Q.E.D.

Proof. [Proof of main theorem...]
We will proceed in 3 steps.
Step 1: We will make circuit C "layered". Each node in C has maximum path-length from an input, called the *level* of the node. We will modify C so that level l gates have inputs of level exactly $l - 1$.
We accomplish this by adding extra "delayed" copies of gates using identity gates (which are in B_2).
This increases the total number of gates to $s \cdot d$ at most (d gates in new circuit for each gate in old circuit).
Step 2: We will take our leveled circuit of depth d and split it into layers of depth c each. Note that there are $\lceil \frac{d}{c} \rceil$ such layers.
Now, we know that any gate at level i+c depends on at most 2^c signals at level i (acheived when the gate in question roots a full binary tree of height c.
Now, we replace the levels i through $i+c$ by disjunctive normal form circuits of depth 3 (using $\{\neg, \vee, \wedge\}$). The top $\wedge$-gate will have fanin $\leq 2^{2^c}$ (one input for each satisfying assignment to the $\leq 2^c$ variables it depends on). The $\wedge$-gates will have fanin $\leq 2^c$. So for each gate at level $i + c$ we need $\leq 2^{2^c} + 1$ many $\wedge$ and $\vee$ gates.
Step 3: Using DeMorgan's Law, we push all $\neg$ gates to the inputs at most doubling the size. This gives C.

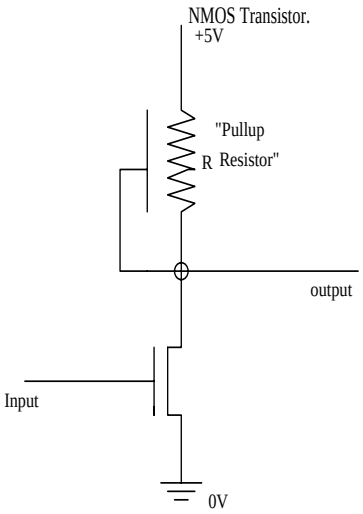
Q.E.D.

Note. Depth $\lceil \frac{d}{c} \rceil + 1$ depth is also achievable by a different proof.

Now we have an aside on real-world NMOS circuits.



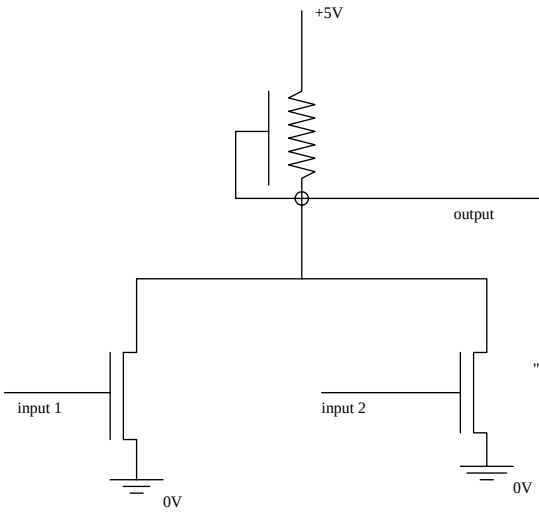
If gate wire has "high" voltage (say >3 volts) then there is low resistance between the source and drain. If gate wire has low voltage (say <1 volt) then there is high resistance between source and drain.



If input is high then the transistor "conducts" with resistance say < 1/4 R. This results in the output being "pulled to ground" (ie, low output). If input is low then the transistor has high resistance (say 4R) so the output remains high.

Note: The pullup resistor is actually another transistor for technical reasons unknown to the scribe.

NOT gate in NMOS.



Note that if either input is high then the output is drawn to ground and becomes low.
Also, we can create multiple input NOR gates by adding more "pulldown" transistors (within some limit dependent on technology I assume).

A 2-input NOR gate

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: Februrary 4, 1992 **Topic: T.M. - Circuit theorems and Addition circuits**

Instructor: Sam Buss **Notes by: John Lillis**

Theorem 32 (Borodin?) *If L is a language recognized by a non-deterministic turing machine using space $s(n)$ (where n is the input length and $s(n) \geq \log n$), then L has UF-circuits which are simultaneously of depth $\mathcal{O}(s(n))$ and size $2^{\mathcal{O}(s(n))}$.*
Remark: $NSPACE(S(n)) \subseteq TIME(2^{\mathcal{O}(S(n))}) \subseteq CIRCUIT - SIZE(2^{\mathcal{O}(S(n))})$
First $\subseteq$ we know by Savitch's theorem and the second is by Ladner's theorem.
The new contribution of theorem 32 is that depth can be $\mathcal{O}(S(n))$.

Proof. A **configuration** of a Nondeterministic Turing Machine (NDTM) M consists of

(1) Contents of work tapes and tape-head position for each work tape.

(2) State of machine.

(3) Tape head position on input tape (When $S(n) < n$, it is important to use head position instead of input tape contents).

Claim 33 *There are $\leq c^{S(n)}$ configurations for some constant c and fixed n.*

Proof. The number of configurations is bounded by
$\alpha^{S(n)\cdot k} \cdot 2^{S(n)\cdot k} \cdot q \cdot n \leq (\alpha^k \cdot 2^k \cdot q \cdot 2)^{S(n)}$
Since $S(n) \geq \log n$, ie $2^{S(n)} \geq n$. Where $\alpha =$alphabet size, $k =$ number of tapes and $q =$ number of states in M.

Q.E.D.

So, on inputs of length n, M can accept within $c^{S(n)}$ steps if it accepts at all.
Now, there is a fixed starting configuration $INIT$ and $wlog$, there is a fixed accepting configuration $ACCEPT$ and further assume that M loops in $ACCEPT$ configuration upon accepting.
Let $d = \lceil \log c \rceil$ such that $2^{d \cdot S(n)} \geq c^{S(n)}$.
An UF-circuit for language L has nodes called (C_1, i, C_2) where C_1 and C_2 are configurations as above and $i \in \{0, 1, \ldots, d \cdot S(n)\}$.
The meaning of this node is "M can go from configuration C_1 to C_2 in exactly 2^i steps."

$$Node(C_1, i+1, C_2) = \begin{cases} 1 & \text{if } \exists \text{ configuration } C_3 \text{ s.t. } ((C_1, i, C_3) \wedge (C_3, i, C_2)) \\ 0 & \text{otherwise} \end{cases}$$

[Ie, Savitch's proof technique. Also, this can be computed with a single UF-$\vee$ gate]
Nodes $(C_1, 0, C_2)$ have value of one of $\{0, 1, x_i, \neg x_i\}$ where i = tape head position in configuration C_1.
We also have output node $(INIT, d \cdot S(n), ACCEPT)$
Each signal $(C_1, i+1, C_2)$ is computed by an Unbounded Fanin $\vee$ gate of fanin n (from $\wedge$-gates):

$$(C_1, i+1, C_2) \equiv \bigvee_{C_3} [(C_1, i, C_3) \wedge (C_3, i, C_2)]$$

Depth of this circuit is $2d \cdot S(n) = \mathcal{O}(S(n))$.
Size of circuit (number of gates):
$(d \cdot S(n) + 1) \cdot (c^{S(n)})^2$ many UF-$\vee$'s for finding the existence of "C_3's".
$(d \cdot S(n) + 1) \cdot (c^{S(N)})^3$ many $\wedge$'s.
Plus n $\neg$'s.
In total, there are $2^{\mathcal{O}(S(n))}$ gates.

Q.E.D.

Remark: Only the UF-$\vee$'s have unbounded fanin and $\neg$'s are only at the inputs. This is called "semi-unbounded fanin."

Corollary 34 *Logspace* $\subseteq NL \subseteq AC^1$ *where* NL *is "Nondeterministic Logspace."*

Proof. By above theorem, NL has UF-circuits of depth $\mathcal{O}(\log n)$ and size $2^{\mathcal{O}(\log n)} = n^{\mathcal{O}(1)}$.
So $AC^0 \subset NC^1 \subseteq L \subseteq NL = co - NL \subseteq AC^1 \subseteq NC^2 \subseteq AC^2 \subseteq \ldots$

Where $L = Logspace$ and the first containment is strict. Also note that $NL = co - NL$ is due to Immerman and Szelepcsényi.

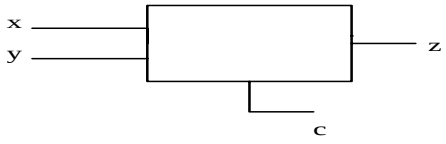
Q.E.D.

Analogies (to within polynomial factors):
Uniform Circuit Size $\approx$ Turing Machine time.
Uniform Circuit depth $\approx$ Turing Machine space.

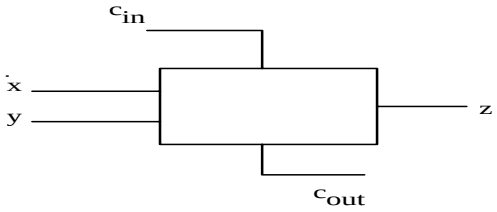
New Topic: Circuits for Addition

The "School Method."
Inputs: 2 non-negative n-bit integers.
$x_{n-1}, \ldots, x_0, y_{n-1}, \ldots, y_0$
These represent $\sum_{i=0}^{n-1} x_i$ and $\sum_{i=0}^{n-1} y_i$.
Outputs: $n+1$ bits $z_n, \ldots, z_0$ giving the binary representation of their sum.

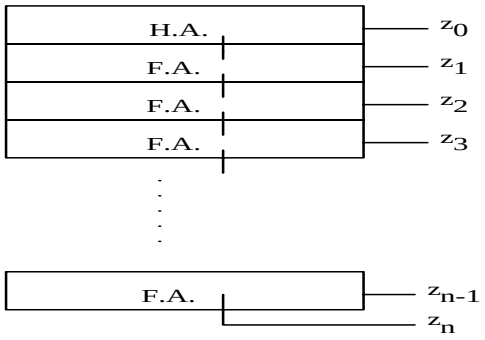
We will use two basic building blocks: The **Half Adder** and the **Full Adder**.
The Half Adder has inputs x and y and outputs z and c where $z = x \bigoplus y$ and $c = x \wedge y$.
The Full Adder has inputs x, y and c_{in} (for "carry-in") and outputs z and c_{out} where
$z = x \bigoplus y \bigoplus c_{in}$ and $c_{out} = (x \wedge y) \vee (x \wedge c_{in}) \vee (y \wedge c_{in}) = (x \wedge y) \vee ((x \bigoplus y) \wedge c_{in})$
So for our circuit, we will have one Half-Adder producing z_0 and its carry out will feed to a Full Adder's carry in. We have $n-1$ such full adders feeding each other with the $(n-1)$th producing z_{n-1} and z_n (on it's c_{out} wire). See drawing.



A Half-Adder (HA)

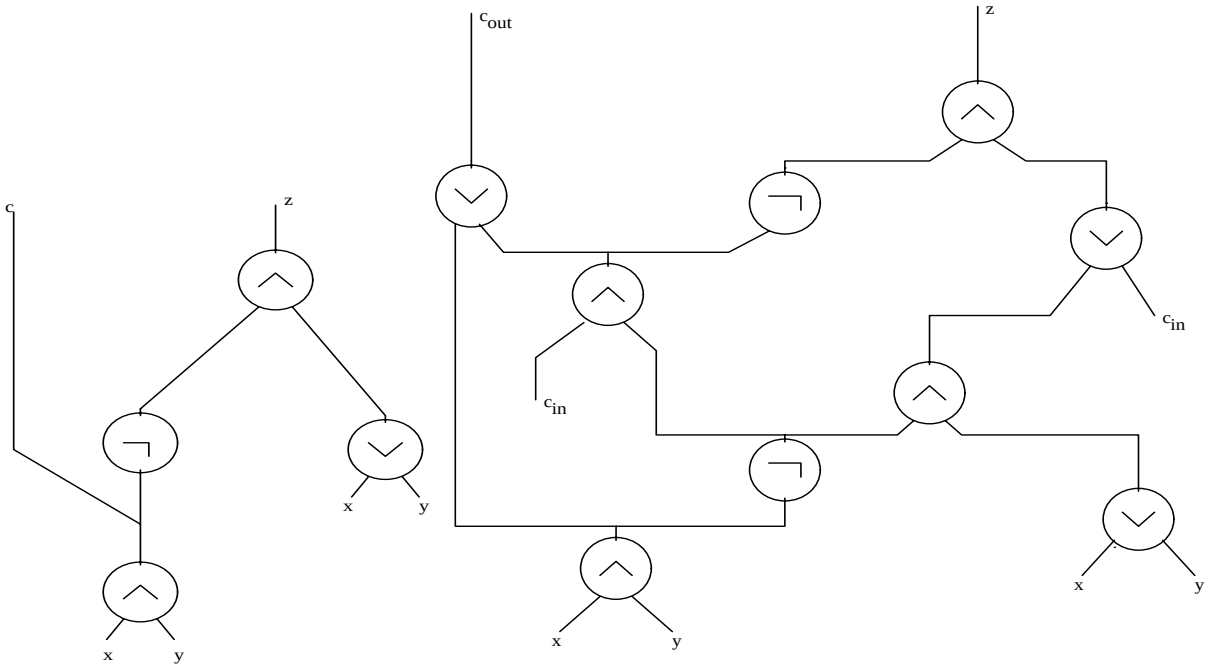


A Full-Adder (FA)



A complete n-bit adder.

How to implement these building blocks:



Homework (*): Is 9 (or 7) the optimal size (or *-size)?

Note: There is an error in the circuit above in the computation of c_{out}. There are two easy fixes, neither one changes the gates in the circuit.

Size of the "School Adder" $= 9(n-1)+4 = 9n-5$
Depth = (depth in HA to c) + (depth from c_{in} to c_{out} in $n-2$ FA's) + (depth in last FA (to c_{out})) =
$1+2(n-2)+3 = 2n$ (at least for $n \geq 3$).
We would like to improve on this linear depth (the size is good).

Theorem 35 *Addition is in AC^o.*

Proof. Let c_i be the carry into column i (ie, the 2^i position).
Carries are "started" or generated in column j if and only if $x_j = y_j = 1$.
Carries are "propegated" from colume j to column i if and only if at least one of x_k and y_k are 1 for all k $(i > k > j)$.
We can express c_i as

$$\bigvee_{j=1}^{i-1} [(x_j \wedge y_j) \wedge \bigwedge_{k=j+1}^{i-1} (x_j \vee y_j)]$$

.
Remark: $(x_j \vee x_i)$ could be replaced by $(x_j \bigoplus x_i)$.
This is a depth-3 UF-circuit.
Now the AC^0 circuit for addition.
Compute c_i as above with $UF-\wedge$ and $UF-\vee$ gates as above for $i = 1 \ldots n$. Feed c_i's, x_i's and y_i's into full adders to get output z_i.
$z_n = c_n$ and z_0 is computed by a half adder as before.

Q.E.D.

Corollary 36 *Addition is in NC^1. Ie, there are $\mathcal{O}(\log n)$-depth, $n^{\mathcal{O}(1)}$-size $\{\neg, \vee, \wedge\}$-circuits for addition.*

$\mathcal{O}(\log n)$ depth is optimal (to within a constant).
The size is pretty bad however (using the binary-tree method of converting a UF circuit to a BF circuit). It is n^2 or worse (looks like n^3). Later it will be shown how to achieve bounded fanin $\{\neg, \vee, \wedge\}$-circuits for addition with $\mathcal{O}(\log n)$ depth and size $\mathcal{O}(n)$.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 6 February 1992 **Topic: Parallel Prefix Problem**
Instructor: Sam Buss **Notes by: Goran Gogić**

We will now abstract the problem of precomputing the carry bits.

Definition 25 *A semigroup is a pair $(S, \otimes)$ where $\otimes$ is an associative operation and S is a set closed under this operation.*

Fix a finite semigroup so that each element in S can be encoded by a finite, fix-length string of 0's and 1's. Let a fixed-number of binary strings code the elements in S. The operation $\otimes$ can be computed by a fixed $\{\wedge, \vee, \neg\}$ circuit.

Parallel Prefix Problem

Inputs: $a_1, a_2, ..., a_n$, each coded in binary.

Outputs: $p_1, p_2, ..., p_n$, where $p_j = a_1 \otimes a_2 \otimes ... \otimes a_j$ for $1 \leq j \leq n$

Example:

Carry semigroup $S = \{C\,(\text{carry}), P\,(\text{propagate}), N\,(\text{no carry})\}$

$\forall x \;\; xC = C$

$\forall x \;\; xP = x$

$\forall x \;\; xN = N$

Consider

$$\begin{array}{r} 0\,1\,0\,1\,0\,1\,1 \\ +1\,1\,0\,0\,1\,1\,0 \\ \hline PCNPPCPN \end{array}$$

reversed$(PCNPPCPN) = NPCPPNCP$

We have written N in columns with two zeros, C in columns with two ones, and P in columns with one one and one zero. If we solve Parallel Prefix Problem for string $NPCPPNCP$, it will tell us where in the addition carry apear, and we will be able to do the addition.

input:$NPCPPNCP$ output:$NNCCCNCC$

There is a carry into column i of the sum iff the $i+1$ symbol in the output is a carry (C).

Convention From now on without lose of generality, we will suppose that n is a power of 2.

Theorem 37 (Ladner,Fischer 70's) *Fix a finite semigroup S, there are* $\{\wedge, \vee, \neg\}$ *circuits of size* $\mathcal{O}(n)$ *and depth* $\mathcal{O}(\log n)$ *for Parallel Prefix Problem for S.*

Proof.

Definition 26 $P_{i,j} = a_{i+1} \otimes ... \otimes a_j$

Observation 2 *There is a circuit that on input* $a_1, ..., a_n$ *outputs only* P_n *and is a binary tree of* $\otimes$ *circuits, and computes each* $P_{i,j}$ *,* $i = k \cdot 2^l, j = (k+1) \cdot 2^l$.

Circuit for Parallel Prefix Problem is:

stage 0:

The circuit above for computing all $P_{i,j}$ where $i = k \cdot 2^l$, $j = (k+1) \cdot 2^l$

stage 1:

$P_{0,\frac{3}{4}n} = P_{0,\frac{1}{2}n} \otimes P_{\frac{1}{2}n,\frac{3}{4}n}$

stage 2:

$P_{0,\frac{3}{8}n} = P_{0,\frac{1}{4}n} \otimes P_{\frac{2}{8}n,\frac{3}{8}n}$
$P_{0,\frac{5}{8}n} = P_{0,\frac{2}{4}n} \otimes P_{\frac{4}{8}n,\frac{5}{8}n}$
$P_{0,\frac{7}{8}n} = P_{0,\frac{3}{4}n} \otimes P_{\frac{6}{8}n,\frac{7}{8}n}$
. . .

stage l:

Compute $P_{0,\frac{k}{2^l \cdot n}}$ when k is odd, $1 < k < 2^l$.

There are $\mathcal{O}(\log n)$ stages. Each stage has $\mathcal{O}(1)$ depth because all $\otimes$ computations in a stage are done in parallel.

$P_{0,\frac{k}{2^l}n} = P_{0,\frac{k-1}{2^l}n} \otimes P_{\frac{k-1}{2^l}n,\frac{k}{2^l}n}$ where the first factor is known from an earlier step , and the second one is known from the zeroth step.

Total number of $\otimes$'s is less than n.

Size of all stages is $\mathcal{O}(n)$.

Depth of all stages is $\mathcal{O}(\log n)$.

The whole circuit has size $\mathcal{O}(n)$ and depth $\mathcal{O}(\log n)$.

Q.E.D.

Corollary 38 *The Addition problem has size $\mathcal{O}(n)$, depth $\mathcal{O}(\log n)$ circuits and hence a polynomial size $\mathcal{O}\left(n^k\right)$ and $\mathcal{O}(\log n)$ depth $\{\wedge, \vee, \neg\}$–formula.*

Vector Addition Problem

Inputs: $n = m \cdot p$ inputs, coding m integers of p bits each
Output: $p + \log m$ many bits giving the sum of the inputs

Applications:

1) Counting

Input: $x_1, x_2, ..., x_n$
Output: $\sum_{i=1}^{n} x_i$ coded $\lceil \log n \rceil$ bit integer.

2) Multiplication:

Input: two n-bit numbers

Output: Their product, a $2n$-bit number

School method for multiplication: Add up to n many integers at most $2n$ bit each.

We shall show that the Vector Addition Problem has a circuit of size $\mathcal{O}(n)$ and depth $\mathcal{O}(\log n)$. Hence, multiplication has a circuit of depth $\mathcal{O}(\log 2n^2) = \mathcal{O}(\log n)$ and size $\mathcal{O}(2n^2) = \mathcal{O}(n_2)$, so multiplication is in $\mathbf{NC}^1$.

Open question: Does multiplication has $\mathcal{O}(n)$ size circuits? The best known has $\mathcal{O}(n \log n \log\log n)$ size and depth $\mathcal{O}(\log n)$ circuit.

Naive approach to the Vector Addition Problem:

Just add them one by one. This solution is fine for the size but bad for the depth.

Carry Save Addition (CSA) circuit

Input: Three p-bit integers $x^0_{p-1}, ..., x^0_0, x^1_{p-1}n, ..., x^1_0, x^2_{p-1}, ..., x^2_0$

Output: Two $p+1$ bit integers, which have the same sum as the inputs.

Outputs are $y_{p-1}, ..., y_0$ and $z_{p-1}, ..., z_0$

$z_i = x^0_i \otimes x^1_i \otimes x^2_i, z_p = 0 \;\; 1 \leq i \leq p-1$

$y_1 = 0, y_{i+1} = (x^0_i \vee x^1_i) \wedge (x^0_i \vee x^2_i) \wedge (x^1_i \vee x^2_i) \;\; 1 \leq i \leq p-1$

Size is $\mathcal{O}(p)$ and depth is constant.

Definition 27 *A $4-2-CSA$ is a circuit composed of 2 CSA as above which on input: 4 numbers, p bits each outputs, 2 numbers of $(p+2)$ bits each such that the sum of the inputs is equal to the sum of the outputs.*

Now we present a circuit for Vector Addition

Stage 1: Let $m = 2^k$.

If $m \geq 4$ use $\frac{m}{4}$ 4-2-CSA's to get $\frac{m}{2}$ many $p+2$–bit integers with the same sum.

Stage 2:

If $m \geq 8$ use $\frac{m}{8}$ 4-2-CSA's to get $\frac{m}{4}$ many $p+3$–bit integers with the same sum.

$\vdots$

Until we get 2 integers $p+k$ bit each with the same sum as the original m input integers. Use a $\mathcal{O}(p+k)$ size and $\mathcal{O}(\log(p+k))$ depth circuit to add these last two integers and produce the same sum.

Depth:

$k-1$ stages of constant depth plus $\mathcal{O}(\log(p+k))$

$\mathcal{O}(\log m) + \mathcal{O}(\log(p+\log n)) = \mathcal{O}(\log m + \log p) = \mathcal{O}(\log n)$

Size:

Stage 1:

the number of 4-2-CSA's is $\frac{m}{4}$

the number of bits is p

Stage 2:

the number of 4-2-CSA's is $\frac{m}{8}$

the number of bits is $p+2$

$\vdots$

Total size is $\leq p(\frac{m}{4}+\frac{m}{8}+\frac{m}{10}+...)+m(\frac{2}{8}+\frac{3}{16}+...) \leq \frac{1}{2}mp+m = \mathcal{O}(n)$ plus $\mathcal{O}(p+\log n) = \mathcal{O}(n)$ for the final addition.

Corollary 39 *Vector addition has circuits of size* $\mathcal{O}(n)$ *and depth* $\mathcal{O}(\log n)$.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 13 February 1992 **Topic: Symmetric functions**
Instructor: Sam Buss **Notes by: Goran Gogić**

Symmetric functions

Definition 28 *Let* $E(x_1, ..., x_n) = \begin{cases} 1 & \text{iff exactly } k \text{ of } x_1, ..., x_n \text{ are equal } 1 \\ 0 & \text{otherwise} \end{cases}$

Lemma 40 *There is a circuit of size* $\mathcal{O}(n)$ *and depth* $\mathcal{O}(\log n)$ *that computes* $E_0^n, ..., E_n^n$.

Proof.

Build a circuit of size $\mathcal{O}(n)$ and depth $\mathcal{O}(\log n)$ that computes bits $y_{\lceil \log n \rceil - 1}, ..., y_0$ coding the number of x_i (Vector Addition Circuit)

The E_k^n's are computed by all full minterms on $y_{\lceil \log n \rceil - 1}, ..., y_0$. This gives all the E_k^n's.

Q.E.D.

Theorem 41 *Let* $f : \{0,1\}^n \to \{0,1\}$ *be a symmetric function. Then,* f *has an* $\{\vee, \wedge, \neg\}$*–circuit of size* $\mathcal{O}(n)$ *and depth* $\mathcal{O}(\log n)$.

Proof.

Build a circuit for f by taking the circuit computing $E_0^n, ..., E_n^n$ and computes f as a disjunction of the apropriate subset of $E_0^1, ..., E_n^n$ — this is $\leq n - 1$ disjunction which can be "balanced" to have depth $\lceil \log n \rceil$.

Q.E.D.

Corollary 42 *Any symetric function is in* $\mathbf{NC}^1$.

Unary Sorting

Consider $f : (x_1, ..., x_n) \to (T_n^n, ..., T_1^n)$ where $T_k^n(x) = 1$ iff at least k of x_i's are equal 1.

HW#14 Show that there is a $\mathcal{O}(n)$ size and $\mathcal{O}(\log n)$ depth circuit computing all of $T_n^n, ..., T_1^n$.

HW#15 Find (and try to improve) the claimed circuit for the comp.theory problem.

Cumulative Counting

Consider $CC(x_1, x_2, ..., x_n) = (y_i^j)_{i=1,n}^{j=0,\lceil \log n \rceil - 1}$ such that $(y_i^{\log n - 1}, ..., y_i^1, y_i^0)$ codes in binary the number of 1's in the array $x_1, ..., x_i$.

HW#16 There are circuits of size $\mathcal{O}(n \log n)$ and depth $\mathcal{O}(\log n)$ that compute $CC(\vec{x})$.

Now, we return to the

Lower Bounds

Recall that $C_{B_2}(f) \geq n - 1$ if f essentially depends on n of its inputs.

Definition 29 *A leaf gate is a gate which has as inputs two literals.*

Definition 30 *A literal is a input x_i or the negation $\overline{x_i}$ of the input.*

Theorem 43 (Schnoor 1974) $C^*_{\{\vee,\wedge,\neg\}}(\mathbf{Parity}_n) = 3n - 3$

Proof. We already showed $C^*_{\{\vee,\wedge,\neg\}}(\mathbf{Parity}_n) \leq 3n-3$. We'll show that $C^*_{\{\vee,\wedge,\neg\}}(\mathbf{Parity}_n) \geq 3n - 3$ by induction on n.

Base case: $n = 1$ Obvious

Induction step: Let C be a circuit of minimal size computing $Parity_n, (n \geq 1)$.

Pick any leaf gate g_1 in C. Suppose g_1 is an $\vee$ - gate and has inputs x_i and x_j (cases where g_1 is an $\wedge$ gate and one or both are negated are similar).

Claim 44 *x_i or $\overline{x_i}$ is input to at least one other $\vee$ or $\wedge$ gate g_2 in C.*

Proof. Otherwise, when $x_j = 1$, then C's output is independent of x_i. This is imposible, $\neg C_{|x_j=1}$ computes parity on $(x_1, ..., x_{j-1}, x_{j+1}, ..., x_n)$ which is not independent of x_i.

Q.E.D.

Eliminating the gate means "bypassing" the gate. $(1 \vee \phi)$ is replaced by ϕ. $(0 \vee \phi)$ is replaced by ϕ.

Form $C_{|x_i=1}$ and remove all constants by taking $(0 \wedge \psi)$ to 0, $(1 \wedge \psi)$ to ψ e.t.c.

In $C_{|x_i=1}$, gate g_1 has been fixed to 1, gate $g_2$has been eliminated and at least one gate g_3 that has g_1 or $\overline{g_1}$ as input is eliminated. If $g_3 = g_2$ then g_2 has inputs $\pm x_i$ and $\pm g_1$. Since both its inputs are set to constants, g_2 is also set to constant, so any gate it feeds into will be eliminated. And g_1 and g_2 are not output gates, since they are $\vee$'s or $\wedge$'s of x_i or $\overline{x_i}$.

In any event, $C_{|x_i=1}$ has at least three fewer gates than C. The $*$-size must be $\geq 3n - 6$ by the induction hypothesis. Hence C has the size $\geq 3n - 6 + 3 = 3n - 3$.

Q.E.D.

Recall that B_2 is the set of all binary connectives.

Definition 31 *A function $f : \{0,1\}^n \to \{0,1\}$ is a $C_{2,3}$ function iff:*

(1) It depends essentially on all its inputs

(2) For all its inputs x_i there is a constant $C \in \{0,1\}$ such that $f_{|x=c}$ is in $C_{2,3}$

(3) For any inputs x_i, x_j, there are at least three subfunctions that can be induced from $\{x_1, ..., x_n\} - \{x_i, x_j\}$ by seting x_i and x_j to constants. In other words, $N_f(\{x_1, ..., x_n\} - \{x_i, x_j\}) \geq 3$

Example

Parity $\notin C_{2,3}$ because it does not satisfy conditions (2) and (3)
mod3 $\in C_{2,3}$

Theorem 45 (Schnorr '74) *If $f \in C_{2,3}$ where $f : \{0,1\}^n \to \{0,1\}$ then $C^*_{B_2}(f) \geq 2n - 3$.*

Proof.

$n = 1$: Trivial

$n = 2$: f depends on both inputs, so at least one two-input gate is needed.

Induction step; Assume C is a minimal circuit for computing f. Pick a "leaf gate" g that has literals $\pm x_i$, $\pm x_j$ where $i \neq j$.

Claim 46 *At least one of x_i, x_j is input (perhaps in negated form) to another gate g_2.*

Proof. Otherwise, the function f depends on x_i, x_j which feed only into $g(x_i, x_j)$. So, x_i, x_j could induce at most 2 subfunctions.

Q.E.D.

Suppose $\pm x_1$ feeds into g_1 and g_2. Set $x_1 = c$, $c \in \{0, 1\}$ such that $f_{|x_i=c}$ is a $C_{2,3}$ function. This eliminates at least 2 gates. So, the original circuit C have at least $2n - 3$ gates.

Q.E.D.

Remark 5: $C^*_{B_2}(f) = \min(C_{B_2}(f), C_{B_2}(\neg f))$

Not gates can be incorporated into input gates, constants also.

All B_2 functions:

- Constant functions $g = 0$, $g = 1$
- Projection functions x_1, x_2, $\neg\ x_1$, $\neg x_2$
- Parity type gates $\oplus$, $\Leftrightarrow$
- $\vee$ type gates $g(x_1, x_2) = \pm(\pm x_1 \vee \pm x_2)$: $\vee$, $\wedge$, NAND, NOR, IMPLIES, NOTIMPLIES, IMPLIEDBY, NOTIMPLIEDBY

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: February 18, 1992 **Topic: Schnorr/Blum lower bounds**
Instructor: Sam Buss **Notes by: Dave Robinson**

Definition 32 *A function f is a $C_{2,3}$-function iff*

(1) *f depends on all its inputs.*

(2) *If $n \geq 3$, then for any 2 inputs x_i, x_j there are ≥ 3 functions that can be induced by setting x_i, x_j to constants.*

(3) *For all x_i, there is a constant $c \in \{0,1\}$ such that $f|_{x_i=c}$ is in $C_{2,3}$.)*

Theorem 47 (Schnorr, 1974) *Let $f : \{0,1\}^n \to \{0,1\}$ be a $C_{2,3}$-function. Then*

$$C_{B_2}(f) \geq 2n - 3.$$

Recall that B_2 is the set of all 2-input gates.

Proof. By induction on n.
Base case: $n = 2$— it's obvious that $C_{B_2}(f) \geq 1$, since f depends on both of its inputs.
Induction step: Assume C is a minimal circuit computing f. Pick a leaf gate g that has literals $\pm x_i, \pm x_j$, $(i \neq j)$ as inputs.

Claim: At least one of x_i, x_j is input (perhaps negated) to another gate g_2.

Pf: otherwise the function f depends on x_i, x_j only via g. So x_i, x_j can induce at most 2 subfunctions on the rest of the variables (g will either be 1 or 0). Q.e.d. claim.

Suppose $\pm x_i$ feeds into g and g_i. Set $x_i = c$, $c \in \{0,1\}$, such that $f|_{x_i=c}$ is a $C_{2,3}$ function. (Such a c exists by part (3) of the definition above.) This eliminates ≥ 2 gates. By induction hypothesis the induced function has $\geq 2(n-1) - 3$ gates, so the original circuit had at least $2n - 3$ gates.

Q.E.D.

References for the next topic:
N. Blum "A boolean function requiring $3n$ network size," TCS (1984) 337-345.
Paul, SIAM J. Comput. (1977).
(Actually the lower bound is $3n - o(n)$.)

Definition: f will be a $(n + 3logn + 1)$-ary function.
$f(a_1, \ldots, a_{logn}, b_1, \ldots, b_{logn}, c_1, \ldots, c_{logn}, q, x_1, \ldots, x_n)$
The first three blocks will be used for addressing, and q will be the "control bit."
$|\vec{a}|$ = the integer with binary representation given by the a's. Similarly for the b's and c's.

$f(\vec{a}, \vec{b}, \vec{c}, q, \vec{x}) = (q \oplus x_{|\vec{a}|}) \wedge (x_{|\vec{b}|} \oplus x_{|\vec{c}|})$

Theorem 48 (N. Blum) $C_{B_2}(f) \geq 3n - 3$. *So if* N *= the number of inputs to* f*, then* $C_{B_2}(f) \geq 3N - o(N)$.

Proof. By induction. Fix n. Letting $s = 0, \ldots, n$ define the proposition P:
$P(s)$ = for all $S \subseteq \{x_0, \ldots, x_{n-1}\}$ such that $|S| = s$, if $f(\vec{a}, \vec{b}, \vec{c}, q, \vec{x}) = (q \oplus x_{|\vec{a}|}) \wedge (x_{|\vec{b}|} \oplus x_{|\vec{c}|})$ whenever $|\vec{a}|, |\vec{b}|, |\vec{c}| \in S$ then $C_{B_2}(f) \geq 3s - 3$.

Prove $P(s)$ by induction on s.
Base cases: $s = 0, 1$ are trivial since $3s - 3 \leq 0$.
In the case of $s = 2$, f depends on two of the x's (the ones in S), the control bit q, and on ≥ 1 address bits, making a total of ≥ 4 bits. This necessitates at the very least $4 - 1 = 3$ two-input gates. So $C_{B_2}(f) \geq 3 = 3(2) - 3$. (Similar arguments work for $s \leq 6$.)

Induction Step: $(s \geq 3)$ Assume C is a circuit of minimal size computing f.
First idea: set one x_i to a constant and kill 3 gates.

Case(1): some x_i feeds into ≥ 2 gates, one of which is an $\wedge$-type gate, $(i \in S)$.
Set $x_i = c$, $c \in \{0, 1\}$, so as to force the $\wedge$-type gate to a constant. This eliminates all gates x_i feeds into and all gates that this $\wedge$-type gate feeds into. This will eliminate ≥ 3 gates. (Similar to the parity argument.)

Case(2): some x_i feeds into a parity-type gate.

Claim: wlog, each parity gate that x_i feeds into has property G (G = good): Either g feeds into ≥ 2 gates or g feeds into an $\wedge$-type gate, or g is an output gate.

Pf: otherwise we have gates $g = g_1,\ g_2,\ g_3, \ldots, g_k$ such that each g_j is a parity-type gate, each g_j feeds into only g_{j+1} for $i = 1, \ldots, k - 1$ and g_{j+1} has property G.

Figure Goes Here

We let A_j denote the other input of g_j. Note that no A_j is equal to x_i, because the ciruit is of minimal size.

We can modify $g_1, \ldots, g_k$ by interchanging the input x_1 of gate g_1 and the input A_k of gate g_k:

Figure Goes Here

This makes g_k still compute the same function and has x_i feeding into a gate with property G. Repeat this with all the $\oplus$-type gates x_i feeds into. This makes the claim true (since none of the A_i's equaled x_i or $\neg x_i$.)

Since a circuit C has no cycles, one of the $\oplus$ that x_i feeds into does not depend on any of the other $\oplus$-gates x_i feeds into. So there is a gate A that feeds into one of the $\oplus$-gates that x_i feeds into that does not depend on x_i.

Now setting $x_i = A$ or $\neg A$ (x_i becomes a function of the rest of the variables) will force g to be constant 0 or 1 (resp.). If g feeds into ≥ 2 gates, either choice eliminates ≥ 3 gates. If g feeds into an $\wedge$-type gate, one of the choices eliminates g, the $\wedge$-type gate, and all the gates that that gate feeds into. (If g or an $\wedge$-type gate it feeds into is the output, this eliminates the whole circuit — in fact, they aren't though.) In any event ≥ 3 gates are eliminated. As before, use the induction hypothesis with S now equal to $S - \{i\}$.

Cases (3) and (4): every x_i feeds into exactly one $\wedge$-type gate.

Def: Let G_i be the $\wedge$-type gate that x_i feeds into. Let $G = \{G_i : i \in S\}$ so $|G| \leq s$.

Case (3): Some G_i feeds into ≥ 2 gates. Set $x_i = c$ such that G_i is forced to a constant. This eliminates G_i and the gates it feeds into, i.e. ≥ 3 gates, reducing to the induction hypothesis with $S - \{i\}$.

Case (4): each g_i feeds into exactly one gate Q_i. Let $Q = \{Q_i : i \in S\}$.

Lemma A: If $i \neq j (\in S)$ then $G_i \neq G_j$.

Pf: otherwise we can set $x_i = c \in \{0, 1\}$ which forces $G_i = G_j$ to a constant. This makes the circuit independent of x_j. But

$f([j],[j],[i],c,x_0 \ldots x_{i-1},c,x_{i+1},\ldots x_{n-1}) = (c \oplus x_j) \wedge (x_j \oplus c) = c \oplus x_j = \pm x_j$.
Where $[j]$ is the bit string that is the binary representation of j. So f depends on x_j still — #contradiction#. Q.e.d. lemma A.

So $|G| = S$.

Def: if u, v are gates in C, a path from u to v is a sequence of gates, starting at u, ending with v such that each gate feeds into the next gate in the sequence. Such a path is denoted $u \Rightarrow v$, (non-uniquely). The output node of C is denoted t. A path $u \Rightarrow v$ is free if none of its internal nodes are in G. If $u_1 \Rightarrow v_1$ and $u_2 \Rightarrow v_2$ are paths then their collector is the first node in common to the two paths (if any).

Lemma B: For all $i \in S$, there is a free path $G_i \Rightarrow t$.

Pf: Fix i, for each $j \in S, j \neq i$ choose $c_j \in \{0,1\}$ so that $x_j = c_j$ forces G_j to a constant. Make the assignments $x_j = c_j$. If there is no free path $G_i \Rightarrow t$ this makes the circuit's value independent of x_i.

However, for any $j \neq i$,
$f([i],[i],[j],c_j,c_0,\ldots c_{i-1},x_i,c_{i+1},\ldots c_{n-1}) = (c_j \oplus x_i) \wedge (x_i \oplus c_j)$

So f is not independent of x_i.
Q.e.d. lemma B

Corollary C: $G \cap Q = \phi$.

Pf: Suppose $Q_i = G_j$. Then any path $G_i \Rightarrow t$ must contain $Q_i = G_j$ and is not free.

Q.E.D.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: February 20, 1992 **Topic: Blum lower bound theorem**
Instructor: Sam Buss **Notes by: Dave Robinson**

— continuation of proof of Blum theorem —

Def: A gate g is a split if g feeds into ≥ 2 gates. A gate g is a free split if for at least 2 gates $g_1,\ g_2$ that g feeds into, there exist free paths $g_1 \Rightarrow t,\ g_2 \Rightarrow t$.

Lemma D: Let g be the collector of two free paths $G_i \Rightarrow t$, $G_j \Rightarrow t$, $(i \neq j,\ i,j \in S)$. Then at least one of the following holds:

(i) There is a free split $\neq g$ on the path $G_i \Rightarrow g$.

(ii) There is a free split $\neq g$ on the path $G_j \Rightarrow g$.

(iii) g is a $\oplus$-type gate and either there exists a free path $G_i \Rightarrow G_j$ or a free path $G_j \Rightarrow G_i$.

Main idea: if (i) and (ii) fail then "all the free information" about x_i and x_j has to flow through g.

Pf: suppose (i) and (ii) fail. Fix assignments $x_k := c_k$ for all $k \in S - \{i,j\}$ which force G_k to a constant. This essentially "chokes off" any path through a G_k with $k \neq i,j$.

Case (a): Suppose g is a $\oplus$-type gate. Set rest of f's inputs other than x_i, x_j to constants so that f computes $x_i \wedge (\pm x_j)$, i.e.,
$f([i],[j],[k],0,\vec{c},x_i,\vec{c},x_j,\vec{c}) = (0 \oplus x_i) \wedge (x_j \oplus c_k),\ \ (k \neq i,j)$
$= x_i \wedge \pm x_j$

So now assume there is no free path $G_i \Rightarrow G_j$ and no free path $G_j \Rightarrow G_i$,
Then claim: one of g's inputs depends only on x_i and the other only on x_j.
Reason: if there is a path from x_i to the other input of g then either

(1) it's free so we get a split, or

(2) it goes through G_j so we have a free path $G_i \Rightarrow G_j$, or

(3) it goes to some G_l, $l \neq i,j$ in which case the path is "choked off."

So g has one input which is x_i, $\neg x_i$, 0 or 1, and the other input is x_j, $\neg x_j$, 0 or 1. So g computes $\pm(x_i \oplus x_j)$, $\pm x_i$, $\pm x_j$, 0, or 1. Hence the output t also computes one of these functions. But this contradicts the claim that t computes $x_i \wedge (\pm x_j)$.

Case (b): g is $\wedge$-type.

Set the rest of the inputs other than x_i, x_j to constants so that f computes $x_i \oplus x_j$. For example,
$f([k],[i],[j],1-c_k,\vec{c},x_j,\vec{c},x_j,\vec{c}) = (\neg c_k \oplus c_k) \wedge (x_i \oplus x_j), \quad k \neq i,j$

Suppose for the sake of contradiction that G_j's input other than x_j depends on x_i. Then we can choose a value c_i such that setting $x_i = c_i$ forces G_j to a constant (G_j is an $\wedge$-type gate). This makes the circuit independent of x_j which contradicts the fact that it computes $c_i \oplus x_j$. Likewise G_i's other input doesn't depend on x_j. So by reasoning exactly as before, g has one input which is x_i, $\neg x_i$, 0, or 1, and the other x_j, $\neg x_j$, 0, or 1. So g computes $\pm(\pm x_i \wedge \pm x_j)$, $\pm x_i$, $\pm x_j$, 0, or 1. Thus t computes one of these, contradicting the fact that it should compute $x_i \oplus x_j$.
Q.e.d. Lemma D

Corollary E: The Q_i's are distinct.

Pf: If $Q_i = Q_j$ then $g = Q_i = Q_j$ violates D.

Lemma F: There are $\geq s-1$ distinct splits.

Pf: Iterate the following process. Start with $S' := S$.

Pick i,j with $G_i \Rightarrow t$ and $G_j \Rightarrow t$ and with collector g such that g is not on any free path $G_l \Rightarrow t$ where $l \neq i,j$ (l,i,j range over S').
Choose the first option that applies:

(a) If $G_i \Rightarrow g$ has a free split, choose it and set $S' := S' - \{i\}$.

(b) If $G_j \Rightarrow g$ has a free split, choose it and set $S' := S' - \{j\}$

(c) otherwise by lemma D there is a free path $G_i \Rightarrow G_j$ or a free path $G_j \Rightarrow G_i$ and so either $G_i \Rightarrow g$ has a split or $G_j \Rightarrow g$ has a split (resp.). Choose that split and set $S' := S' - \{j\}$ or $S' := S' - \{i\}$ resp.

Iterate this process until $|S'| = 1$.

Claim: all the free splits chosen under under (a) and (b) are distinct.

Pf: By choice of i,j,g.

Lemma H: Option (c) is picked at most once.

By lemma H all $s-1$ splits picked are distinct. In fact, we have $\geq s-2$ distinct free splits.

Lemma G: Suppose option c is taken for some i,j in the process with a split being found on $G_i \Rightarrow G_j$. Then for all $v \in S - \{j\}$ there exists a free path from $G_v \Rightarrow G_j$ or $G_v \Rightarrow G_i$.

Pf: If $v = i$ —true by assumption. If $v \in S - \{i,j\}$ (Assume no free path $G_v \Rightarrow G_j$ or $G_v \Rightarrow G_i$ exists) fix assignments $x_k := C_k \in \{0,1\}$ ($k \neq i,j,v$) such that each G_k is forced to a constant and set rest of f's inputs to constants so that f computes $x_j \wedge (x_i \oplus x_v)$.
Cases:

(i) G_j's other input depends on x_i.
Then set $x_i = c_i$ so as to force G_j to a constant. This makes the circuit independent of x_j, contradicting the fact that it should compute $x_j \wedge (c_i \oplus x_v)$
(ii) otherwise G_j is independent of x_i then set $x_v = c_v$ so as to force G_v to a constant. So the circuit computes $x_j \wedge (x_i \oplus c_v)$. However, when option (c) is picked, the situation of (iii) of Lemma D holds. So g must be a $\oplus$-type gate and as argued in the proof of Lemma D circuit outputs $\pm(x_i \oplus x_j)$ or $\pm x_i$ or $\pm x_j$ or 0 or 1 (since G_j doesn't depend on x_i), which contradicts the fact that it is supposed to compute $x_j \wedge (x_i \oplus c_v)$.
Q.e.d. Lemma G.

Proof of Lemma H: Suppose option (c) is chosen twice with values i, j (first) and i', j' (second). Since j was discarded the first time option (c) was used, $j \neq j'$. Lemma G (used twice) implies there is a path $G_j \Rightarrow G_{j'}$ and a path $G_{j'} \Rightarrow G_j$. Impossible: this is a cycle.
Q.e.d. Lemmas H and F.

Proof of the Theorem: Trying to find $3s - 3$ gates.
Found so far: G — s gates
Q — s gates.
Problem: the $s-2$ free splits may not be distinct from the Q's. Let $m = \#\, Q$'s that are not one of these free splits. So $m \geq 2$ (note: this differs from the Blum proof). The $s-2$ free splits give $2s-4$ beginnings of free paths. Plus we have m many beginnings of free paths from the nodes in Q that are not free splits. So the total number of beginnings of free paths is $2s + m - 4$. These paths all have to be "collected" (not in the technical sense) before they reach t. How can they be collected?
(a) s can feed into the Q's that are not free splits (actually it's $s-1$)
(b) there are $(s-2) - (s-m)$ free splits not in Q. They can collect two paths each— i.e., $2m-4$ free paths can enter them.
(c) there are still $\geq (2s + m - 4) - s - (2m - 4) = s - m$ beginnings left. They can't be collected by G, Q or the free splits, so we need $s - m - 1$ more gates to collect them.

Total # of gates found in (a), (b), (c): $s + (m-2) + (s-m-1) = 2s - 3$.

Adding in the gates in G: $s + 2s - 3 = 3s - 3$.

Q.E.D.

Additional notes:

In fact $3s - 2$ is achievable because of the comment in (a).

The best known upper bound is around $6n$.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 25 Febuary 1992 **Topic: Subbotovskaya's theorem**
Instructor: Sam Buss **Notes by: Jinghui Yang**

We will prove the results of Subbotovskaya and Andreev on lower bounds of $\{\neg, \vee, \wedge\}$ formula size. The development of this approach to show lower bounds is roughly as follows:

- Subbotovskaya(1961)
- Andreev (1986)
- Nissan-Impagliazzo (~ 1992)
- Paterson-Zwick ($FOCS'91, \approx n^{2.63}$ lower bounds)

Miniproject: understand Paterson-Zwick's paper.

Convention: By $\{\neg, \vee, \wedge\}$ formulas we mean those whose negation has been pushed to leaves.

Proposition 49 *The formula $x \wedge \psi$ is equivalent to $x \wedge \psi_{|x=1}$. Similarly for $(\neg x) \wedge \psi$, $x \vee \psi$ and $(\neg x) \vee \psi$*

Proof. Obvious.

Q.E.D.

Convention: Whenever a formula has $(\pm x_i) \wedge \psi$ or $(\pm x_i) \wedge \psi$ as a subformula, then x_i does not occur in ψ.

Notation:

$$|\varphi| = \textit{the leaf-size of } \varphi$$

$$|0| = |1| = 0,$$

$$|\varphi|_0 = \begin{cases} |\varphi| & \text{if } |\varphi| > 0 \\ -\frac{1}{2} & \text{if } |\varphi| = 0 \end{cases}$$

Theorem 50 *(Subbotovskaya) Let φ be an $\{\neg, \vee, \wedge\}$-formula with variables from $\{x_1, x_2, \ldots, x_n\}$. $1 < k < n$. Let σ be a partial truth assigment randomly selected as follows:*

a. *Pick subset of variables of $\{x_1, x_2, \ldots x_n\}$ of size $n - k$ randomly, and let the domain of σ, $Dom(\sigma)$, be this set.*
b. *For each $x_i \in dom(\sigma)$, set $\sigma(x_i) = 0$ or 1 with equal (independent) probabilities.*

Then

$$E(|\varphi_{|\sigma}|) \leq C(\frac{k}{n})^{\frac{3}{2}}|\varphi| + \frac{1}{2}$$

for some constant C. (In fact C is close to 1 for large n).

($\varphi_{|\sigma}$ is the formula ϕ with the values of variables assigned according to σ, eliminating $\wedge$, $\vee$ gates with constant inputs.)

Lemma 51 *Let φ be as in Subbotovskaya's theorem. Assume $|\varphi| \neq 1$. Pick a partial assignment σ with $|dom(\sigma)| = 1$ by choosing a random x_i and setting $\sigma(x_i) = 0$ or 1 with eaqual probabilities.*

Then

$$E(|\varphi_{|\sigma}|) \leq (1 - \frac{1.5}{n})|\varphi|.$$

$\frac{1.5}{n}$ is a little surprising, since one might have expected $\frac{1}{n}$ instead.

Proof. The proof is by induction on $|\varphi|$. It is obviously true for $|\varphi| = 0$.
Base case (1): $|\varphi| = 2$, i.e. φ is $(\pm x_i) \wedge (\pm x_j)$ or $(\pm x_i) \vee (\pm x_j)$

Let us just do the subcase that φ is $x_i \wedge x_j$. The probability with which x_i or x_j is chosen is $\frac{2}{n}$.

with further probability $\frac{1}{2}$: $\sigma(x_i)$ or $\sigma(x_j) = 1$,
with further probability $\frac{1}{2}$: $\sigma(x_i)$ or $\sigma(x_j) = 0$

and thus

with probability $\frac{1}{n}$, $|\varphi_{|\sigma}| = 1$,
with probability $\frac{1}{n}$, $|\varphi_{|\sigma}| = 0$,
with probability $\frac{n-2}{n}$, $\sigma(x_i)$ and $\sigma(x_j)$ are undefined. Hence $|\varphi_{|\sigma}| = |\varphi| = 2$.
So

$$\begin{aligned} E(|\varphi_{|\sigma}|) &= \frac{1}{n} \cdot + \frac{1}{n} \cdot 0 + \frac{n-2}{n} \cdot 2 \\ &= \frac{1}{n} + (1 - \frac{2}{n}) \cdot 2 \\ &= (1 - \frac{2}{n} + \frac{1}{2n}) \cdot 2 \\ &= (1 - \frac{1.5}{n}) \cdot 2 \\ &= (1 - \frac{1.5}{n})|\varphi| \end{aligned}$$

Induction step 1:
Assume that the top gate of φ has one input being a literal, e.g. φ is of the form $x_i \wedge \psi$ with $|\psi| \geq 1$. Then
with probability $\frac{1}{n} \cdot \frac{1}{2}$, $\sigma(x_i) = 0$ and $|\varphi_{|\sigma}| = 0$,
with probability $\frac{1}{n} \cdot \frac{1}{2}$, $\sigma(x_i) = 1$ and $|\varphi_{|\sigma}| = |\psi|$,
with probability $\frac{n-1}{n}$, $\sigma(x_i)$ is undefined.

Since x_i is not in ψ, $E(|\psi_{|\sigma}| : \sigma(x_i)\ undefined) \leq (1 - \frac{1.5}{n-1})|\psi|$ by the induction hypothesis.

So,

$$\begin{aligned}
E(|\varphi_{|\sigma}|) &\leq \frac{1}{2n}\cdot 0 + \frac{1}{2n}\cdot|\psi| + \frac{n-1}{n}\cdot[(1-\frac{1.5}{n-1})|\psi|+1] \\
&= \frac{1}{2n}|\psi| + (\frac{n-1}{n} - \frac{1.5}{n})|\psi| + 1 - \frac{1}{n} \\
&= (1 - \frac{1}{n} - \frac{1.5}{n} + \frac{1}{2n})|\psi| + 1 - \frac{1}{n} \\
&= (1-\frac{1.5}{n})|\psi| + 1 - \frac{1}{n} - \frac{1}{2n}|\psi| \\
&\leq (1-\frac{1.5}{n})|\psi| + 1 - \frac{1}{n} - \frac{1}{2n} \\
&\leq (1-\frac{1.5}{n})|\psi| + (1-\frac{1.5}{n}) \\
&= (1-\frac{1.5}{n})|\varphi|
\end{aligned}$$

Induction step 2: φ is $\psi \wedge \chi$ or $\psi \vee \chi$ with $|\psi|$ and $|\chi| > 1$. Then $|\varphi| = |\psi| + |\chi|$ hence $|\varphi_{|\sigma}| \leq |\psi_{|\sigma}| + |\chi_{|\sigma}|$.

$$\begin{aligned}
E(|\varphi_{|\sigma}|) &\leq E(|\psi_{|\sigma}|) + E(\chi_{|\sigma}) \\
&\leq (1-\frac{1.5}{n})|\psi| + (1-\frac{1.5}{n})|\chi| \\
&= (1-\frac{1.5}{n})|\varphi|
\end{aligned}$$

Q.E.D.

Remark: In Subbotovskaya's Theorem, the lower bound 1.5 cannot be improved past 2 because of "parity". The best result so far is 1.63.

Lemma 52 *Let φ be as in Subbotovskaya's theorem, where φ can be any formula. Choose σ randomly as in Lemma 51. Then*

$$E(|\varphi_{|\sigma}|_0) \leq (1-\frac{1.5}{n})|\varphi|_0$$

Proof. Case 1: $|\varphi| = 0$, then $|\varphi|_0 = -\frac{1}{2}$ and $E(|\varphi_{|\sigma}|_0) = -\frac{1}{2}$, Lemma 52 follows since $1 - \frac{1.5}{n} < 1$.
Case 2: $|\varphi| > 1$,

$$\begin{aligned} E(|\varphi_{|\sigma}|_0) &\leq E(|\varphi_{|\sigma}|) \\ &\leq (1 - \frac{1.5}{n})|\varphi| \quad (\textit{by Lemma } 51) \\ &= (1 - \frac{1.5}{n})|\varphi|_0 \end{aligned}$$

Case 3: $|\varphi| = 1 = |\varphi|_0$, then φ is a literal, say x_i. Then with probability $\frac{1}{n}$, $\sigma(x_i)$ is defined and $|\varphi_{|\sigma}|_0 = -\frac{1}{2}$, otherwise, with probability $\frac{n-1}{n}$, $|\varphi_{|\sigma}| = 1$. So

$$\begin{aligned} E(|\varphi_{|\sigma}|_0) &= \frac{1}{n} \cdot (-\frac{1}{2}) + \frac{n-1}{n} \cdot 1 \\ &= 1 - \frac{1}{n} - \frac{1}{2n} \\ &= 1 - \frac{1.5}{n} \end{aligned}$$

Q.E.D.

Now we prove Subbotovskaya's theorem.

Proof. (For Subbotovskaya's theorem) Assuming the hypothesis of the theorem. By Lemma 52,

$$E(|\varphi_{|\sigma}|_0) \leq (1 - \frac{1.5}{n})(1 - \frac{1.5}{n-1}) \ldots (1 - \frac{1.5}{k+1})|\varphi|_0$$

It suffices to show

$$r = (1 - \frac{1.5}{n}) \ldots (1 - \frac{1.5}{k+1}) \leq C(\frac{k}{n})^{\frac{3}{2}}$$

for some constant C. Now,

$$\ln r = \ln(1 - \frac{1.5}{n}) + \ln(1 - \frac{1.5}{n-1}) + \ldots + \ln(1 - \frac{1.5}{k+1}),$$

$$\ln(1 - x) < -x, \text{ for } 0 < x < 1.$$

So,

$$\begin{aligned}
\ln r &< -\frac{1.5}{n} - \frac{1.5}{n-1} - \frac{1.5}{n-2} \cdots - \frac{1.5}{k+1} \\
&< -1.5 \int_{k+1}^{n+1} \frac{1}{x} \\
&= -1.5 \ln x|_{k+1}^{n+1} \\
&= -1.5(\ln(n+1) - \ln(k+1))
\end{aligned}$$

and

$$\begin{aligned}
r &< e^{-1.5 \ln \frac{n+1}{k+1}} \\
&= (\frac{n+1}{k+1})^{-1.5} \\
&= (\frac{k+1}{n+1})^{1.5} \\
&\leq C(\frac{k}{n})^{1.5}
\end{aligned}$$

where C is some constant.

Q.E.D.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 27 Febuary 1992 **Topic: Andreev's lower bound on formula size**
Instructor: Sam Buss **Notes by: Jinghui Yang**

We'll prove Andreev's theorem and Riordan-Shannan's theorem. If we take a function h of the form

$$h(\vec{x}) = g(x_{1,1} \oplus x_{1,2} \oplus \ldots \oplus x_{1,n}, x_{2,1} \oplus x_{2,2} \oplus \ldots x_{2,n}, \ldots, x_{\log n,1} \oplus x_{\log n,2} \ldots, \oplus x_{\log n,n})$$

where $g(z_1, \ldots, z_{\log n})$ will be known to have $\{\neg, \vee, \wedge\}$-fomula size $\approx \frac{n}{\log\log n}$. Then an obvious formula for h would have size $= (size\ of\ g) \cdot (size\ of\ parity_n) \approx \frac{n^3}{\log\log n}$. Actually we shall get the size to be $\approx n^{\frac{5}{2}}$ by working to get some g explicitly defined.

Lemma 53 *Let $s = \log n$, $x_{1,1}, \ldots, x_{1,n}, x_{2,1}, \ldots, x_{2,n}, \ldots, x_{s,1}, \ldots, x_{s,n}$ be variables, and $\{x_{i,j}\}_j$ with i fixed is called a "block of variables". Let $k = (\log n)(\log\log n)$, choose σ randomly as in Subbotovskaya's theorem to assign $\{0,1\}$ values to all but k variables. Then with probability $> \frac{1}{2}$, σ will not map any entire block of variables to constants.*

Proof. By $\sigma(x_{i,j}) \downarrow$ we will mean $\sigma(x_{i,j})$ is defined.
Fix i,

$$\begin{aligned}
Pr[(dom(\sigma) \supset i^{th}\ block] &= Pr[\sigma(x_{i,1}) \downarrow] \cdot Pr[\sigma(x_i, 2) \downarrow | \sigma(x_{i,1}) \downarrow] \cdot \cdots \\
&\quad \cdot Pr[\sigma(x_{i,n}) \downarrow | \sigma(x_{i,1}) \downarrow, \sigma(x_{i,2}) \downarrow, \ldots, \sigma(x_{i,n-1}) \downarrow] \\
&< \prod_{j=1}^{n} Pr[x_{i,j} is\ in\ domain\ of\ \sigma] \\
&= \prod_{j=1}^{n} \frac{n\log n - (\log n)(\log\log n)}{n\log n} \\
&= \prod_{j=1}^{n} (1 - \frac{\log\log n}{n})
\end{aligned}$$

$$
\begin{aligned}
&= (1 - \frac{\log\log n}{n})^n \\
&\leq e^{-\log\log n} \\
&< \frac{1}{2\log n}
\end{aligned}
$$

Then,

$$
\begin{aligned}
Pr[\textit{some block is contained in domain of } \sigma] &\leq \sum_{i=1}^{\log n} Pr[\textit{the } i^{th} \textit{ block is in domain of } \sigma] \\
&< (\log n)(\frac{1}{2\log n}) \\
&= \frac{1}{2}
\end{aligned}
$$

Q.E.D.

Lemma 54 *Suppose $\varphi = \varphi(x_{1,1}, \ldots, x_{s,n})$ is an $\{\neg, \vee, \wedge\}$ -formula. Let $\vec{x}$ be as in Lemma 53. Then there is at least one σ such that*

a. σ sets all but $(\log n)(\log\log n)$ of the $x_{i,j}$'s to $\{0,1\}$.

b. the domain of σ does not contain any entire block.

c. $|\varphi_{|\sigma}| \leq C'(\frac{\log\log n}{n})^{\frac{3}{2}}|\varphi| + 1$ for some constant C'.

Proof. By Subbotovskaya's theorem, $E(|\varphi_{|\sigma}|) \leq C(\frac{\log\log n}{n})^{\frac{3}{2}}|\varphi| + \frac{1}{2}$. Since $|\varphi_{|\sigma}| \geq 0$, for all σ, it follows that for at least $\frac{1}{2}$ of the possible σ's, $|\varphi_{|\sigma}| \leq 2C(\frac{\log\log n}{n})^{\frac{3}{2}}|\varphi| + 1$ also. By Lemma 53, more than $\frac{1}{2}$ of the σ's satisfy b. Taking $C' = 2C$, at least one σ satisfies the condition.

Q.E.D.

For the proof of the next theorems, we need to define a new function.

Definition 33 *Let* $SA(y_0, y_1, \ldots, y_{n-1}, z_1, z_2, \ldots, z_{\log n}) = y_{|\vec{z}|}$ *where* $|\vec{z}|$ *is the integer with binary representation given by* $\vec{z}$*. Then define*

$$\lambda(y_0, y_1, \ldots, y_{n-1}, x_{1,1}, \ldots, x_{1,n}, \ldots, x_{s,1}, \ldots, x_{s,n}) = SA(y_0, \ldots, y_{n-1}, \bigoplus_j x_{1,j}, \bigoplus_j x_{2,j}, \ldots, \bigoplus_j x_{s,j}).$$

Theorem 55 *(Andreev, 1986)*

$$L^*_{\{\neg, \vee, \wedge\}}(\lambda) \geq d \cdot \frac{N^{\frac{5}{2}}}{(\log N)^{\frac{5}{2}} (\log \log N)^{\frac{5}{2}}}$$

where $N = n(\log n + 1)$ *is the number of inputs to* λ *and* d *is a constant.*

Theorem 56 *(Riordan-Shannan) Almost all* f*:* $\{0,1\}^n \to \{0,1\}$ *have*

$$L^*_{\{\neg, \vee, \wedge\}}(f) \geq (1 - \epsilon) \cdot \frac{2^n}{\log n} \quad (for\ any\ fixed\ \epsilon)$$

for sufficiently large n*.*

Andreev's theorem can be derived from Riordan and Shannan's theorem.

Proof. (of Andreev's theorem) From Riordan and Shannan's theorem, we know that there exist constants $c_0, \ldots, c_{n-1}$, such that if

$$f(z_1, \ldots, z_{\ logn}) = SA(c_0, \ldots, c_{n-1}, z_1, \ldots, z_{\log n})$$

then

$$L^*_{\{\neg, \vee, \wedge\}}(f) \geq \frac{1}{2} \cdot \frac{n}{\log \log n}. \quad (for\ large\ n)$$

Let φ_f be an $\{\neg, \vee, \wedge\}$-formula which computes $f(\bigoplus_j x_{1,j}, \ldots, \bigoplus_j x_{s,j})$. Choosing a σ satisfying Lemma 54, we have

$$|\varphi_{f|\sigma}| \leq C''(\frac{\log \log n}{n})^{\frac{3}{2}} |\varphi_f| + 1$$

and

$$\begin{aligned} |\varphi_f| &\geq \frac{1}{C''}(\frac{n}{\log \log n})^{\frac{3}{2}}(|\varphi_f|_\sigma| - 1) \\ &= \frac{1}{C''}(\frac{n}{\log \log n})^{\frac{3}{2}} L^*_{\{\neg, \vee, \wedge\}}(\varphi_f|_\sigma) \end{aligned}$$

Since σ does not set any entire block to a constant,

$$L^*_{\{\neg,\vee,\wedge\}}(\varphi_{f|\sigma}) \geq L^*_{\{\neg,\vee,\wedge\}}(f).$$

Thus,

$$\begin{aligned} |\varphi| &\geq d \cdot (\frac{n}{\log\log n})^{\frac{3}{2}} \frac{n}{\log\log n} \\ &= d \cdot \frac{n^{\frac{5}{2}}}{(\log\log n)^{\frac{5}{2}}} \\ &\geq d \cdot \frac{N^{\frac{5}{2}}}{(\log\log N)^{\frac{5}{2}}(\log N)^{\frac{5}{2}}} \end{aligned}$$

where $d = \frac{1}{2C''}, N = n\log(n+1)$.
Finally, $|\lambda| \geq |\varphi|$, $L^*(\lambda) + 1 \geq |\varphi|$ and Andreev's theorem follows.

Q.E.D.

Now we prove Riordan and Shannan's theorem.

Proof. (of Riordan and Shannan's theorem) We will actually prove $L_{B_2}(f) \geq (1-\epsilon)\frac{2^n}{\log n}$ which is stronger. If $L_{B_2}(f) \leq s$, then there is a B_2-formula of size s which computes f, so it suffices to count the number of functions which can be computed by formulas of size exactly s on inputs $x_1, x_2, \ldots, x_n$. Let $F(s,n)$ denote this number, then $F(s,n)$ can be bounded as follows (count the number of Reverse Polish notation formula):

$$F(s,n) \leq \left(\begin{array}{c} 2s+1 \\ s \end{array} \right) \cdot 10^s \cdot n^{s+1}$$

where in the righthand side of the above formula, the first term is the number of ways of placing s gates in between $s+1$ inputs, the second term is the number of choices of gates, and the third term is the number of choices of inputs.

So, $F(s,n) \leq 2^{2s+1} \cdot 10^s \cdot n^{s+1} \leq C^s \cdot n^{s+1}$ for some constant C. Hence $\frac{F(s,n)}{n} \leq (Cn)^s$.

Thus if $s < (1-\epsilon)\frac{2^n}{\log n}$, then

$$\begin{aligned}
\log \frac{F(s,n)}{n} &\leq s\log(Cn) \\
&< (1-\epsilon)\cdot\frac{2^n}{\log n}(\log C + \log n) \\
&= (1-\epsilon)\cdot 2^n \cdot (1+\frac{\log C}{\log n}) \\
&< (1-\frac{\epsilon}{2})\cdot 2^n \ \ (for\ large\ n)
\end{aligned}$$

And we get

$$F(s,n) \leq n\cdot 2^{(1-\frac{\epsilon}{2})\cdot 2^n} = 2^{2^n}(\frac{n}{(2^{2^n})^{\frac{\epsilon}{2}}}) \ll 2^{2^n}$$

for n large enough.

Q.E.D.

Miniproject: Read Luponov's upper bound of $(2+\epsilon)\frac{2^n}{\log n}$ on $\{\neg, \vee, \wedge\}$ formula size. Try to improve it to $(1+\epsilon)\frac{2^n}{\log n}$.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: March 3 and 5, 1992 **Topic: Hastad's Lemma**
Instructor: Sam Buss **Notes by: Elias Koutsoupias**

The following lectures are about lower bounds for constant depth $\{\wedge, \vee, \neg\}$-circuits with unbounded fanin. We are going to employ some conventions:

- Negations are pushed down with De Morgan's Laws. Inputs are literals $x_i, \overline{x}_i$. This at most doubles the size of the circuit.
- ANDs and ORs alternate. So, no OR (AND) is an input to an OR (AND) gate.
- ANDs and ORs are layered, so every gate at a given depth is of the same type.
- Gates at depth d have all their inputs from depth $d+1$. This may increase the size of a depth d circuit by a factor of d and its depth by at most one.
- An AND gate with 0 inputs has truth value 1 (true).
- An OR gate with 0 inputs has truth value 0 (false).

Notice that the fanout of a gate may be greater than one. If the fanout were required to be one, i.e. to have a formula instead of a circuit, this would increase the circuit from size s and depth d to a formula of size at most s^{d-1} and depth d. In other words, poly-size constant-depth circuits are the same with poly-size constant-depth formulae. But we are going to consider circuits, not formulae.

Our interest for constant depth circuits comes mainly from the fact that a constant-depth circuit can be thought of as a generalization of a CNF and DNF formula. In the following lectures, we are going to show that Parity does not have a poly-size constant-depth circuit. The intuition behind the proof is the following: Consider a circuit that computes Parity and assume that the bottom level consists of OR gates (the case of AND gates is treated similarly). If we can replace the bottom two levels (ANDs of ORs) with ORs of ANDs, without increasing significantly the size of the circuit and the fanin at the bottom level, then we can collapse levels 2 and 3, since they will consist of OR gates. If we repeat this process, then we will get

a circuit of depth 2 with small fanin at the bottom level. Since we will show that this is not possible, we can conclude that the size of the initial circuit was large.

The problem is that in general, we cannot replace an AND of ORs with OR of ANDs without dramatically increasing the fanin. But if we are willing to sacrifice a small set of variables, i.e. fix the value of some variables, them we can do it. A combinatorial proof of this fact seems extremely hard. Instead, the probabilistic method can be used, i.e. it can be shown that the probability of this event is nonzero. Here we are going to prove this result, known as Hastad's Lemma.

We will need some definitions first:

Definition 34 *Fix n, the number of variables. Let $0 < p < 1$. R_p is the probability space of restrictions ρ, i.e. ρ is a partial truth assignment to $x_1, x_2, \ldots, x_n$, chosen so that:*

$$\rho(x_i) = \begin{cases} \star & \textit{with probability } p \\ 0 & \textit{with probability } \frac{1-p}{2} \\ 1 & \textit{with probability } \frac{1-p}{2} \end{cases}$$

The values of $\rho(x_i)$, $i = 1, 2, \ldots, n$, are chosen independently and the value $\star$ means undefined.

The following fact will be useful later, when we shall prove that Parity does not have a poly-size constant-depth circuit.

Fact 1 *$Parity_n(x_1, x_2, \ldots, x_n)\mid_\rho$ is equal to a parity function or the negation of a parity function on the variables left undefined (value $\star$) by ρ.*

Definition 35 *A minterm of f is a partial truth assignment σ with $f\mid_\sigma \equiv 1$ and there is no other partial truth assignment $\sigma' \subset \sigma$ such that $f\mid_{\sigma'} \equiv 1$. The size of the minterm is the number of variables for which σ is defined.*

Example 6 The only minterm of $x \wedge y$ is $\sigma(x) = \sigma(y) = 1$. We may say that the minterm is just xy, that is we use conjunctions of literals to represent minterms. Also, $x \oplus y$ has minterms $x\overline{y}$ and $\overline{x}y$ and Majority(x, y, z) has minterms xy, yz and zx.

□

The following fact is obvious but it will be useful later.

Fact 2 *Parity($x_1, x_2, \ldots, x_n$) has 2^{n-1} minterms, each of size n.*

Since every function can be written as the disjunction of its minterms we have that:

Proposition 57 *$f(x_1, x_2, \ldots, x_n)$ can be written as an OR of ANDs each of fanin $\leq s$, if every minterm of f has size $\leq s$.*

Now we are ready to state the Hastad's Switching Lemma:

Lemma 58 (Hastad's First Switching Lemma) *Let G be an AND of ORs, with each OR of fanin $\leq t$. Let $0 < p < 1$. Let ρ be randomly chosen from the probabilistic space R_p. Then*

$$Pr[G|_\rho \text{ can } \textit{not} \text{ be written as an OR of ANDs each of fanin} < s] \leq \alpha^s \tag{1}$$

where α is the unique root of the equation

$$\left(1 + \frac{4p}{p+1}\frac{1}{\alpha}\right)^t = \left(1 + \frac{2p}{p+1}\frac{1}{\alpha}\right)^t + 1 \tag{2}$$

By virtue of Proposition 57, Inequality 1 is identical to

$$Pr[\min(G|_\rho) \geq s] \leq \alpha^s \tag{3}$$

where $\min(f)$ denotes the maximum size of the minterms of a Boolean function f.

We are going to use Hastad's Lemma to convert many ANDs of ORs gates to ORs of ANDs gates. So, we need the probability in Inequality 1 to be very small, since we want the sum of all these probabilities to be less than one. By duality, Hastad's Lemma is also true for converting an OR of ANDs to an AND of ORs, because ρ is symmetric with respect to values 0 and 1.

Before we go on to prove Hastad's Lemma, let's first examine the Equation 2.

Claim 59 *Equation 2 has a unique positive root.*

Proof. Let $x = \frac{2p}{p+1}\frac{1}{\alpha}$ and $q(x) = (1+2x)^t - (1+x)^t$. Then the equation becomes $q(x) = 1$. It is easy to check that $q'(x) > 0$ for $x > 0$ and $q(0) = 0$ and $q(1) \geq 1$. So, there exists a unique positive x and consequently there exists a unique positive α that satisfies the equation.

Q.E.D.

Claim 60 *The unique positive root of Equation 2 is* $\alpha \approx 2pt/\ln\phi$, *where* $\phi = \frac{1+\sqrt{5}}{2}$, *i.e.* $\alpha \approx 4.156pt$, *assuming* p *small and* t *large.*

Proof. Using the fact that $(1+a/t)^t \approx e^a$ we have that:

$$\left(1 + \frac{4p}{p+1}\frac{1}{\alpha}\right)^t \approx \left(1 + \frac{4p}{\alpha}\right)^t \approx e^{2\beta}$$

where $\beta = \frac{2pt}{\alpha}$. Similarly,

$$\left(1 + \frac{2p}{p+1}\frac{1}{\alpha}\right)^t \approx e^{\beta}$$

So, we have:

$$e^{2\beta} - e^{\beta} - 1 \approx 0 \Rightarrow e^{\beta} \approx \frac{1+\sqrt{5}}{2} \Rightarrow \beta \approx \ln(\frac{1+\sqrt{5}}{2})$$

Q.E.D.

Claim 61 *If* $p < 1/10$ *then* $\alpha < 5pt$.

Proof. It suffices to show that

$$\left(1 + \frac{4p}{p+1}\frac{1}{5pt}\right)^t - \left(1 + \frac{2p}{p+1}\frac{1}{5pt}\right)^t \leq 1$$

Using $e^{a-a^2/(2t)} \leq (1+a/t)^t \leq e^a$ which follows easily by considering the Taylor series of their logarithms, we have that:

$$\left(1 + \frac{4p}{p+1}\frac{1}{5pt}\right)^t \leq e^{\frac{4}{5(p+1)}} \leq e^{4/5} \leq 2.226$$

and

$$\left(1 + \frac{2p}{p+1}\frac{1}{5pt}\right)^t \geq e^{\frac{2}{5(p+1)} - \frac{4}{2\cdot 25(p+1)^2 t}} \geq e^{\frac{2}{5\cdot 1.1} - \frac{4}{50}} \geq 1.327$$

Q.E.D.

Trying to prove Lemma 58 by induction it turns out that a stronger induction hypothesis is needed. So, instead of proving Lemma 58 we will prove the following stronger lemma:

Lemma 62 (Hastad's First Switching Lemma Revisited) *Let G be an AND of ORs, with each OR of fanin $\leq t$. Let $0 < p < 1$. Let ρ be randomly chosen from the probabilistic space R_p. Let F be an arbitrary Boolean function. Then*

$$Pr[\min(G|_\rho) \geq s \mid F|_\rho \equiv 1] \leq \alpha^s \tag{4}$$

where α as in Lemma 58.

Notice that this implies Lemma 58, since we can take $F \equiv 1$. From now on we will assume that if $Pr[F|_\rho \equiv 1] = 0$ then $Pr[\min(G|_\rho) \geq s \mid F|_\rho \equiv 1] = 0$. Let $G = \bigwedge_{i=1}^{w} G_i$, where G_i is an OR of $\leq t$ literals. Before proving the lemma let's see what is going on in the simple case where each G_i has size t and no variables appear twice in G (G is a 'read-once' function). So, let $G_i = \bigvee_{j=1}^{t} x_{ij}$.

If $\rho(x_{ij}) = 1$ then $G_i|_\rho \equiv 1$ and may be omitted from $G|_\rho$. So, we can have $G_i|_\rho \not\equiv 1$ when

(1) $\rho(G_i) = \{0\}$ with probability $(\frac{1-p}{2})^t$, or

(2) $\rho(G_i) \neq \{0\}$ and $G_i|_\rho \not\equiv 1$ with probability $(\frac{1+p}{2})^t - (\frac{1-p}{2})^t$.

Intuitively, a minterm of size s exists iff (2) 'occurs' s times without (1) 'occuring'. Since the ratio of the probability of (2) over (1) is $\left(\frac{1+p}{1-p}\right)^t - 1 \approx e^{2pt} - 1 \approx 2pt \leq \alpha$ we have that (2) 'occurs' s times without (1) 'occuring' at all with probability at most α^s.

We will prove Lemma 62 by induction on w. First the base case, $w = 0$. Then $G \equiv 1$ and the lemma trivially holds.

Assume now that the lemma holds for every number of G_i's less than w. In order to use induction, let T be the set of variables occuring in G_1 and let the restriction ρ be chosen in two stages: First pick ρ_1, a random restriction for the variables in T and then pick a random restriction ρ_2 on the rest of the variables. It is clear that:

$$Pr[\min(G|_\rho) \geq s \mid F|_\rho \equiv 1] \leq max \left\{ \begin{array}{l} Pr[\min(G|_\rho) \geq s \mid F|_\rho \equiv 1 \wedge G_1|_\rho \equiv 1] \\ Pr[\min(G|_\rho) \geq s \mid F|_\rho \equiv 1 \wedge G_1|_\rho \not\equiv 1] \end{array} \right.$$

So, it suffices to bound both these probabilities by α^s. The first is easy:

$$Pr[\min(G|_\rho) \geq s \mid F|_\rho \equiv 1 \wedge G_1|_\rho \equiv 1] = Pr[\min(G|_\rho) \geq s \mid (F \wedge G_1|_) \rho \equiv 1] \leq \alpha^s$$

because we can apply the induction hypothesis i.e. the number of G_i's is $w-1$ since we can drop the $G_1 \equiv 1$ from G.

Without loss of generality, we can assume that $G_1 = \bigvee_{i=1}^{t} x_i$ (i.e. interchange x_i and $\overline{x}_i$ if necessary). If $G_1|_\rho \equiv 0$ then $\min(G|_\rho) = 0$, so we will assume that $G_1|_\rho \not\equiv 0$. Now, any minterm σ of $G|_\rho$ must have $\sigma(x_i) = 1$ for some $x_i \in T$ such that $\rho(x_i) = \star$. For any minterm σ of $G|_\rho$, let Y_σ be the set of $x_i \in T$ such that $\sigma(x_i) \neq \star$.

For $Y \subset T$, let $\min(G|_\rho)^Y \geq s$ denote the event: "$G|_\rho$ has a minterm σ of size $\geq s$ such that $Y_\sigma = Y$".

We have:

$$\begin{aligned}
&Pr[\min(G|_\rho) \geq s \mid F|_\rho \equiv 1 \wedge G_1|_\rho \not\equiv 1] \\
&\leq \sum_{\emptyset \neq Y \subseteq T} Pr[\min(G|_\rho)^Y \geq s \mid F|_\rho \equiv 1 \wedge G_1|_{\rho_1} \not\equiv 1] \qquad (5) \\
&= \sum_{\emptyset \neq Y \subseteq T} \Big(Pr[\rho_1(Y) = \star \mid F|_\rho \equiv 1 \wedge G_1|_{\rho_1} \not\equiv 1] \cdot \\
&\qquad \cdot Pr[\min(G|_\rho)^Y \geq s \mid F|_{\rho_1} \equiv 1 \wedge G_1|_\rho \not\equiv 1 \wedge \rho_1(Y) = \star] \Big)
\end{aligned}$$

where for notational convenience the singleton $\{\star\}$ is denoted by $\star$.

Now we bound each part separately.

Lemma 63 $Pr[\rho_1(Y) = \star \mid F|_\rho \equiv 1 \wedge G_1|_{\rho_1} \not\equiv 1] \leq \left(\frac{2p}{p+1}\right)^{|Y|}$.

First we drop the condition $F|_\rho \equiv 1$ and show that the lemma holds. The intuition behind this is that this condition can only decrease the probability.

So, if $G_1|_{\rho_1} \not\equiv 1$ then $\rho_1(Y) \subseteq \{0, \star\}$ and since the probability of $\star$ is p and the probability of 0 is $(1-p)/2$ we have:

$$Pr[\rho_1(Y) = \star \mid G_1|_{\rho_1} \not\equiv 1] = \left(\frac{p}{p + (1-p)/2}\right)^{|Y|} = \left(\frac{2p}{p+1}\right)^{|Y|}$$

Now we will need the following fact:

Fact 3 $Pr[A \mid B \wedge C] \leq Pr[A \mid C] \Leftrightarrow Pr[B \mid A \wedge C] \leq Pr[B \mid C]$

Proof. By the definition of conditional probability we have that:

$$Pr[A \mid B \wedge C] \leq Pr[A \mid C] \Leftrightarrow Pr[A \wedge B \wedge C]/Pr[B \wedge C] \leq Pr[A \wedge C]/Pr[C]$$

$$Pr[B \mid A \wedge C] \leq Pr[B \mid C] \Leftrightarrow Pr[B \wedge A \wedge C]/Pr[A \wedge C] \leq Pr[B \wedge C]/Pr[C]$$

Q.E.D.

So, in order to prove Lemma 63 we need to show that:

$$Pr[F|_\rho \equiv 1 \ \mid \ \rho_1(Y) = \star \wedge G_1|_{\rho_1} \not\equiv 1] \ \leq \ Pr[F|_\rho \equiv 1 \ \mid \ G_1|_{\rho_1} \not\equiv 1] \tag{6}$$

Consider the restrictions ρ with $\rho(G_1) \subseteq \{0, \star\}$ and for each fixed $\rho_1' \colon Y \to \{0, \star\}$ consider

$$P_{\rho_1'} = Pr_\rho[F|_\rho \equiv 1 \ \mid \ G_1|_{\rho_1} \not\equiv 1 \wedge \forall x \in Y : \rho_1(x) = \rho_1'(x)]$$

But $P_{\rho_1'}$ is minimized by ρ_1' such that $\rho_1'(Y) = \star$, because if $F|_\rho \not\equiv 1$, then modifying ρ in order to have $\rho_1(Y) = \star$, can't make $F|_\rho \equiv 1$. So, Inequality 6 holds and consequently Lemma 63 holds.

Now we are going to bound the second part:

$$Pr[\min(G|_\rho)^Y \geq s \ \mid \ F|_\rho \equiv 1 \wedge G_1|_\rho \not\equiv 1 \wedge \rho(Y) = \star] \tag{7}$$

Each minterm σ with $Y_\sigma = Y$ consists of two parts:

σ_1 with domain Y and

σ_2 with domain the remaining variables.

Since, by definition, σ is not defined on $T - Y$, σ_2 has domain the variables not in T. Let $\min(G|_\rho)^{Y,\sigma_1} \geq s$ be the event "$G|_\rho$ has a minterm σ of size s such that $Y_\sigma = Y$ and σ and σ_1 agree on Y". We shall consider σ_1 fixed and then we will look at the probability 7 with σ_2 varying.

Then the probability 7 is at most:

$$\sum_{\sigma_1 : Y \to \{0,1\}, \sigma_1(Y) \neq \{0\}} Pr[\min(G|_\rho)^{Y,\sigma_1} \geq s \ \mid \ F|_\rho \equiv 1 \wedge G_1|_{\rho_1} \not\equiv 1 \wedge \rho_1(Y) = \star]$$

Now we think of ρ_1 as being fixed and bound the probability by:

$$\sum_{\sigma_1: Y \to \{0,1\}, \sigma_1(Y) \neq \{0\}} \max_{\rho_1: T \to \{0,\star\}, \rho_1(Y) = \star} Pr_{\rho_2}[\min(G|_{\rho_1\rho_2})^{Y,\sigma_1} \geq s \ \mid \ F|_{\rho_1\rho_2} \equiv 1] \tag{8}$$

because we can drop the condition $G_1 \mid_{\rho_1} \not\equiv 1 \ \wedge \ \rho_1(Y) = \star$ which does not affect the probability.

The crucial point here is that:

$$\min(G|_{\rho_1\rho_2})^{Y,\sigma_1} \geq s \Rightarrow \min((G|_{\rho_1\sigma_1})|_{\rho_2}) \geq s - |Y|$$

We want to use the induction hypothesis to show that:

$$Pr_{\rho_2}[\min((G|_{\rho_1\sigma_1})|_{\rho_2}) \geq s - |Y| \ \mid \ (F|_{\rho_1})|_{\rho_2} \equiv 1] \ \leq \ \alpha^{s-|Y|} \tag{9}$$

But there is a problem here. In order to use the induction hypothesis we have to get rid of the variables in $T - \mathit{domain}(\rho_1) - Y$. The reason is that by definition, ρ_2 does not assign values to these variables. The solution is easy. Let $G'_{\rho\sigma_1}$ be $G|_{\rho\sigma_1}$ *but* with all variables in $T - \mathit{domain}(\rho_1) - Y$ omitted (deleted). This helps because:

Claim 64 *$G'_{\rho\sigma_1}|_{\rho_2}$ has a minterm $\geq s$ iff $(G|_{\rho\sigma_1})|_{\rho_2}$ has a minterm of size $\geq s$ which does not use any variables from T.*

The proof is easy. Let term be a partial assignment that makes a function 1 (true). Then a term of $G'_{\rho\sigma_1}|_{\rho_2}$ is precisely a term of $(G|_{\rho\sigma_1})|_{\rho_2}$ that does not involve variables in T. The claim follows.

So, we have that

$$Pr_{\rho_2}[\min((G|_{\rho_1\sigma_1})|_{\rho_2}) \geq s-|Y| \ \mid \ (F|_{\rho_1})|_{\rho_2} \equiv 1] \ = \ Pr_{\rho'_2}[\min(G'_{\rho\sigma_1}|_{\rho'_2}) \geq s-|Y| \ \mid \ (F|_{\rho_1})|_{\rho'_2} \equiv 1]$$

where ρ'_2, like ρ_2, is a restriction on the variables not in T. Notice that $G'_{\rho\sigma_1}$ has less that $w-1$ ORs.

We can now use the induction hypothesis. We get that:

$$Pr_{\rho'_2}[\min(G'_{\rho\sigma_1}|_{\rho'_2}) \geq s - |Y| \ \mid \ (F|_{\rho_1})|_{\rho'_2} \equiv 1] \ \leq \ \alpha^{s-|Y|}$$

and consequently Inequality 9 holds. So, the probability 8 is bounded by

$$\sum_{\sigma_1: Y \to \{0,1\}, \sigma_1(Y) \neq \{0\}} \max_{\rho_1: T \to \{0,\star\}, \rho_1(Y) = \star} \alpha^{s-|Y|} \ = \ \sum_{\sigma_1: Y \to \{0,1\}, \sigma_1(Y) \neq \{0\}} \alpha^{s-|Y|} \ = \ \left(2^{|Y|} - 1\right) \alpha^{s-|Y|}$$

Using this and Lemma 63 we can bound Probability 5 by

$$\sum_{\emptyset \neq Y \subseteq T} \left(\frac{2p}{p+1})\right)^{|Y|} \left(2^{|Y|}-1\right) \alpha^{s-|Y|} = \alpha^s \sum_{Y \subseteq T} \left(\frac{2p}{p+1}\frac{1}{\alpha}\right)^{|Y|} (2^{|Y|}-1)$$

We have used the fact that $Y = \emptyset$ does not contribute to the sum. But now this is equal to

$$\begin{aligned}
& \alpha^s \sum_{i=0}^{|T|} \binom{|T|}{i} \left[\left(\frac{4p}{p+1}\frac{1}{\alpha}\right)^i - \left(\frac{2p}{p+1}\frac{1}{\alpha}\right)^i] \right] \\
&= \alpha^s \left[\left(1+\frac{4p}{p+1}\frac{1}{\alpha}\right)^{|T|} - \left(1+\frac{2p}{p+1}\frac{1}{\alpha}\right)^{|T|} \right] \\
&\leq \alpha^s \left[\left(1+\frac{4p}{p+1}\frac{1}{\alpha}\right)^{t} - \left(1+\frac{2p}{p+1}\frac{1}{\alpha}\right)^{t} \right] \\
&\leq \alpha^s
\end{aligned}$$

by the definition of α. This concludes the proof of Hastad's Lemma.

Example 7 Let us give an example here to help clarify some points in the proof.

Let $G = (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_4) \wedge (x_1 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee x_5)$, then $T = \{x_1, x_2, x_3\}$.

Suppose that $\rho(x_1) = \rho(x_2) = \rho(x_3) = \star$ and consider $Y = \{x_2, x_3\}$, i.e. σ_1 can be x_2x_3, $\overline{x}_2x_3$, or $x_2\overline{x}_3$. Let $\sigma_1 = \overline{x}_2x_3$. Then

$G|_{\rho_1} \equiv G$ and

$G|_{\rho_1\sigma_1}$ is $1 \wedge (x_1 \vee x_4) \wedge (x_1 \vee 1) \wedge (\overline{x}_1 \vee x_5) \equiv (x_1 \vee x_4) \wedge (\overline{x}_1 \vee x_5)$

ρ_2 has domain $\{x_4, x_5\}$

$G'_{\rho_1\sigma_1}$ is $(x_4) \wedge (x_5)$ since we drop x_1 that is in T and is undefined by both ρ and σ_1.

Notice that the only minterm of $G'_{\rho_1\sigma_1}$ is x_4x_5 and the only minterm of $G|_\rho$ which agrees with σ_1 in T is $\sigma = \overline{x}_2x_3x_4x_5$

□

References

J.Hastad. *Almost Optimal Lower Bounds For Small Depth Circuits*, Advances in Computing Research, 5 (1989) 143–170. (Also appeared in STOC '86)

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: March 10-12, 1992 **Topic: Applications of Hastad's Switching Lemma**

Instructor: Sam Buss **Notes by: Sam Buss**

We now discuss how Hastad's Switching Lemma can be used to obtain lower bounds on the size of constant-depth (and restricted depth) circuits for PARITY.

Theorem 65 *There are no depth k circuits of size*

$$\leq 2^{\left(\frac{1}{10}\right)^{\frac{k}{k-1}} \cdot n^{\frac{1}{k-1}}}$$

for $Parity_n$, provided $n > N^k$ for N some constant.

Corollary 66 *Polynomial size circuits which compute* Parity *must have depth at least*

$$\log n/(c + \log\log n)$$

for some constant c.

Proof. of Corollary 66 from Theorem 65: Given a family of polynomial-size circuits of depth $k(n)$, we must have

$$2^{\left(\frac{1}{10}\right)^{\frac{k(n)}{k(n)-1}} \cdot n^{\frac{1}{k(n)-1}}} \quad < \quad n^a \quad = \quad 2^{a \log n} \quad = \quad 2^{2^{(\log a + \log\log n)}},$$

for some constant a. Define $c(n)$ to be the real number such that $k(n) = \log n/(c(n) + \log\log n)$. We must prove that $c(n)$ is bounded by a constant.

Now,

$$\begin{aligned}
\left(\tfrac{1}{10}\right)^{\frac{k(n)}{k(n)-1}} \cdot n^{\frac{1}{k(n)-1}} &= \left(\tfrac{1}{10}\right)^{\frac{\log n}{\log n - c(n) - \log\log n}} \cdot n^{\frac{c(n)+\log\log n}{\log n - c(n) - \log\log n}} \\
&= n^{\frac{c(n)+\log\log n - \log 10}{\log n - c(n) - \log\log n}} \\
&= 2^{\frac{c(n)+\log\log n - \log 10}{(\log n - c(n) - \log\log n)/\log n}} \\
&\geq 2^{(\log\log n + c(n) - \log 10)}.
\end{aligned}$$

And thus $c(n) - \log 10 < \log a$, so $c(n) = O(n)$, which is what we wished to show.

Q.E.D.

Before proving Theorem 65, we need the following lemma (we are still using the convention that circuits use unbounded fanin AND's and OR's in alternating levels):

Lemma 67 *$Parity_n$ can not be computed by a depth k circuit containing $\leq 2^{\left(\frac{1}{10}\right)} \cdot n^{\frac{1}{k-1}}$ gates at each level except possibly the bottom level, and having bottom fanin $\leq \left(\frac{1}{10}\right) n^{\frac{1}{k-1}}$; for all $n > N^k$ (where N is a fixed constant).*

Proof. The proof of Lemma 67 is by induction on k.

Base Case: $k = 2$. This case is easy since any disjunctive or conjunctive normal form expression for $Parity_n$ must have bottom fanin equal n.

Induction Step: Let $k > 2$. Suppose, the sake of a contradiction that a circuit C exists which violates the statement of Lemma 67. We shall apply Hastad's Switching Lemma to show that, in this case, there is a circuit of depth $k-1$ that violates Lemma 67.

We now apply Hastad's Switching Lemma to C: the fanin of bottom gates in C is bounded by $t = \left(\frac{1}{10}\right) n^{\frac{1}{k-1}}$; the restriction ρ is chosen randomly from R_p where $p = n^{\frac{-1}{k-1}}$. (Thus ρ is expected to leave $\approx n \cdot n^{\frac{-1}{k-1}} = n^{\frac{k-2}{k-1}}$ variables $*$'ed.)

After applying the random restriction ρ, we can rewrite a *single* (arbitrary) bottom AND of OR's in C (or OR of AND's) as an OR of AND's (or AND of OR's, respectively) of fanin $\leq s$ with probability at least $1 - \alpha^s$. In previous lectures, we proved that $\alpha < 5pt$ provided $p < \frac{1}{10}$, but examination of the proof shows that $\alpha < (5-\epsilon)pt$ for some constant ϵ; thus

$$\alpha \ < \ (5-\epsilon)pt \ < \ \frac{5-\epsilon}{10} n^{\frac{-1}{k-1}} n^{\frac{1}{k-1}} \ = \ \frac{1}{2} - \delta$$

for some constant δ. (We needed $p < \frac{1}{10}$, but this will hold if $N \geq 10$.) Let $s = \frac{1}{10} n^{\frac{1}{k-1}}$; the probability that *any* of the bottom depth 2 subcircuits of C can not be switched is

$$\leq \ 1 - \left(2^{\left(\frac{1}{10}\right) n^{\frac{1}{k-1}}}\right) \cdot \alpha^{\left(\frac{1}{10}\right) n^{\frac{1}{k-1}}} \ = \ 1 - (1-2\delta)^{\left(\frac{1}{10}\right) n^{\frac{1}{k-1}}}$$

which tends to 1 for $n^{\frac{1}{k-1}}$ large enough (i.e., $n > N^k$).

On the other hand, with probability $> \frac{1}{3}$, the random restriction ρ leaves at least $n^{\frac{k-2}{k-1}}$ variables $*$'ed. So with non-zero probability, there is a ρ for which the follow hold:

(a) Leaves at least $m = n^{\frac{k-1}{k-2}}$ variables $*$'ed, and

(b) Collapses the bottom 3 levels in 2 — making the circuit into a depth $k-1$ circuit, and

(c) Makes the new bottom fanin $\frac{1}{10}n^{\frac{1}{k-1}} = \frac{1}{10}m^{\frac{1}{k-2}}$, and

(d) Has $\leq 2^{\left(\frac{1}{10}\right)n^{\frac{1}{k-1}}} = 2^{\left(\frac{1}{10}\right)m^{\frac{1}{k-2}}}$ gates at each level, except possibly the bottom level. In fact, the number of gates on any level except the bottom level is unchanged.

That completes the proof of Lemma 67

Q.E.D.

Proof. Of Theorem 65.

Assume, for sake of contradiction, that we are given a depth k circuit for parity which satisfies the size and fanin conditions in Theorem 65. This circuit can be viewed as a depth $k+1$ circuit of bottom fanin 1, by adding a bottom level of one input gates.

Use a random restriction from R_p with $p = \frac{1}{10}$. Take $s = \left(\frac{1}{10}\right)^{\frac{k}{k-1}} n^{\frac{1}{k-1}}$ and $t = 1$ and apply Hastad's Switching Lemma. By calculations exactly as in the proof of Lemma 67, we obtain, with non-zero probability, a depth k circuit with $m \geq \frac{1}{10}n$ variables and with bottom fanin $\leq s \leq \frac{1}{10}m^{\frac{1}{k-1}}$ and with $\leq 2^{\left(\frac{1}{10}\right)^{\frac{k}{k-1}} \cdot n^{\frac{1}{k-1}}} \leq 2^{\left(\frac{1}{10}\right)^{\frac{k}{k-1}} \cdot (10m)^{\frac{1}{k-1}}} \leq 2^{\left(\frac{1}{10}\right) \cdot m^{\frac{1}{k-1}}}$ gates at each level other than the bottom level. But this is impossible by Lemma 67.

Q.E.D.

Constant Depth Circuits and the Polynomial Time Hierarchy

Reference: Furst-Saxe-Sipser, Math. Systems Theory, 17 (1984) 13-27.

Next we discuss the connections between constant depth, polynomial size (more correctly, $2^{(\log n)^{O(1)}}$ size) circuits and relativized Turing compuation in the polynomial time hierarchy.

Observation: There is a PSPACE Turing Machine M^ϕ where ϕ denotes an oracle $\phi \supseteq \{0,1\}^*$, such that for all oracle ϕ and all inputs of length n, M^ϕ accepts if and only if there are an even number of strings $w \in \phi$ of length equal to n.

Theorem 68 *There is no oracle Turing machine M^ϕ in the (relativized) class Σ_k^p of the polynomial time hierarchy which computes for* every *oracle ϕ the same parity predicate as the Turing machine in the Observation.*

Proof. We shall give an informal sketch of the proof. The essential idea is that a Σ_k^p Turing machine M^ϕ can be translated into a family of depth $k+1$ circuits of *super polylogarithmic* size. And then since there are no super polylogarithmic size, constant depth circuits for Parity (by Theorem 65) then there can not be a Σ_k^p Turing machine computing the parity of the number of strings of length n in an oracle.

Since M^ϕ is in Σ_k^p, M can w.l.o.g. be presumed to make k blocks of alternating existential and universal guesses and then perform some polynomial time computation. In particular, M^ϕ can be constrained to make all non-deterministic moves before consulting the oracle. Thus $M^\phi(x)$ accepts if and only if the following predicate is true:

$$(\exists x_1, |x_1| < p(|x|))(\forall x_2, |x_2| < p(|x|)) \cdots (Qx_k, |x_k| < p(|x|))R^\phi(x, x_1, \ldots, x_k)$$

where Q is $\forall$ or $\exists$ depending on whether k is even or odd and where R^ϕ computable by a polynomial time Turing machine which may consult the oracle ϕ. Each $R^\phi(\vec{x})$ may ask up to $q(n)$ queries where $n = |x|$ and q is some fixed polynomial (q depends only on M). Thus, there are 2^n many possible sequences of Yes/No answers that R^ϕ may receive to its oracle queries. Hence, for fixed values of $\vec{x}$, the predicate $R^\phi(\vec{x})$ may be computed by an OR of AND's where there are $\leq 2^{q(n)}$ many AND's, one AND for each sequence of oracle answers that causes $R^\phi(\vec{x})$ to accept; and where each AND is a conjunction of statements $w \in \phi$ or $w' \notin \phi$ that describe the sequence of oracles that the AND corresponds to.

In other words, for *fixed* $x, x_1, \ldots, x_k$ the predicate $R^\phi(\vec{x})$ is a OR of $\leq 2^{q(n)}$ ANDs of fanin $\leq q(n)$. Similar reasoning (applied to the negation of R^ϕ) shows that that the predicate $R^\phi(\vec{x})$ can be expressed as an AND of $\leq 2^{q(n)}$ ORs of fanin $\leq q(n)$ with the inputs the ORs are of the forms $w \in \phi$ and $w' \notin \phi$. Taking $R^\phi(x)$ to be an OR of AND's if k is odd and to be an AND or OR's if k is even, we see that the condition that $M^\phi(\vec{x})$ accepts can be written as a depth $k+1$ unbounded fanin circuit with $\leq 2^{r(n)}$ gates at each level for some polynomial r. Without loss of generality, $M^\phi(0^n)$ asks only queries of the form "$w \in \phi$?" with $|w| = n$ (since if M^ϕ correctly computes the parity of the number of strings $w \in \phi$ of length n for *all* oracles ϕ, then M^ϕ can correctly compute the parity without consulting strings not of length n).

Now, for each string w, let x_w be a propositional variable. In the constant depth circuits for M^ϕ constructed in the previous paragraphs, replace each query "$w \in \phi$?" with the variable x_w and replace each query "$w \notin \phi$?" with the negated variable $\overline{x}_w$. For fixed n, there are $m = 2^n$ strings of length w and thus $m = 2^n$ many variables x_w. Thus, the above construction has given circuits that compute the parity of m many bits and these circuits are of constant depth and of size $2^{(\log m)^c}$ for some constant c. But this contradicts Theorem 65

Q.E.D.

Theorem 68 states that there is no polynomial hierarchy Turing machine M which computes parity for **all** oracles ϕ. This can be strengthened (by a diagonalization argument) to show that there is a single oracle ϕ for which no polynomial hierarchy Turing machine can compute the parity of all strings of length n (on input of length n). This oracle separates the polynomial time hierarchy from polynomial space.

J. Cai has shown that a randomly chosen oracle separates the polynomial time hierachy from polynomial space. (We shall discuss this further in a later lecture; the argument above just shows that there is at least one oracle separating the polynomial time hierarchy from polynomial space.)

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 7 April 1992 **Topic: Constant Depth Reducibilities**
Instructor: Sam Buss **Notes by: Frank Baeuerle**

References:

a. Chandra-Stockmeyer-Vishkin, SIAM J. Comput. Vol 13, 1984, 423-439 "Constant Depth Reducibility"

b. R. Smolensky, 19^{th} STOC, 1987, 77-82

Notation: $F = \{f_n\}$ is a family of boolean functions, $f_n : \{0,1\}^n \longrightarrow \{0,1\}^{m_n}$ $(m_n \leq n^{\mathcal{O}(1)})$

Definition 36 *$F \leq_{cd} G$, F is constant depth reducible to G, iff each $f_n(x_1, \ldots, x_n)$ can be computed by a constant depth, polynomial size circuit using $\neg$ gates, unbounded fan in $\wedge, \vee$ gates and G-gates. A G-gate computes g_k (multiple G-gates with different k can occur) and the size of such a gate is the number k of inputs and its depth is one.*

Definition 37 *$F \leq_{cd-tt} G$, F is constant depth truth table reducible to G, iff $F \leq_{cd} G$ with the additional condition that on any path in the circuit for f_n there is at most one G-gate.*

Definition 38 *$F \leq_{proj} G$, F is projection reducible to G, iff for all $n f_n(x_1, \ldots, x_n)$ can be computed by a circuit which contains one G-gate with inputs chosen from $0, 1, x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}$ such that the outputs of the circuit are chosen from the outputs of the G-gate and the number of inputs of the G-gate is $\leq p(n)$ for some polynomial p.*

Definition 39 *$F \leq_{proj} G$ iff above holds except no negated inputs are allowed.*

Remark: We always give positive projection reductions in this class. See Skyum-Valiant for original definition of $\leq_{proj}$.

Definitions 1-4 also apply to families of F's and G's. For example, if $Exact_n^k(x_1, \ldots, x_n) = \begin{cases} 1 & \text{iff } \sum x_i = k \\ 0 & \text{otherwise} \end{cases}$ then when we write $Majority \leq_{proj} \{Exact_n^0, \ldots, Exact_n^k\}$ we mean:

$$Majority \leq_{proj} \{g_n\} \text{ where } g_n(x_1, \ldots, x_n) = (Exact_n^0, \ldots, Exact_n^k).$$

Theorem 69 *$\leq_{proj}, \leq_{cd}, \leq_{cd-tt}$ are transitive and reflexive.*

Proof: Obvious

Theorem 70 $F \leq_{proj} G \Rightarrow F \leq_{cd} G \Rightarrow F \leq_{cd-tt} G$

Proof: Obvious

Theorem 71 *If $F \leq_{cd} G$ and $G \in AC^i$ then $F \in AC^i$. In particular for $i = 0$, if $G \in AC^0$ then $F \in AC^0$*

Proof. For $n > 0, f_n(x_1, \ldots, x_n)$ is computed by a circuit of polynomial size and constant depth containing $\{\neg, \vee, \wedge\}$ and G-gates. Since $G \in AC^i$ each G-gate (with $n^{\mathcal{O}(1)}$ inputs) can be replaced by a $\mathcal{O}\left(\log^i(n^{\mathcal{O}(1)})\right)$ depth polysize circuit. The result is a depth $\mathcal{O}\left((\log n^{\mathcal{O}(1)})^i\right) = \mathcal{O}\left(\log^i n\right)$ circuit which is also polysize in n.

Q.E.D.

Corollary 72 *This Theorem also holds for $\leq_{proj}, \leq_{cd-tt}$.*

Definition 40 *Let $m > 1$*

$$Mod\text{-}m(x_1, \ldots, x_n) = \begin{cases} 0 & \text{iff } \sum_{i=1}^n x_i = 0 \ mod\ m \\ 1 & \text{otherwise} \end{cases}$$

With this definition $Parity \equiv Mod\text{-}2$.

Theorem 73 *Fix $c > 0$*
(a) $Parity \leq_{proj} Mod\text{-}2^c$
(b) $Mod\text{-}2^c \leq_{cd-tt} Parity$

Proof. (a) Make 2^{c-1} copies of each x_i and feed them into *Mod-*2^c.

$$Parity(x_1,\ldots,x_n) \equiv \textit{Mod-}2^c(\underbrace{x_1,\ldots,x_1}_{2^{c-1}},\underbrace{x_2,\ldots,x_2}_{2^{c-1}},\ldots,\underbrace{x_n,\ldots,x_n}_{2^{c-1}})$$

(b) By induction on c:
Base case $c = 1$ is obviously true. For the induction step it suffices to show:

$$\textit{Mod-}2^c \leq_{cd-tt} \textit{Mod-}2^{c-1}$$

The inputs are $x_1,\ldots,x_n$. Let $y_{ij} = x_i \wedge x_j$ for $1 \leq i < j \leq n$ (Depth 1, $\mathcal{O}(n^2)$ size). If $s = \sum x_i$ then $\sum y_{ij} = \frac{s(s-1)}{2}$. Now $s \equiv 0 \ mod\ 2^c$ iff ($s \equiv 0 \ mod\ 2^{c-1}$ and $\frac{s(s-1)}{2} \equiv 0 \ mod\ 2^{c-1}$). Thus it suffices to compute *Mod-*$2^{c-1}(x_1,\ldots,x_n)$ and *Mod-*$2^{c-1}(y_{ij}, 1 \leq i < j \leq n)$ to compute *Mod-*$2^c(x_1,\ldots,x_n)$.

Q.E.D.

Facts:
(a) If $p|m$ then *Mod-*$p \leq_{cd-tt}$ *Mod-*m
(b) For p a prime, c a constant, *Mod-*$p^c \leq_{cd-tt}$ *Mod-*p
(c) If m's prime factors are $p_1,\ldots,p_k$ then

$$\textit{Mod-}m \leq_{cd-tt} \{\textit{Mod-}p_1,\ldots,\textit{Mod-}p_k\}$$

Definition 41

- $Exact_n^k(x_1,\ldots,x_n) = 1$ *iff* $\sum x_i = k$
- $Threshold_n^k(x_1,\ldots,x_n) = 1$ *iff* $\sum x_i \geq k$
- $Majority(x_1,\ldots,x_n) = 1$ *iff* $\sum x_i \geq \frac{n}{2}$
- *Addition (two* m *bit integers in binary* $\rightarrow$ *their sum in binary (*$m+1$ *bits))*
- *Multiplication (two* m *bit integers in binary* $\rightarrow$ *their product in binary (at most* $2m-1$ *bits))*
- *Vectoraddition (*m *m-bit numbers* $\longrightarrow m + \lceil \log m \rceil$ *bit sum)*
- *Binary Count*$(x_1,\ldots,x_n) = \sum x_i$ *coded in binary*
- *Unary Count*$(x_1,\ldots,x_n) = \sum x_i$ *coded in unary (= Unary Sorting)*

Recall $Addition \in AC^0$. For fixed $k, Exact_n^k$ and $Threshold_n^k \in AC^0$

$$Threshold_n^k(x_1, \ldots, x_n) = \bigvee_{1 \leq i_1 < i_2 < \ldots < i_k \leq n} x_{i_1} \wedge \ldots \wedge x_{i_k}$$

This is a depth 2 size $n^k = n^{\mathcal{O}(1)}$ circuit.

$$Exact_n^k(x_1, \ldots, x_n) = Threshold_n^k(x_1, \ldots, x_n) \wedge \neg Threshold_n^{k+1}(x_1, \ldots, x_n)$$

Theorem 74 *$Majority \leq_{cd-tt} \{Exact_n^k : 0 \leq k \leq n\}$*
$Majority \leq_{proj} \{Threshold_n^k : 0 \leq k \leq n\}$

Proof.

$$\begin{aligned} Majority(x_1, \ldots, x_n) &= Threshold_n^{\lceil \frac{n}{2} \rceil}(x_1, \ldots, x_n) \\ &= \bigvee_{k=\lceil \frac{n}{2} \rceil}^{n} Exact_n^k(x_1, \ldots, x_n) \end{aligned}$$

Q.E.D.

Theorem 75 *$Majority \leq_{proj}$ Unary Count*
Binary Count $\leq_{cd-tt}$ Unary Count

Proof.

$$Unary\ Count(x_1, \ldots, x_n) = (Threshold_n^1(\vec{x}), \ldots, Threshold_n^n(\vec{x}))$$

and $Majority = \lceil \frac{n}{2} \rceil^{th}$ bit of *Unary Count*. The i^{th} bit of $Binary\ Count(x_1, \ldots, x_n)$ is computed by

$$\bigvee_{\substack{m \in \{1, \ldots, n\} \\ i^{th} \text{ bit of } m \text{ in} \\ \text{binary is a 1}}} \Big(\underbrace{Threshold_n^m(\vec{x}) \wedge \neg Threshold_n^{m+1}(\vec{x})}_{m^{th} \text{ and } m+1^{st} \text{ bits of } Unary\ Count(\vec{x})} \Big)$$

Q.E.D.

Theorem 76 *Unary Count $\leq_{cd-tt}$ Binary Count*

Let $y_1, \ldots, y_{\lceil \log n \rceil}$ be the output of $Binary\ Count_n(x_1, \ldots, x_n)$. $Exact_n^i(x_1, \ldots, x_n)$ is $\pm y_1 \wedge \ldots \wedge \pm y_{\lceil \log n \rceil}$ where $\pm$'s are chosen by the binary representation of i. Now the i^{th} bit of Unary Count $(\vec{x}) is \bigvee_{j=i}^{n} Exact_n^j(\vec{x})$

Theorem 77 *(a) Parity $\leq_{proj}$ Multiplication [due to Furst, Saxe, Sipser]*
(b) Binary Count $\leq_{proj}$ Multiplication

Proof. Suffices to prove (b). To compute $Binary\ Count(x_1, \ldots, x_n)$ let $l = \lceil \log n \rceil$. Form following integers in binary:

$$1\overbrace{00\cdots0}^{l}1\overbrace{00\cdots0}^{l}\ldots1\overbrace{00\cdots0}^{l}1$$

$$x_n\overbrace{00\cdots0}^{l}x_{n-1}\overbrace{00\cdots0}^{l}\ldots x_2\overbrace{00\cdots0}^{l}x_1$$

such that there are n ones (or x_i's).The block size $l+1$ avoids carries from one block to the next. In the center block of the product is $x_1 + x_2 + \ldots + x_n$ in binary.

Q.E.D.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 9 April 1992 **Topic: Constant Depth Reducibilities**
Instructor: Sam Buss **Notes by: Frank Baeuerle**

Theorem 78 *Multiplication* $\leq_{cd-tt}$ *Vector Addition*

Proof: Use "highschool" method of multiplication

Theorem 79 (Chandra-Stockmeyer-Vishkin) *Vector Addition*$\leq_{cd-tt}$ *Binary Count*

Proof. Inputs to *Vector Addition* are m m-bit numbers $(x_{i,m-1}, \ldots, x_{i,0})_2$ is the m^{th} number.
Circuit Computation:

Stage 1: For each k compute *Binary Count*$(x_{1,k}, \ldots, x_{n,k})$, the sum of the bits in the k^{th} column of the summation. For each k this gives a $l_1 = \lceil \log m \rceil$ bit number. Pack these l_1-bit numbers into l_1 many $m + l_1 - 1$ bit numbers. These have the same sum as the input numbers.

Stage 2: Let $l_2 = \lceil \log l_1 \rceil$ and compute the sums of the bits in each column of the result of stage 1. This produces l_2 many numbers of $\leq m + \lceil \log m \rceil$ bits each, with the same sum as the inputs. But, instead of using *Binary Count* gates to compute sums use a DNF circuit. Each column has $l_1 = \lceil \log m \rceil$ many bits, thus using a DNF circuit with l_1 inputs has size $\mathcal{O}\left(2^{l_1}\right) = \mathcal{O}(m)$.
Stage 3 thru i^*: In general let $l_{i+1} = \lceil \log l_i \rceil$. Let i^* be the least number such that $l_{i^*} = 2$. So $i^* \approx \log^* m$.
Idea: Iterate in stages 3-i^* same process until 2 numbers are obtained. After stage i^* the two numbers can be added with an AC^0-circuit. Problem: straightforward iteration gives $\log^* m$ depth. To fix this compute stages 3-i^* in one step using DNF-circuits. This is possible since these stages compute a boolean function, thus we can collapse them into one DNF- circuit. Any signal computed in stage j depends on l_{j-1} many signals from the previous stage. So results from stage i^* depends on $l_{i^*-1} \cdot l_{i^*-2} \cdot \ldots \cdot l_2$ many bits from stage 2. So it will suffice to show that

$$l_{i^*-1} \cdot l_{i^*-2} \cdot \ldots \cdot l_2 = \mathcal{O}(\log m)$$

$$\approx (\log\log m)\cdot(\log\log\log m)\cdot\ldots = \mathcal{O}(\log m)$$

To see this consider the following:
$\forall a > 2, (\lceil \log a \rceil)^2 < 2a$ and $\forall a \geq 37, (\lceil \log a \rceil)^2 < a$
Now $l_{i^*} = 2$ and $l_{i^*-1} > 2$. Thus

$$\begin{aligned}
l_{i^*-2} &> \frac{1}{2}(l_{i^*-1})^2 > \frac{1}{2} l_{i^*-1} l_{i^*} \\
l_{i^*-3} &> \frac{1}{2}(l_{i^*-2})^2 > \frac{1}{4} l_{i^*-2} l_{i^*-1} l_{i^*} \\
l_{i^*-4} &> 37 \text{ so} \\
l_{i^*-4} &> (l_{i^*-3})^2 > \frac{1}{4} l_{i^*-3} l_{i^*-2} l_{i^*-1} l_{i^*}
\end{aligned}$$

by induction

$$l_{i^*-i'} > \frac{1}{4} l_{i^*-i'+1} l_{i^*-i'+2} \ldots l_{i^*-1} l_{i^*}$$

Q.E.D.

Homework*: Is the $\leq_{cd-tt}$ reduction of *Vector Addition* to *Binary Count logtime* or *loghierarchy* or *logspace* uniform? (it is *polytime* uniform, *logspace* uniformity pretty easy)

Homework: Show *Binary Count* $\leq_{cd-tt}$ *Majority* (Suffices to show *Unary Count* $\leq_{cd-tt}$ *Majority*) (due April 14)

Homework: Show $\forall q \in N$ *Mod-q* $\leq_{cd-tt}$ *Binary Count*

Definition 42 $TC^0 = \{f : f \leq_{cd} Majority\}$

So far we have

$$Addition \begin{array}{c}\leq_{cd-tt}\\ \not\geq_{cd-tt}\end{array} \left\{\begin{array}{c} Mod\text{-}1 \\ Mod\text{-}2 \\ Mod\text{-}3 \\ . \\ . \end{array}\right\} \leq_{cd-tt} \left\{\begin{array}{c} \text{Majority} \\ \text{Unary Count} \\ \text{Binary Count} \\ \text{Multiplication} \\ \text{Vector Addition} \\ \{Exact_n^k : n \geq k \geq 0\} \\ \{Threshold_n^k : n \geq k \geq 0\} \end{array}\right\} \in NC^1$$

The functions *Mod-q*, $q \geq 2$ will be defined later as ACC. We also know that Majority $\not\leq_{cd}$ *Mod-*p^k, p a prime by Smolensky

Mod-2 $\notin AC^0$ (*Mod*-2$q \notin AC^0$) by Hastad

Open Questions:

a. $TC^0 = NC^1$??

b. $\{f : f \leq_{cd} \textit{Mod-6}\} \equiv TC^0$??

c. $ACC = TC^0$??

References:

a. Razborov, 1986

b. Smolensky, 19^{th} STOC, 1987

Definition 43 *$F = \{f_n\}, F \in ACC$ iff $\exists q \in N$ such that $F \leq_{cd}$ Mod-q.*

Equivalently, $F \in ACC$ iff $\exists$ primes $p_1, p_2, \ldots, p_k$ such that $F \leq_{cd} \{\textit{Mod-}p_1, \textit{Mod-}p_2, \ldots, \textit{Mod-}p_k\}$.

Definition 44 *Let F be a field. $U_F^n = \{f : \{0,1\}^n \longrightarrow F\}$. An F-valued function $f(x_1, \ldots, x_n)$ is called a Boolean function if range$(f) \subseteq \{0,1\}$, where 0,1 are the zero and one of F.*

Goal: Translate Boolean operations to operations in the field. $\neg$ can be represented by $f(x) = 1 - x$, $\wedge$ can be represented by $f(x_1, x_2) = x_1 \cdot x_2$. If F has characteristic p represent *Mod-p* by addition and raising to power $p - 1$.

In Z_p, for $x \neq 0, x^p = x$ and $x^{p-1} = 1$.

U_F^n has $+, \cdot$, which are pointwise operations, i.e. $(f + g)(x) = f(x) + g(x)$ etc. . U_F^n is a commutative ring and has a vector space structure, more generally it is an algebra.

Represent $f \in U_F^n$ as a polynomial (in $F(x_1, \ldots, x_n)$). U_F^n is a 2^n dimensional vector space (over F). A basis is

$$\{f : f(\vec{x}) = \left\{ \begin{array}{lcl} 1 & = & \text{for one value of } \vec{x} \\ 0 & = & \text{for rest of the values of } \vec{x} \end{array} \right\}$$

$\vec{x}$ ranges over vectors of 0,1 entries of length n. Thus there are 2^n many basis elements.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 14 April 1992 **Topic: Smolensky's theorem (cont'd)**
Instructor: Sam Buss **Notes by: Frank Baeuerle**

Let F be a field. $U_F^n = \{f : \{0,1\}^n \longrightarrow F\}$. U_F^n is a commutative ring and a vector space over F.

Examples:

a. $f(x) = 1 - x$

b. $f(x_1, \ldots, x_n) = (x_1 + x_2 + \ldots + x_n)^{p-1}$ is *Mod-p* provided $char(F) = p$.

Representing $f \in U_F^n$ by polynomials:

a. If $p(X_1, \ldots, X_n)$ with $X_1, \ldots, X_n$ new variables then p represents a function in U_F^n. p is a polynomial over F.

b. Conversely, any $f \in U_F^n$ can be represented by a polynomial over F. Fix $\vec{d} = (d_1, \ldots, d_n) \in \{0,1\}^n$

$$p_{\vec{d}}(X_1, \ldots, X_n) = \Big(\prod_{\substack{i \text{ such that} \\ d_i = 1}} X_i\Big) \cdot \Big(\prod_{\substack{i \text{ such that} \\ d_i = 0}} (1 - X_i)\Big)$$

$p_{\vec{d}}(\vec{d}) = 1$ and for all $\vec{x} \neq \vec{d}, p_{\vec{d}}(\vec{x}) = 0$. So

$$f(\vec{x}) = \sum_{\vec{d} \in \{0,1\}^n} f(\vec{d}) \cdot p_{\vec{d}}(X_1, \ldots, X_n)$$

This is the polynomial representation of f and its degree is n. So $f(\vec{x})$ is represented by a linear combination of monomials. Each monomial contains some of $X_1, \ldots, X_n$. Each X_i in a monomial has degree ≤ 1.

c. f is uniquely representable by a linear combination of monomials. This is because

1. there are 2^n many monomials
2. U_F^n is a vector space of dimension 2^n

Example: "ring sum expansion of a Boolean function"
Let $F = Z_2 = (\{0,1\}, \bigoplus, \wedge, 0, 1)$. Any Boolean function is uniquely representable as follows: Z is a set of subsets of $x_1, \ldots, x_n$

$$f(x) = \bigoplus_{Y \in Z} (\bigwedge Y)$$

with $\bigwedge \emptyset = 1$ and $\bigwedge\{x_1, x_2\} = x_1 \wedge x_2$. This is a canonical representation of f over basis $(\wedge, \bigoplus, 1)$

Algebraic description of U_F^n is

$$F(x_1, \ldots, x_n)(mod) x_1^2 = x_1, x_2^2 = x_2, \ldots, x_n^2 = x_n$$

Definition 45 *J is an ideal of U_F^n iff*

a. $J \subseteq U_F^n, J \neq \emptyset$

b. If $f, g \in J$ then $f - g \in J$ ($0 \in J, -f \in J, f + g \in J$)

c. If $f \in J, g \in U_F^n$, then $f \cdot g \in J$

Example: Let E be a subset of $\{0,1\}^n$. Let $J = \{f : f(x) = 0 \text{ for all } \vec{x} \notin E\ \}$. This is an ideal.

Proposition 80 *Any ideal J of U_F^n can be obtained as in this example.*

Proof. Let $E_J = \{\vec{x} \in \{0,1\}^n : \exists f \in J, f(\vec{x}) \neq 0\}$
Let

$$f_E(\vec{x}) = \begin{cases} 1 & : \quad x \in E_J \\ 0 & : \quad x \notin E_J \end{cases}$$

Claim: $f_E \in J$
To prove the claim it suffices to show that there exists an $f \in J$ such that $f(\vec{x}) \neq 0$ for all

$x \in E_J$. Since then $f_E = f \cdot f^*$ where $f^*(\vec{x}) = \begin{cases} \frac{1}{f(\vec{x})} & if \quad f(\vec{x}) \neq 0 \\ 0 & otherwise \end{cases}$ Prove the claim by showing (by induction) for $j = 0, \ldots, |E_J|$ that there exists $f \in J$ with $\geq j$ non-zero values.
Base case: $j = 0$ is obvious
Induction Step: Suppose $f(\vec{x})$ has $j < |E_J|$ non-zero values, pick $\vec{x_0} \in E_J$ such that $f(\vec{x_0}) = 0$ and pick $g \in J$ such that $g(\vec{x_0}) \neq 0$. Let

$$h(\vec{x}) = \begin{cases} 1 & if \quad g(\vec{x}) = 0 \\ 0 & if \quad g(\vec{x}) \neq 0 \end{cases}$$

Then $f \cdot h + g \in J$ and $(f \cdot h + g)(\vec{x})$ is non-zero iff either $f(\vec{x})$ or $g(\vec{x})$ is non-zero. This proves the claim.
Now want to show: $J = \{f : f(\vec{x}) = 0 \text{ for all } \vec{x} \notin E_J\} = J_{E_J}$. By definition of E_J, $J \subseteq J_{E_J}$.
Now suppose $k(\vec{x}) = 0$ for all $\vec{x} \notin E_J$. Then $k = k \cdot f_E$ so $k \in J$.

Q.E.D.

Definition 46 *Let J be an ideal of U_F^n. Then $A = U_F^n/J = \{f + J : f \in U_F^n\}$ where $f + J = g + J$ iff $f - g \in J$ iff $f(\vec{x}) = g(\vec{x})$ for all $\vec{x} \notin E_J$.*

U_F^n/J is a commutative ring, a vector space of dimension $2^n - |E_J|$.
Example When $E = \emptyset$ then $U_F^n/J \cong U_F^n$ because $J_E = \{0\}$.

Definition 47 $\Omega_F^n = \{U_F^n/J : J \text{ an ideal}\}$

Definition 48 *Let $f = \{f_n : n \geq 1\}$ be a family of Boolean functions. f is F-easy iff for all n, f_n is represented by a polynomial over F, of degree $\mathcal{O}(1)$.*

Examples

a. Negation $1 - x$

b. *Mod-p*$(x_1, \ldots, x_n) = (x_1 + \ldots + x_n)^{p-1}$ if $char(F) = p$

However

$$AND(x_1, \ldots, x_n) = \bigwedge_{i=1}^{n} = \prod_{i=1}^{n}$$

and has degree n and is definitely not F-easy.
Composition of F-easy functions is an F-easy function. If $\{f_n\}$ is computed by a family of constant depth circuits, using only F-easy gates (uniformly F-easy), then $\{f_n\}$ is F-easy.

Definition 49 *$\{f_n(x_1,\ldots,x_n)\}$ is nearly F-easy iff $\exists\lambda>0$ such that for all n and for all n-ary Boolean functions $g^1(x_1,\ldots,x_n),\ldots,g^n(x_1,\ldots,x_n)$ for all $0\leq l\leq n$ there exists a quotient algebra $A=U_F^n/J$ of dimension $>2^n-2^{n-l}$ such that $f_n(g^1,\ldots,g^n)$ can be written in A as a degree $\lambda\cdot l$ polynomial of $g^1,\ldots,g^n$.*

In other words, $\forall 0\leq l\leq n,\exists$ polynomial p of degree $\lambda\cdot l$ such that

$$p(g^1(\vec{x}),\ldots,g^n(\vec{x}))=f_n(g^1(\vec{x}),\ldots,g^n(\vec{x})) \text{ for } \ >2^n-2^{n-l} \text{ many values of } \vec{x}$$

Lemma 81 *Suppose $char(F)\neq 0$. Then OR is nearly F-easy, hence AND is nearly F-easy.*

Proof. Given $f(g^1(\vec{x}),\ldots,g^n(\vec{x}))=OR(g^1(\vec{x}),\ldots,g^n(\vec{x}))$. Let $0\leq l\leq n$. Need polynomial f_n^* of degree $\lambda\cdot l$ such that $f_n^*(g^1(\vec{x}),\ldots,g^n(\vec{x}))\neq f_n(g^1(\vec{x}),\ldots,g^n(\vec{x}))$ for at most 2^{n-l} many values of $\vec{x}$. Since then we can choose $E_J=\{\vec{x}: f_n^*(g^1(\vec{x}),\ldots,g^n(\vec{x}))\neq f_n(g^1(\vec{x}),\ldots,g^n(\vec{x}))\}$. $Char(F)=p$, a prime. $Z_p\subseteq F$. f_n^* will be a polynomial of $g^1,\ldots,g^n$ withcoefficients in Z_p. We shall choose $c_{i,j}\in Z_p$ and let

$$\begin{aligned} f_n^*(g^1(\vec{x}),\ldots,g^n(\vec{x})) &= \bigvee_{i=1}^{l}((\sum_{j=1}^{n}c_{i,j}g_j)^{p-1}) \\ &= 1-\left[\prod_{i=1}^{l}(1-(\sum_{j=1}^{n}c_{i,j}g_j)^{p-1})\right] \end{aligned}$$

Claim Fix $\vec{x}\in\{0,1\}^n$. If $c_{i,j}$ are chosen randomly from Z_p then

$$Pr[f_n^*(g^1(\vec{x}),\ldots,g^n(\vec{x}))\neq f_n(g^1(\vec{x}),\ldots,g^n(\vec{x}))]\leq(\frac{1}{p})^l$$

Proof case (1) $g^1(\vec{x})=0=g^2(\vec{x})=g^3(\vec{x})=\ldots=g^n(\vec{x})$ then $f_n^*(\vec{g})=0=f_n(\vec{g})$
case (2) Suppose $g^k(\vec{x})\neq 0$. Fix i. If $c_{i,j},j\neq k$ have also been fixed then there is exactly one "bad" choice for $c_{i,k}$ such that $(\sum_{j=1}^{n}c_{i,j}g_j)=0$. So

$$Pr[(\sum_{j=1}^{n}c_{i,j}g_j)=0]=\frac{1}{p}$$

Thus, since $c_{i,j}$ are chosen independently,

$$Pr[\forall i\leq l(\sum_{j=1}^{n}c_{i,j}g_j)=0]=(\frac{1}{p})^l$$

This proves the claim. From the claim it follows that there exist choices for $c_{i,j}$ so that $f_n^*(g^1(\vec{x}), \ldots, g^n(\vec{x})) \neq f_n(g^1(\vec{x}), \ldots, g^n(\vec{x}))$ for at most $(\frac{1}{p})^l \cdot 2^n$ many values of $\vec{x}$. The degree of f_n^* is $l \cdot (p-1)$ and

$$\begin{aligned} (\frac{1}{p})^l \cdot 2^n &\leq 2^n \cdot 2^{-l} \\ &= 2^{n-l} \end{aligned}$$

In fact, $(\frac{1}{p})^l \cdot 2^n = 2^{n-l \cdot \log p}$.

Q.E.D.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: April 16 & 21, 1992 **Topic: Smolensky's theorem**
Instructor: Sam Buss **Notes by: Goran Gogić**

Lemma 82 *Let C_n be a family of circuits of depth k, such that C_n consists of 2^r many nearly F-easy gates, where $r(n) = o(n^{\frac{1}{2k}})$ and arbitrarily many F-easy gates. Then there is some $A \in \Omega_F^n$ of dimension $2^n - o(2^n)$ such that each output of C_n computes a function of degree $o(\sqrt{n})$ in A. In other words, for each output $h \in C_n$, there exists an $\tilde{h}$ of degree $o(\sqrt{n})$ and $2^n - o(2^n)$ values of $(x_1, x_2, ..., x_n)$ where $h(x_1, x_2, ..., x_n) = \tilde{h}(x_1, x_2, ..., x_n)$.*

Proof. Let $l = 2r$. Consider a nearly F-easy gate $f \in C_n$. It computes $f(g^1, g^2, ..., g^n)$ where $g^1, g^2, ..., g^n$ are its inputs. Pick $\tilde{f}$ of degree λl such that $f(g^1, g^2, ..., g^n) = \tilde{f}(g^1, g^2, ..., g^n)$ in some U_F^n/I_F of dimension $2^n - 2^{n-l}$ ($f(g^1, g^2, ..., g^n) \neq \tilde{f}(g^1, g^2, ..., g^n)$ for $\leq 2^{n-l}$ values of $(x_{1,2}, ..., x_n)$). Replace gate f by $\tilde{f}$. Note that each F-easy gate computes a function of degree at most λ. So, as the result we have a depth k circuit where each gate computes a function of degree at most $\lambda l = 2r\lambda$. Circuits outputs are of degree $\leq (2r\lambda)^k = (2\lambda o(n^{\frac{1}{2k}}))^k = o(\sqrt{n})$. The new circuit and the original circuit output different values for at most $2^r \times 2^{n-l} = 2^{n-r} = o(2^n)$ because there are 2^n nearly F-easy gates and 2^{n-l} $\vec{x}$'s where $f(g^1, g^2, ..., g^n) \neq \tilde{f}(g^1, g^2, ..., g^n)$. Without loss of generality we suppose that $n \to \infty$.

Q.E.D.

Definition 50 *Let $A \in \Omega_F^n$ and $g \in U_F^n$. Then degree of g in A is*

$$\deg_A(g) = \min_{h \in U_F^n} \{\deg(h) : g(\vec{x}) = h(\vec{x}) \, when \, \vec{x} \in A\}$$

.

Definition 51 *Let $\{f_n^1, f_n^2, ..., f_n^s\}$ be a family of functions. We say $\{f_n^1, f_n^2, ..., f_n^s\}$ is U_F^n-complete iff for any quotient algebra A and for any $g \in U_F^n$,*

$$\deg_A(g) \leq \frac{n}{2} + \max_{1 \leq i \leq n} \{\deg(f_n^i)\}$$

.

Lemma 83 *If $\{f_n^1, f_n^2, ..., f_n^s\}$ is U_F^n-complete and if $A \in \Omega_F^n$ and*

$$\max_{1 \leq i \leq n} \{\deg(f_n^i)\} = o(\sqrt{n})$$

then dimension of A is at most $2^{n-1} + o(2^n) = 2^{n-1}(1 + o(1))$.

Proof. Suppose $g \in A \in \Omega_F^n$, then by the definition of U_F^n-completeness,

$$\deg_A(g) \leq \frac{n}{2} + \max_{1 \leq i \leq n} \{\deg(f_n^i)\} = \frac{n}{2} + o(\sqrt{n})$$

So, all functions in A are expressible as linear combination of monomials of degree at most $\frac{n}{2} + o(\sqrt{n})$. So, dimension of A is less than or equal to the number of such monomials of degree $\leq \frac{n}{2} + o(\sqrt{n})$ which is equal to the number of subsets of $\{x_1, ..., x_n\}$ of cardinality at most $\frac{n}{2} + o(\sqrt{n})$

$$= \sum_{j=0}^{\frac{n}{2}+o(\sqrt{n})} \binom{n}{j} = 2^{n-1} + \sum_{j=\frac{n}{2}}^{\frac{n}{2}+o(\sqrt{n})} \binom{n}{j}$$

$$\leq 2^{n-1} + o(\sqrt{n}) \frac{\sqrt{2\pi n}(\frac{n}{e})^n}{(\sqrt{2\pi \frac{n}{2}}(\frac{n}{2e})^{\frac{n}{2}})^2} = 2^{n-1} + o(\sqrt{n}) \frac{\sqrt{2\pi n}}{\pi n (\frac{1}{2})^n}$$

$$\leq 2^{n-1} + o(\sqrt{n}) \frac{1}{\sqrt{n}} \sqrt{\frac{2}{\pi}} 2^n = 2^{n-1} + o(2^n).$$

Q.E.D.

Corollary 84 *If $f_n^1, f_n^2, ..., f_n^{s(n)}$ are U_F^n-complete, then there is no family of depth k circuits consisting of $2^{o(n^{\frac{1}{2k}})}$ nearly F-easy gates, and arbitrarily many F-easy gates computing $f_n^1, f_n^2, ..., f_n^s$.*

Proof. Obvious, by combining Lemma 82 and Lemma 83. We will soon prove a more general theorem.

Q.E.D.

Lemma 85 *Let $h \in F\backslash\{0,1\}$. Define*

$$Y_i(X_1, X_2, ..., X_n) = \begin{cases} h, & X_i = 1 \\ 1, & X_i = 0 \end{cases}$$

So, $Y_i = (h-1)X_i + 1$. Then, $\prod_{i=1}^n Y_i$ is U_F^n-complete.

Proof.

Note that h^{-1} and $(h-1)^{-1}$ exist because $h \notin \{0,1\}$. It is easy to see that $\frac{1}{Y_i} = (\frac{1}{h}-1)X_i+1$, $X_i = \frac{1}{h-1}(Y_i - 1)$. Let $A \in \Omega_F^n$ and $g \in A$. So, g can be written as a polynomial in Y_i's since $X_i = (h-1)^{-1}(Y_i - 1)$. So, g is a linear combination of monomials of the form $\prod_{i\in\omega} Y_i$ where $\omega \subseteq \{1, 2, ..., n\}$. It suffices to show that

$$\deg_A(\prod_{i\in\omega} Y_i) \leq \frac{n}{2} + \deg_A(\prod_{i=1}^n Y_i)$$

If $|A| \leq \frac{n}{2}$ it is obvious. If $|A| > \frac{n}{2}$ then

$$\prod_{i\in\omega} Y_i = (\prod_{i=1}^n Y_i)(\prod_{i\notin\omega} \frac{1}{Y_i}) \leq \deg_A(\prod_{i=1}^n Y_i) + \frac{n}{2}$$

since $n - |A| \leq \frac{n}{2}$.

Q.E.D.

Homework 2 *Is $\{Y_1, ..., Y_n\}$ U_F^n-complete for $h \in F \setminus \{0,1\}$?*

Homework 3 *Is $\prod_{i=1}^n X_i$ U_F^n-complete?*

Definition 52 *Let q be an integer. Then $h \in F$ is called a q-th root of unity iff $h^q = 1$.*

Definition 53 $MOD_{i,q} = \begin{cases} 1, & \text{iff } \sum_{j=1}^n X_j = i \bmod q \\ 0, & \text{otherwise} \end{cases}$

Proposition 86 *If F has a q-th root of unity different of 1 then $\{MOD_{0,q}, ..., MOD_{q-1,q}\}$ is U_F^n-complete.*

Proof. Let h be a q-th root of unity, $h \neq 1$. Let $Y_i = (h-1)X_i + 1$. Then

$$\prod_{i=1}^{n} Y_i(x_1, ..., x_n) = h^k = h^s$$

where $k = \sum_{i=1}^{n} X_i$ and $s = (k \bmod q)$. So, $\prod_{i=1}^{n} Y_i = \sum_{s=0}^{n-1} h^s \times MOD_{s,q}$. Thus, for any $A \in \Omega_F^n$ $\deg_A(\prod_{i=1}^{n} Y_i) \leq \max_s \deg_A(MOD_{s,q})$. So, by previous lemma and definition of U_F^n completeness, the set $\{MOD_{0,q}, ..., MOD_{q-1,q}\}$ is U_F^n-complete.

Q.E.D.

Theorem 87 *Suppose C_n is a family of circuits of depth k, comprising of $2^{o(n^{\frac{1}{2k}})}$ nearly F-easy gates, and arbitrarily many F-easy gates. Let $g_n^1, ..., g_n^{s(n)}$ be the output of C_n. Also suppose that $\{f_n^1, ..., f_n^{s(n)}\}$ are U_F^n complete. Then, the outputs of C_n are equal to the f_n values on at most $2^{n-1} + o(2^n)$ many inputs. So, the circuit can compute f_n only slightly more than one half of the time.*

Proof. Since $g_n^1, ..., g_n^{s(n)}$ are computed by C_n, by Lemma 82, there is an algebra $A \in \Omega_F^n$ of dimension $2^n - o(2^n)$ such that each function $g_n^1, ..., g_n^{s(n)}$ has degree $o(\sqrt{n})$ in A. Let $p_n^1, ..., p_n^{s(n)}$ be polynomials of degree $o(\sqrt{n})$ which are equal to $g_n^1, ..., g_n^{s(n)}$ in A. Let $\tilde{A} \in \Omega_F^n$ be the algebra where $p_n^1, ..., p_n^{s(n)}$ are computing $\{f_n^1, ..., f_n^{s(n)}\}$. By Lemma 83, $\tilde{A}$ has dimension at most $2^{n-1} + o(2^n)$. So, $g_n^1, ..., g_n^{s(n)}$ are equal to $\{f_n^1, ..., f_n^{s(n)}\}$ for at most $2^{n-1} + o(2^n)$ many x values.

Q.E.D.

Corollary 88 (Yao-Hastad) *Parity cannot be computed by depth k AND/OR/NOT circuits of size $2^{o(n^{\frac{1}{2k}})}$.*

Corollary 89 (Cai) *Any formula of depth k circuits of size $2^{o(n^{\frac{1}{2k}})}$ correctly computes Parity (for inputs of length n) for $\leq 2^{n-1} + o(2^n)$ many inputs.*

Proof. Take $F = \mathbf{Z}_3 = \{0, 1, 2\}$ and notice that 2 is a primitive square of unity. So, $\{\text{Parity}, \text{Not(Parity)}\}$ is U_F^n complete. Recall that AND/OR are nearly F-easy, and now we can apply the previous lemma for $C_n = \{\text{Parity}, \text{Not(Parity)}\}$ which gives the claimed result.

Q.E.D.

We can see that the corollaries hold if arbitrarily MODs gates are allowed because those gates are F-easy.

Corollary 90 *Let p and q be distinct primes. Then, there is no family of depth k circuits comprising of $2^{o(n^{\frac{1}{2k}})}$ $AND/OR/MOD_p$ gates and arbitrarily many NOT gates that compute MOD_q.*

Proof. Let F be a field of characteristic p with a q-th root of unity (different of 1). We can get F by starting with $\mathbf{Z}_p$ (which has characteristic p) and adjoining all the roots of the equation $x^q - 1 = 0$. Now, MOD_p and NOT are F-easy, AND/OR are nearly F-easy and $\{MOD_{0,q}, ..., MOD_{q-1,q}\}$ is U_F^n-complete, and we can apply the theorem, which gives that $\{MOD_{0,q}, ..., MOD_{q-1,q}\}$ does not have such circuits. Hence, MOD_q does not have such circuits because

$$MOD_{k,q}(x_1, .., x_n) = MOD_q(x_1, ..., x_n, \overbrace{1, ..., 1}^{n-k})$$

which means that $\{MOD_{0,q}, ..., MOD_{q-1,q}\} \leq_{proj} MOD_q$.

Q.E.D.

Results can be made stronger by fixing the exponent in the number of gates: there exists a function $s(n) = 2^{o(n^{\frac{1}{2k}})}$ such that for every n there are no circuits of constant depth with at most $s(n)$ nearly F-easy gates.

Homework 4 *Suppose that $\{f_n\}$ is a family of functions which are not computable by circuits of depth k of at most $2^{o(n^{\frac{1}{2k}})}$ many nearly F-easy gates for any k. Further suppose $\{f_n\} \leq_{cd} \{g_n\}$. Prove that there exists a constant r such that it is not the case that $\forall n$ g_n has circuits of depth k with $\leq 2^{o(n^{\frac{1}{rk}})}$ nearly F-easy gates.*

Corollary 91 *Depth k circuits for Majority require $2^{o(n^{\frac{1}{rk}})}$ nearlu F-easy gates for some constant r.*

Homework 5 (Smolensky-Barrington) *Prove that depth k circuits for Majority require $2^{\Omega(n^{\frac{1}{2k}})}$ nearly F-easy gates.*

Open Problem 1 *Is $MOD_5 \leq_{cd} MOD_6$ (note that $MOD_6 \leq_{cd-tt} \{MOD_2, MOD_3\}$? Is $NP \leq_{cd} MOD_6$?*

Theorem 92 (Cai-Babai) *$PH^A \neq PSPACE_A$ relative to a random oracle A.*

A "random" oracle A is chosen by independently letting $w \in A$ or $w \notin A$ with a probability (for all $w \in \{0,1\}$). Let us define a measure μ as the space $\{0,1\}$ with $\mu(\{0\}) = \mu(\{1\}) = \frac{1}{2}$. It makes sense to say $\Pr[PH^A = PSPACE^A] = 0$. Oracles form a probability measure space.

Proof. Fix a TM M^ϕ with an unspecified oracle ϕ, constrained to make $\mathcal{O}(1)$-alternations and run in polinomial time.

Claim 93 $\Pr[L(M^A) = Parity^A] = 0$ *where* $Parity^A = \{x : \exists$ *odd number of* $w \in A$ *of length* $|x|\}$.

Recall from earlier lectures: If M^ϕ makes k-alternations, then M^ϕ can be transformed into a family of depth $k+1$ circuits with literals of the form $w \in A$ or $w \notin A$ and of size $2^{n^{o(1)}}$. Can the μ-circuit compute Parity$\{x_w : |w| = n\}$? Fix all truth values of x_w where $|w| \neq n$, then randomly chose truth values for x_w, $|w| = n$. Then, $Pr[$circuit correctly computes Parity of x_w, $|w| = n] \leq \frac{1}{2} + o(1) < \frac{2}{3}$ for n large enough. Choosing $n_1 \ll n_2 \ll ...$, oracle A can be randomly chosen by deciding if $w \in A$ for w in input to n_1-th circuit; then if $w \in A$ for $|w| \leq n_2$ ... unless $w \in A$ is already decided, etc. $\Pr[M^A$ correctly computes $\text{Parity}_{n_i}] < \frac{2}{3}$ (independently of rest $n_1, n_2, ...$). So $\Pr[M^A$ correctly computes Parity of inputs of the $n_1, n_2, ...] = 0$. Thus, the claim is proven.

$$\begin{aligned} Pr[PH^A = PSPACE^A] &\leq Pr[\exists M^\phi \in PH \text{ such that } M^A = \text{Parity}^A] \\ &\leq \sum_{i=1}^{\infty} Pr[M_i^A = \text{Parity}] \\ &= \sum_{i=1}^{\infty} 0 = 0 \end{aligned}$$

where M_i is the i-th oracle TM in PH.

Q.E.D.

Open Problem 2 *Does the Polynomial-Time Hierarchy collapse at a random oracle, i.e.*

$$Pr[PH^A \, collapses] = 0 \, or \, 1 \, ?$$

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: April 23, 1992 **Topic: Probabilistic circuits**
Instructor: Sam Buss **Notes by: Goran Gogić**

References

1. E.Allender, A Note on the Power of Threshold Circuits, 30^{th} FOCS 1989, 580-584
2. S.Toda, On the computational power of PP and $\oplus P$, 30^{th} FOCS 1989, 514-519
3. L.Valiant, V.Vazirani, NP is as easy as detecting unique solutions, Theoretical Computer Science, 47 (1986) 85-93
4. E.Allender, U. Hertrampf, On the power of uniform families of constant depth threshold circuits, Mathematical Foundations of Computer Science 1990

Definition 54 *A probabilistic circuit is a circuit C with inputs $x_1, x_2, ..., x_n$ and with probabilistic inputs $b_1, ..., b_s$. If $f : \{0,1\}^n \to \{0,1\}$ then $C(x_1, ..., x_n, b_1, ..., b_n)$ computes $f(x)$ with error $\leq \varepsilon$ iff, for all $\vec{x}$,*

$$Pr[C(x_1, ..., x_n, b_1, ..., b_n) = f(x)] \geq 1 - \varepsilon$$

where the pobability is taken over all choices for values $b_1, b_2, ..., b_n$.

Lemma 94 (Allender) *Let us fix $k \geq 1$ and let OR_n be an n-input OR gate. Then OR_n can be computed by a probabilistic circuit of depth 2 consisting of a Parity of $n^{\mathcal{O}(\log n)}$ AND gates of fan-in $O(\log n)$ with error $\leq n^{-k}$ and uses $\mathcal{O}(kn \log n)$ probabilistic bits.*

Proof. Let B_n be the circuit which computes Parity$(1, x_1 \wedge b_1, ..., x_n \wedge b_n)$. When $x_1 = x_2 = ... = x_n = 0$ then B_n outputs 1 independently of the choice of the b's. If some $x_i \neq 0$ then B_n outputs 0 with probability $\frac{1}{2}$ and 1 otherwise. Now make $k \log n$ copies of B_n with new probability bits and form the AND of all. This circuit computes NOR with error $(\frac{1}{2})^{k \log n} = \frac{1}{n^k}$. Since if $x_1 = x_2 = ... = x_n = 0$ then every copy of B_n outputs 1 so the AND also outputs 1. Otherwise, B_n outputs 0 with probability $\frac{1}{2}$. So, AND outputs 0

with probability $(\frac{1}{2})^{k \log n}$. Use the distributive law to rewrite this as a parity of $(n+1)k \log n$ $ANDs$ of fan-in $\leq 2k \log n$. Invert the output of the circuit by omitting the 1 input to the Parity gate.

Q.E.D.

Corollary 95 *The same thing holds for AND.*

Lemma 96 *Fix $d \geq 0$. Let C_n be a family of AC^0 circuits of depth d and size n^l. Then there are probabilistic circuits of depth 2 consisting of a Parity of $n^{\mathcal{O}\left((\log n)^d\right)}$ many ANDs of fan-in $\mathcal{O}\left((\log n)^d\right)$ which use $\mathcal{O}\left(n(d+l)\log n\right)$ probabilistic inputs and approximate C_n with error $\leq \frac{1}{n^c}$.*

Remark 6: A similar theorem holds for C_n of size $2^{(\log n)^{\mathcal{O}(1)}}$. In this case d has to be increased.

Proof. Replace each OR and AND gate in C_n by a depth 2 probabilistic circuit which has Parity of $n^{\mathcal{O}(\log n)}$ ANDs of fan-in $\mathcal{O}(\log n)$ which aproximates the gate with error $\leq \frac{1}{n^{d+l}}$ and uses $n(d+l)\log n$ many probabilistic inputs. Use the same probabilistic inputs for each probabilistic circuits. After these replacements, the probability of error $\leq \frac{1}{n^{d+l}} n^l = \frac{1}{n^d}$. Finaly, use the distributive law to rewrite this as a parity of AND's. Use induction on d to argue that the circuit has the desired size. For depth $d = 1$ we are done already. For $d > 1$ suppose C_n has an AND of some number of at most n^l inputs, each computed in depth $d = 1$. This becomes a parity of $(n^{\mathcal{O}\left((\log n)^{d-1}\right)})^{\log n} = n^{\mathcal{O}\left((\log n)^d\right)}$ ANDs of fan-in $\mathcal{O}\left((\log n)^{d-1}\right)\mathcal{O}(\log n) = \mathcal{O}\left((\log n)^d\right)$.

Q.E.D.

Next Goal: Reduce the number of probabilistic inputs to $(\log n)^{\mathcal{O}(1)}$.

Definition 55 *Let $\mathbf{Z}_2 = \{0,1\}$ and for the vectors $v, w \in \mathbf{Z}_2^n$ let us define the inner product $\cdot$: $v \cdot w = \sum_{i=1}^n v^i w^i$ where $v = (v^1, ..., v^n), w = (w^1, ..., w^n)$.*

Theorem 97 *[Valiant-Vazirani] Let $S \neq \emptyset$, $S \subseteq \{0,1\}^n$. Chose randomly some vectors $w_1, ..., w_n \in \{0,1\}^n$ (n^2 many bits). Let $S_0 = S$ and $S_i = \{v \in S : v \cdot w_1 = v \cdot w_2 = ... = v \cdot w_i = 0\}$. Obviously, $S_0 \supseteq S_1 \supseteq ... \supseteq S_n$. Then, $\exists i$ such that $|S_i| = 1$ with probability $\geq \frac{1}{4}$.*

Intuition: $|S_{i+1}| \approx \frac{1}{2}|S_i|$.

Proof. We will show:

Claim 98 *$Pr[\exists i : |S_i| = 1 \mid w_1, ..., w_n$ are linearly independent $] \geq \frac{1}{2}$.*

Claim 99 *a) $Pr[w_1, ..., w_n$ are linearly independent $] \geq \frac{1}{4}$*

b) $Pr[w_1, ..., w_{n-1}$ are linearly independent $] \geq \frac{1}{2}$.

Claim 98 and Claim 99.a almost prove the theorem, giving the probability that $|S_i| = 1$ for some i is $\geq \frac{1}{2} \cdot \frac{1}{4} = \frac{1}{8}$.

To get the probability $\geq \frac{1}{4}$ argue two cases as follows:

Case 1: $0^n \in S$. By Claim 99.a,

$$\begin{aligned} Pr[\exists i : |S_i| = 1] &\geq Pr[w_1, ..., w_n \text{are linearly independent} \\ &\geq \frac{1}{4} \end{aligned}$$

Case 2: $0^n \notin S$.

$$\begin{aligned} Pr[\exists i : |S_i| = 1] &\geq Pr[\exists i : |S_i| = 1 \mid w_1, ..., w_{n-1} \text{are linearly independent} \\ &\quad \cdot Pr[w_1, ..., w_{n-1} \text{ are linearly independent} \\ &\geq \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4} \end{aligned}$$

So the theorem follows from Claim 98 and Claim 99.

Proof. of Claim 99:

$Pr[w_1, ..., w_i$ are linearly indep. $|w_1, ..., w_{i-1}$ are linearly indep. $]$

$= Pr[w_i$ is linearly indep. of $w_1, ..., w_{i-1} \mid$ given $w_1, ..., w_{i-1}$ are linearly indep. $]$

$= 1 - \frac{2^{i-1}}{2^n}$

for all i such that $0 \leq i \leq n$, since $i - 1$ independent wectors $w_1, ..., w_{i-1}$ span exactly 2^{i-1} vectors.

$Pr[w_1, ..., w_n$ are linearly independent$] = (1 - \frac{1}{2^n})(1 - \frac{1}{2^{n-1}}) \cdots (1 - \frac{1}{8})(1 - \frac{1}{4})(1 - \frac{1}{2}) = (1 - \frac{1}{2^n}) \cdots (1 - \frac{1}{4}) \cdot \frac{1}{2} \geq [1 - (\frac{1}{2^n} + ... + \frac{1}{4})] \cdot \frac{1}{2} \geq \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$

This proves Claim 99.a, Claim 99.b is similar.

Q.E.D.

So, it remains to prove Claim 98 in order to finish the proof of the Theorem.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 28 April 1992 **Topic: Proof of Valiant-Vazirani Theorem**
Instructor: Sam Buss **Notes by: Jinghui Yang**

We continue to talk about the proof of the Valiant-Vazirani's theorem.

If $S \neq \phi$, $S \subseteq \{0,1\}^n$, and if $w_1, ..., w_n \in \{0,1\}^n$ are randomly chosen and if $S_0 = S$,

$$S_i = \{v \in S : v \cdot w_1 = v \cdot w_2 = v \cdot w_2 = ... = v \cdot w_i = 0\}$$

then $Prob[\exists i | S_i|] \geq \frac{1}{4}$.

So far, we have proved that

$$Prob[w_1, w_2, ..., w_n \textit{ are linearly independent}] \geq \frac{1}{4}$$

and

$$Prob[w_1, ..., w_{n-1} \textit{ are linearly independent}] \geq \frac{1}{2}.$$

For $0 \in S$, if $w_1, w_2, ..., w_n$ are linearly independent, then $|S_n| = 1$, so we are done if $0 \in S$.

Now we consider the case that $0^n \notin S$. We have

$$\begin{aligned} Prob[\exists i | S_i| = 1 \quad : \quad & w_1, ..., w_{n-1} \textit{ are linearly independent}] \\ & = Prob[\exists i | S_i| = 1 : w_1, ..., w_n \textit{are linearly independent}]. \end{aligned}$$

To prove the Valiant- Vazirani theorem, it suffices to prove the following claim.

Claim 100 *Assume that $S \neq \phi$,*

$$Prob[\exists i | S_i| = 1 : w_1, ..., w_n \textit{ linearly independent}] \geq \frac{1}{2}.$$

Proof.

By induction on the rank of s: $k = r(s)$, where $r(s)$ is the dimension of the smallest subspace containing S and by subinduction on n.

Let

$$P(S) = Prob[\exists i | S_i| = 1],$$

and let E be the event that there is some i such that $|S_i| = 1$.

Base case (1): $k = 0$, so $S = \{0^n\}$, and $P(S) = 1$.

Base case (2): $k = 1$, so $S = \{0^n, v\}$ or $S = \{v\}$.
In the second subcase, $S_0 = S$ and S has cardinality one, hence $P(S) = 1$.
In the first case, $S_n = \{0^n\}$ which has cardinality one, $P(S) = 1$.

We'll prove by induction that

$$P(S) \geq \frac{2^{k-1}}{2^k - 1} \ \ (k \geq 1).$$

Induction step: Assume that S has rank $k > 1$,i.e., S has k elements that span the vector space containing S.

We will make a change of basis such that we may assume that these k elements are $e_1,\ e_2,\ ...,\ e_k$ where $e_j = 0^{j-1} \cdot 1 \cdot 0^{n-j}$. Note that this doesn't change the probability distribution since each vector still has probability $\frac{1}{2^n}$ of being chosen.

We will think $S \subseteq Z^n$ as a set of row vectors. Thus each $v \in S$, can be written as $v = (v^1,\ v^2,\ ...,\ v^n)\ with\ v^i \in Z_2$. Similarly, let w be a column vetor, define

$$H_w = \{v : v \cdot w = 0\}$$

where $v \cdot w = \sum_j v^j \cdot w_j\ (mod\ 2)$.

Given $v_1,\ ...,\ v_k \in S$ where $v_1, v_2,\ ...,\ v_k$ are linearly independent with $k = rank(S)$. Find $v_{k+1},\ ...,\ v_n$ such that $v_1,\ ..., v_k,\ v_{k+1},\ ...,\ v_n$ are linearly independent in Z_n. Let $e_1,\ e_2,\ ...,\ e_n$ are the unit vectors in Z_n and so any element x can be expressed as

$$x = x^1 e_1 + x^2 e_2 + ... + x^n e_n.$$

There is a matrix M such that if

$$(x^1, \ ..., \ x^n)M = (y^1, \ ..., \ y^n),$$

then

$$x = y^1 v_1 + ... + y^n v_n.$$

To change the basis, we multiply row row vectors on the right by M and multiply column vectors on left by M^{-1}. So

$$v \cdot w = (vM)(M^{-1}w).$$

Suppose $e_1, \ ..., \ e_k \in S$ where $k = rank(S)$. Given w, $H_w = \{v : v \cdot w = 0\}$.
(1). If $H_w \cap S = \phi$, then $w \in 1^k\{0,1\}^{n-k}$ (column vector).

Proof. This is because $e_j \cdot w \neq 0$ for $j = 1, \ ..., \ k$.

Q.E.D.

(2). $rank(H_w \cap S) < rank(S)$ iff $w \notin 0^k\{0,1\}^{n-k}$.

Proof. " $\Rightarrow$ " If $w \in 0^k\{0,1\}^{n-k}$, then $S \subseteq H_w$ and hence $H_w \cap S = S$.
" $\Leftarrow$ " If $w \notin 0^k\{0,1\}^{n-k}$, let $l_1, < l_2 < ... < l_j \leq k$ be the positions where w has 1's (in the first k positions).
Claim: The following $k-1$ vectors span $H_w \cap S$.
(a). e_i such that $i \in \{1, \ ..., \ k\} \setminus \{l_1, \ ..., \ l_j\}$
(b). $e_i - e_{l_i+1}$, for $i = 1, ..., j-1$.
So $rank(H_w \cap S) < k = rank(S)$.

Q.E.D.

Homework: Prove the claim.

Let E denote the event that there exists an i such that $|S_i| = 1$. Let $S_1 = H_w \cap S$. We have

$$\begin{aligned} Prob[E] \ &= \ Prob[E \mid rank(S_1) = rank(S)] \cdot Prob[\ rank(S_1) = rank(S)] \\ &\quad + Prob[E \mid rank(S_1) < rank(S)] \cdot Prob[\ rank(S_1) < rank(S)\ ] \end{aligned}$$

by our subinduction on n,

$$Prob[E|rank(S_1) = rank(S)] > \frac{2^{k-1}}{2^k - 1}$$

So it suffices to show that

$$Prob[E \mid rank(S_1) < rank(S)] \geq \frac{2^{k-1}}{2^k - 1}.$$

If $rank(S_1) < rank(S)$, then there are two possibilities:
(a). $rank(S_1) = 0$,i.e., $S = \phi$, E doesn't occur.
(b). $0 < rank(S_1) < rank(S)$. In this case, we apply the induction hypothesis to S_1 and get

$$Prob[E] \geq \frac{2^{k-2}}{2^{k-1} - 1}, \quad since\ rank(S_1) \leq k - 1.$$

The probability of case (a) occuring is:

$$\begin{aligned} Prob[S_1 = \phi|rank(S_1) < rank(S)] &\leq Prob[w_1 \in 1^k\{0,1\}^{n-k}|w_1 \notin 0^k\{0,1\}^{n-k}] \\ &\quad \text{(by facts 1. and 2.)} \\ &= \frac{2^{n-k}}{2^n - 2^{n-k}} \end{aligned}$$

So

$$\begin{aligned} Prob[E \mid rank(S_1) < rank(S)] &\geq \frac{2^{k-2}}{2^{k-1} - 1} \cdot (\frac{1 - 2^{n-k}}{2^n - 2^{n-k}}) \\ &= \frac{2^{k-2}}{2^{k-1} - 1}(\frac{2^n - 2 \cdot 2^{n-k}}{2^n - 2^{n-k}}) \\ &= \frac{2^{k-2}}{2^{k-1} - 1}(\frac{2^k - 2}{2^k - 1}) \\ &= \frac{2^{k-2} \cdot 2}{2^k - 1} \\ &= \frac{2^{k-1}}{2^k - 1} \end{aligned}$$

Q.E.D.

That completes the proof of Claim 100 and thus of the Vazirani-Valiant Theorem (Theorem 97).

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 5 May 1992 **Topic: Allender-Hertrampf theorem**
Instructor: Sam Buss **Notes by: Jinghui Yang**

Theorem 101 (Allender-Hertrampf) *Fix $k \geq 1$, $\{OR_n\}$ (n input OR's) can be computed by probabilistic circuits $\{B_n\}$ consisting of a PARITY of $n^{O(\log^2(n)\log\log n)}$ ANDs of fanin $O(\log^3 n)$ using $O(k\log^3 n)$ probabilistic inputs and having error $< \frac{1}{n^k}$.*

This improves earlier theorems. We proved (due to Allender) by reducing the number of probabilistic inputs from $O(kn\log n)$ to $O(k\log^3 n)$.

Proof.

Fix n, build B_n as follows: Inputs $x_1, \ldots, x_n$. Let $m = \lceil \log n \rceil$. For $l \in \{1, \ldots, n\}$, we code l as a string of m bits and hence we can think l as a row vector from $(Z_2)^m$).

Let $\{b_{i,j}\}_{i,\ j=1,\ \ldots,\ m}$ be the probabilistic inputs. We write

$$l \cdot b_i = \sum l^j b_{ij} \ mod\ 2$$

where l is coded by $l^1, \ldots, l^m$.

Idea: $S = \{l : x_l = 1\}$, we use Valiant-Vazirani's method to build $S_1, \ldots, S_m$ using $\bar{b}_1, \ldots, \bar{b}_m$.

(α) Build $D_{l,i}$ ($l \in \{1, \ldots, n\}$, $i \in \{1, \ldots, m\}$). $D_{l,i}$ outputs 1 iff

$$x_l = 1 \ and \ l \cdot \vec{b_1} = l \cdot \vec{b_2} = \ldots = l \cdot \vec{b_i} = 0$$

$l \cdot \vec{b_j}$ is computed by a PARITY of the form:

$$(b_{j_1} \wedge l^1) \oplus (b_{j_2} \wedge l^2) \oplus \ldots \oplus (b_{j_i} \wedge l^i)$$

where l^1, ..., l^i are constants, so by omitting l^k's that are 1s and omitting any $b_{jk} \wedge l^k$ where $l^k = 0$ we can get a PARITY of a subset of $b_{j,1}$, ..., $b_{j,i}$.

Now $D_{l,i}$ is an AND of $(m+1)$ Parity's of fanin $\leq m$, Rewrite $D_{l,i}$ as a PARITY of $m^{O(m)} = 2^{O(\log n \cdot \log\log n)}$ AND's of fanin $O(\log n)$.

(β). Fix i, let E_i be the PARITY of $D_{1,i}$, ..., $D_{n,i}$. E_i is a PARITY of AND's. In fact, a PARITY of $n \cdot 2^{O(\log n \log\log n)} = n^{O(\log\log n)}$ AND's of fanin $O(\log n)$.

E_i outputs 1 if an odd number of D_{1i}, ..., D_{ni} output 1. Let

$$S_i = \{l : l \in S \text{ and } l \cdot \vec{b_1} = l \cdot \vec{b_2} = ... = l \cdot \vec{b_i} = 0\}$$

$$S = \{l : x_l = 1\}$$

So E_i outputs 1 iff $|S_i|$ is odd, since $D_{l,i}$ outputs 1 iff $l \in S_i$. If all x_l's are zero, then $S = \phi$ and E_i outputs 0. Otherwise, E_i might output 1.

(γ). From Valiant-Vazirani's theorem, if $S \neq \phi$, $\exists i$ such that E_i outputs 1 with probability $\geq \frac{1}{4}$.
Let $\bar{E_i} = \neg E_i$ be obtained from E_i by letting an additional 1 be input x to the top PARITY gate.

Form a circuit F as the AND of $\bar{E_1}$, ..., $\bar{E_m}$. F will outputs 1 if $x_l = 0$ for all l and will output 0 otherwise with probability $\geq \frac{1}{4}$. Rewrite F as a PARITY of $n^{O(\log n \log\log n)}$ AND's of fanin $O(\log^2 n)$.

Form $3k \log n$ many copies of F with didstinct probabilistic inputs.

Form the AND of all of them and call this circuit $\bar{B_n}$. $\bar{B_n}$ will output 1 if all x_l's are 0, otherwise, $\bar{B_n}$ will output 0 with probability $\geq 1 - (\frac{3}{4})^{3k \log n}$. Since

$$(\frac{3}{4})^{3k \log n} = \frac{1}{n^{3k \log(\frac{3}{4})}} = \frac{1}{n^{k \log(\frac{64}{27})}} < \frac{1}{n^k},$$

$\bar{B_n}$ computes NOR with error $< \frac{1}{n^k}$.

Rewrite $\bar{B_n}$ as a PARITY of $n^{O(\log^n \log\log n)}$ AND's of fanin $O(\log^3 n)$ by putting another 1 input to the PARITY gate for the desired circuit B_n.

Q.E.D.

Corollary 102 (Allender-Hertrampf) *Fix k, let $\{C_n\}$ be a family of AC^0 circuits of depth d. Then C_n can be computed by probabilitic circuits of depth 2 consisting of PARITY of $n^{O(\log n)^{3d}}$ AND's of fanin $O((\log n)^{3d}$ using $O((\log n)^3)$ many probabilistic inputs and having error $\leq \frac{1}{n^k}$.*

Proof. Exactly like the proof of the similar theorem of Allende proved in an earlier lecture.

Q.E.D.

The same theorem holds for C_n having depth d and size $2^{(\log n)^{O(1)}}$.

How can we remove randomness? One way is to try all possible choices for the probabilistic inputs. E.g., form MAJORITY of all $2^{O(\log^3 n)}$ circuits obtained from all such choices. That is, a MAJORITY of $2^O(\log^3 n)$ PARITYs of $n^{O(\log n)^{3d})}$ AND's of fanin $O((\log n)^{3d})$.

We will discuss a better way to remove randomness in the next lecture.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: 7 May 1992 **Topic: Toda's Method**
Instructor: Sam Buss **Notes by: Jinghui Yang**

We will use Toda's method to get rid of randomness in previous theorems. We introduce Toda's polynomials:

Let

$$g(n) = 3n^4 + 4n^3$$
$$f_0(n) = n$$
$$f_{i+1} = g(f_i(n))$$

We have the following facts.

Lemma 103 *The degree of f_i is 4^i.*

Lemma 104 *For all $x \geq 2$, $x^{4i} \leq f_i(x) \leq x^{7i}$.*

Proof. $g(x) = 3x^4 + 4x^3 \leq 7x^4 \leq x^7$ for $x \geq 2$. The lemma is now easy to prove by induction.

Q.E.D.

Lemma 105 *Let $N \geq 2$,*

a. *If $x \equiv 0 \ modN$, then $g(x) \equiv 0 \ modN^2$.*
b. *If $x \equiv -1 \ modN$, then $g(x) \equiv -1 \ modN^2$.*
c. *If $x \equiv 0 \ modN$, then $f_i(x) \equiv 0 \ modN^{2^i}$.*
d. *If $x \equiv -1 \ modN$, then $f_i(x) \equiv -1 \ modN^{2^i}$.*

Proof. 1.

$$\begin{aligned} g(aN) &= 3(aN)^4 + 4(aN)^3 \\ &\equiv 0 \; mod \; N^2 \end{aligned}$$

2.

$$\begin{aligned} g(aN-1) &= 3((aN)^4 - 4(aN)^3 + 6(aN)^2 - 4(aN) + 1) \\ &\quad +4((aN)^3 + 3(aN)^2 + 3(aN) - 1) \\ &= 3(aN)^4 - 8(aN)^3 + 30(aN)^2 - 1 \\ &\equiv -1 \; modN^2 \end{aligned}$$

3. and 4. follow immediately from 1. and 2. by induction.

Q.E.D.

Remark: Toda's polynomials are also called modulus amplification polynomials.

Homework: Suppose we use $g(x) = 3x^2 - 2x^3$ instead of Toda's polynomial.

(a) Restate and prove the analogue of the above three lemmas use $x \equiv 1 \; modN$ in place of $x \equiv -1 \; modN$..

(b) Why wouldn't $x^2 - 2x$ work?

Recall that the Allender Hertrampf theorem converted any AC^0 circuit into a circuit which is a PARITY of $2^{(\log n)^{O(1)}}$ AND's of fanin $(\log n)^{O(1)}$ probabilistic bits.

Proposition Given q AND's of fanin $\leq r$, we can construct $g(q)$ many AND's of fanin $\leq 4r$ such that for all input values, if m of the original AND's have value true, then $g(m)$ of the constructed AND's have value True.

Proof.

Number the AND's from 1 to q. For each i, j, $k \in \{1, ..., q\}$, form the AND's of the i-th, j-th , k -th AND's. Make four copies of the AND's obtained this way. Now for each i, j, k,

$l \in \{1, ..., q\}$, form the AND's of the i-th , j-th, k-th and l-th AND's and make three copies of these AND's.

Q.E.D.

Corollary 106 *Given q AND's of fanin $\leq r$, we can construct $f_i(q)$ ANDs of fanin $\leq 4^i r$ such that for all inputs, if m of the original AND's have value True, then $f_i(m)$ of the constructed ANDs have value True.*

Given a depth 2 circuit consisting of PARITY of $2^{(\log n)^{O(1)}}$, AND's of fanin $(\log n)^{O(1)}$ using $(\log n)^a$ many probabilistic inputs (a is a constant) making error $< \frac{1}{2}$.

We do the following construction:

Step 1: Pick i such that $2^{2^i} > 2^{(\log n)^a}$, i.e., $i = a \cdot \lceil \log \log n \rceil$. Note that $2^{\lceil \log n \rceil^a}$ is the number of possible values for probabilistic inputs. 2^{2^i} will be the modulus after amplification.

Step 2: For each choice of probabilistic inputs, apply the corollary to the $2^{(\log n)^{O(1)}}$ AND's in the circuit obtained by fixing the probabilistic inputs. So we get

$$\begin{aligned} f_i(2^{(\log n)^{O(1)}}) &\leq (2^{(\log n)^{O(1)}})^{7i} \\ &= 2^{(\log n)^{O(1)}} \cdot 7^{\log \log n} \\ &= 2^{(\log n)^{O(1)}} \end{aligned}$$

many AND's of fanin $\leq 4^i \cdot (\log n)^{O(1)} = 4^{O(\log \log n)} (\log n)^{O(1)} = (\log n)^{O(1)}$.

For all choices of non-probabilistic inputs, the number of these AND's with value True will be congruent to 0 or -1 mod 2^{2^i}.

Step 3: Repeat step 2 for all $2^{(\log n)^{O(1)}}$ choices of probabilistic inputs. Then the result is a total of $2^{(\log n)^{O(1)}} \cdot 2^{(\log n)^{O(1)}} = 2^{\log n)^{O(1)}}$ many AND's.

Suppose for the probabilistic inputs and suppose that for M choices of the probabilistic inputs, the original "given" circuit had value True. Then the number of AND's constructed in step 3 which have True is congruent to $-M$ mod 2^{2^i}. Note here that $0 \leq M \leq 2^{(\log n)^{O(1)}} < 2^{2^i}$.

Step 4: Form the depth 2 circuit consisting of a symmetric function g of all the ANDs from step 3, such that g accepts (outputs True) if and only if the number of inputs with value True is congruent to $-M \mod 2^{2^i}$ with $\frac{2^{(\log n)^a}}{2} < M \leq 2^{(\log n)^a}$.

This circuit has no probabilistic inputs.

So we've actually proved the following theorem.

Theorem 107 *If $\{C_n\}$ is a family of AC_0 circuits, then $\{C_n\}$ is equivalent to a family of depth 2 circuits which consist of a symmetric gate of $2^{poly \log(n)}$ ANDs of fanin poly $\log n$.*

This theorem has been generalized as follows:

Theorem 108 (Yao, FOCS'90, Beigel-Tarui, FOCS'91) *If $\{C_n\}$ is a family of ACC circuits (or even a family of depth k, size $2^{(\log n)^{O(1)}}$ circuits of AND, OR, NOT, Mod m gates), then $\{C_n\}$ is equivalent to a family of depth 2 circuits which consist of a symmetric gate of $2^{poly \log(n)}$ ANDs of fanin poly $\log n$.*

Open Problem: Can Majority of Majorities be expressed as a family of depth 2 circuits consisiting of a symmetric function of $2^{poly \log(n)}$ ANDs of fanin *poly* $\log(n)$?

We now just state a result due to Toda. Let P be the class of predicates which are recognizable in polynomial time. Define

$$PP = \{Q : for\ some\ R(x,y) \in P\ and\ some\ polynomial\ p(n),$$
$$\forall x Q(x) \Leftrightarrow (the\ number\ of\ y < 2^{p(|x|)} R(x,y) > 2^{p(|x|)-1})\}.$$

PP is called "Probabilistic P".
Let P^{PP} denote the class of predicates recognizable in polynomial time with an oracle for a predicate in PP. Then Toda's result is

$$PH \subseteq P^{PP}.$$

If $Q \in PH$, Q is accepted by a depth k circuit of size $2^{n^{O(1)}}$ for inputs of length n.
Transform this to a depth 2 circuit consisting a symmetric gate g of $2^{n^{O(1)}}$ ANDs of fanin $N^{O(1)}$ by similar methods as used above.

Now whether a particular AND of fanin $N^{O(1)}$ has value True can be decided in polynomial time.
To decide if the depth 2 circuit has value True, we use a PP oracle and binary search to get the number of ANDs with value True. Then accept according to the symmetric gate g.

Homework*: Explain how to use $g(x) = 3x^2 - 2x^3$ for the circuit constructions. [The problem here is how to handle the minus sign].

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: May 12, 1992 **Topic: Uniform Circuit Families**
Instructor: Sam Buss **Notes by: Dave Robinson**

Reference: W. Ruzzo "On Uniform Circuit Complexity," JCSS 22, 1981 pp. 365–383.

The basic questions here are 1) how hard is it to generate a circuit family that computes a given function, and 2) how to make circuit families "uniform." Convention: circuits consist of unbounded fanin ORs, ANDs (and NOTs).

Suppose we are given a circuit family of size $S(n)$ and depth $D(n)$. We begin coding each circuit by numbering the gates such that:
1) the output gate is numbered 0
2) the input gates are numbered 1 through n
3) the rest have distinct numbers $\leq (S(n))^{O(1)}$

The standard representation of a circuit is:
$\{\langle g, t, g_L, g_R\rangle : g$ is a gate #, t is a gate type $\in \{\wedge, \vee, \neg, x\}$, g_L, g_R are gate #'s of the up to 2 inputs to $g\}$. (All the gate numbers are in binary.)

Definition: $\{C_n\}$ is a logspace-uniform family of circuits iff the map $n \rightarrow$ the std rep of $\{C_n\}$ is in logspace. (The output of this function is the concatenation of the above tuples.)

Definition: Let $\{C_n\}$be a family of circuits of size $S(n)$ and depth $D(n)$. Then $\{C_n\}$ is U_B-uniform iff $n \rightarrow$ the std rep of $\{C_n\}$ is in $SPACE(\, D(n)\,)$.

Definition: $\{C_n\}$ is U_{BC}-uniform iff $n \rightarrow$ std rep of $\{C_n\}$ is in $SPACE(\log(\, S(n)\,)$.

Note: "B" = Borodin and "C" = Cook.

U_B-uniform $NC^k = \log^k(n)$-space uniform.

U_{BC}-uniform $NC^k =$ logspace uniform.

The original definition of NC^k used U_{BC}-uniformity.

Definition: Let $\{C_n\}$ be a circuit family. The direct connection language for $\{C_n\}$ is: $L_{DC} = \{\langle n, g, p, g' \rangle : g$ is a gate #, $p \in \{L, R, \epsilon\}, g'$ is the gate # of the Left/Right input to g if $p = L$ or R (resp.), and g' is the gate-type of g if $p = \epsilon\}$. In this definition the value of n must be coded in binary, and the symbol ϵ denotes the empty string.

Definition: The extended connection language for $\{C_n\}$ is: $L_{EC} = \{\langle n, g, p, g' \rangle : g$ is a gate $\#, p \in \{L, R\}^*, |p| \leq \log(\, S(n)\,)$; if $p = \epsilon$, then g' is the gate type of g, otherwise it is the # of the gate reached by following the sequence of Left/Right inputs to the gate $\}$.

Theorem 109 *Let $Z(n) = \Omega(\log(S(n))$. Then the following are equivalent:*
(1) $n \rightarrow$ the std rep of $\{C_n\}$ is in $SPACE(Z(N))$
(2) $L_{DC} \in SPACE(Z(n))$
(3) $L_{EC} \in SPACE(Z(n))$

Proof: (1) $\Longleftrightarrow$ (2): L_{DC} and the std representation are essentially equivalent.
(3) $\Longrightarrow$ (2) is trivial.
(2) $\Longrightarrow$ (3): The obvious algorithm works. Given a candidate L_{EC} string follow the path using the L_{DC} algorithm and see if you end up at the correct gate.

Definition: let $\{C_n\}$ be a family of size $S(n)$ and depth $D(n)$ circuits, with $S(n) = \Omega(n)$.

1) $\{C_n\}$ is U_D-uniform iff L_{DC} is recognized by a DTM which on input $\langle n, g, p, g' \rangle$ uses $TIME(O(\log(S(n)))$. (This is almost equivalent to saying that $L_{DC} \in$ linear time because the length of the longest tuples for a given n are $O(\log(S(n)))$.)

2) $\{C_n\}$ is U_E-uniform iff L_{EC} is recognized by a DTM which on input $\langle n, g, p, g' \rangle$ runs in time $O(\log(S(n)))$.

3) $\{C_n\}$ is U_{E^*}-uniform iff L_{EC} if recognized by an ATM which on input $\langle n, g, p, g' \rangle$ runs simultaneously in $TIME(D(n))$ and $SPACE(\log(S(n))$

In circuits of fanin 2: $S(n) \leq 2^{D(n)}$, and $D(n) \geq \log(S(n))$. Thus if $\{C_n\}$ is U_E-uniform then $\{C_n\}$ is U_{E^*}-uniform , and for the same reason if $\{C_n\}$ is U_{BC}-uniform then $\{C_n\}$ is U_B-uniform. Obviously, if $\{C_n\}$ is U_E-uniform then it is U_D-uniform.

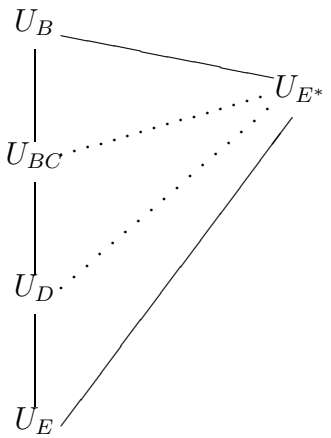
Theorem 110 *If $D(n) = \Omega(\log^2(S(n)))$ then if $\{C_n\}$ is a family of U_{BC}-uniform circuits then $\{C_n\}$ is also U_{E^*}-uniform .*

(Remark: the condition $D(n) \geq \log^2(S(n))$ is nontrivial. So e.g. the theorem applies to NC^k circuits for $k \geq 2$ but not to NC^1 circuits.)

Proof: $\{C_n\}$ is U_{BC}-uniform $\Rightarrow$ L_{EC} is in $SPACE(O(\log(S(n))))$ —by earlier—
$\Rightarrow$ $L_{EC} \in ATIME - SPACE(\log^2(S(n)), \log(S(n)))$ —proof: Savitch's Theorem—
$\Rightarrow$ L_{EC} is in $ATIME - SPACE(D(n), \log(S(n)))$.

Homework: Prove that if $D(n) = \Omega(\log(S(n)) \cdot \log\log(S(n)))$ then if $\{C_n\}$ is a U_D-uniform family of circuits then $\{C_n\}$ is also U_{E^*}-uniform .

So far we've proved:



The solid lines show inclusions among uniformities. The upper dotted line applies only for circuit depth $\geq \log^2 S(n)$, and the lower dotted line applies only for circuit depth $\geq \log S(n) \log\log S(n)$.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: May 14, 1992 **Topic: Uniform Circuit Families**
Instructor: Sam Buss **Notes by: Dave Robinson**

Theorem 111 *(Ruzzo): Let* $S(n) = n^{\Omega(1)}$, $D(n) \geq \Omega(\log n)$. *Suppose* $S(n)$ *and* $D(n)$ *can be computed from* n *in time* $O(\log(S(n)))$. *Suppose* L *is a language. Then if* L *has* U_{E^*}*-uniform circuits of size* $S(n)$ *and depth* $D(n)$ *then* L *has* U_E*-uniform circuits of size* $(S(n))^{O(1)}$ *and depth* $D(n)$.

Corollary 112 *Let* $S(n)$, $D(n)$ *be as above, and suppose* $D(n) = \Omega(\log^2 S(n))$. *Then if* L *has* U_{BC}*-uniform circuits of size* $S(n)$ *and depth* $D(n)$ *then* L *has* U_E*-uniform circuits of size* $(S(n))^{O(1)}$ *and depth* $D(n)$.

Corollary 113 *Let* $k \geq 1$. *Then:*
U_{E^*}*-uniform* $NC^k = U_E$*-uniform* NC^k.

Proof: Immediate.

Corollary 114 *Let* $k \geq 2$. *Then:*
U_{BC}*-uniform* $NC^k = U_D$*-uniform* $NC^k = U_{E^*}$*-uniform* $NC^k = U_E$*-uniform* NC^k.

Proof: for $k \geq 2$, U_{E^*}-uniform and U_E-uniform "sandwich" the others.

(Note: it is not known if Log^kspace is in NC so this is why U_B-uniform is not on the list.

The original definition of uniform NC^k is U_{BC} but better than this is U_E. They are the same for $k \geq 2$ but U_E is not "more powerful than the circuit class" for $k = 1$.

Lemma 115 *Let $S(n)$ and $D(n)$ be as in the theorem. (Actually the constructiblity part is not necessary for this lemma.) If L has U_{E^*}-uniform circuits of size $S(n)$ and depth $D(n)$, then $L \in$ A-SPACE-TIME$(D(n), \log(S(n)))$.*

Proof: Let's assume wlog that circuit has gates of type AND and OR, and its inputs are x_i or $\overline{x_i}$. The basic idea is to have the ATM simultaneously guess and simulate the circuit while in parallel verifying the guesses.

Algorithm for ATM

The algorithm will be a big loop, with the "loop variable" $\langle g, p \rangle$, where
g is a gate #
$p \in \{L, R\}^*$, $|p| < \log(S(n))$
$(g)p =$ the gate reached by path p from g

Initially we have $\langle 0, \epsilon \rangle$, ($0 =$ output gate, $\epsilon =$ the empty string.) If the gate type of $(g)p$ is OR (resp. AND) then branch existentially (resp. universally) to $(g)pL$ or $(g)pR$.
If $|pL|, |pR|$ is $\log(S(n))$ compute $h = (g)p$. Use $\langle h, \epsilon \rangle$ for the next iteration.
If $(g)p$ is input then accept or reject.

In more detail:
(1) Guess the gate type t of $(g)p$ [time $O(1)$]
Universally choose to:
(a) Verify t by
Guess $h = (g)p$ [time $\log(S(n))$]
Universally choose to
(i) verify h by checking if $\langle n, g, p, h \rangle \in L_{EC}$, accepting if so, rejecting otherwise.
(ii) verify t is the type of h by checking if $\langle n, h, \epsilon, t \rangle \in L_{EC}$
[(i) and (ii) together take time-space ($D(n)$,$\log(S(n))$)]
(b) continue with step (2)
(2) If t is AND or OR
then either existentially set $p = pL$ or pR or universally set $p = pL$ or pR respectively. [time $O(1)$]
If $|p| = \log(S(n))$, guess $h = (g)p$ and universally verify $\langle m, g, p, h \rangle \in L_{EC}$ and goto (1)
[time $O(\log(S(n))$ but this step is done only once every $\log(S(n))$ steps]
(3) Otherwise t is input type
Guess $h = (g)p$ [time $O(\log(S(n)))$ –only done once]
Universally
(a) verify $h = (g)p$ by checking if $\langle n, g, p, h \rangle \in L_{EC}$
(b) accept iff the input has value "true"

One can confirm straightforwardly that (1) this algorithm simulates the circuit, (2) the space used is $O(\log(S(n)))$, and (3) the time used is $O(D(n))$. QED

Lemma 116 *(converse): Let $S(n) = \Omega(\log n)$, $D(n) = \Omega(\log n)$ be constructible in time $\log n$. Then if $L \in$ ATIME-SPACE $(D(n), S(n))$ then L has U_E-uniform circuits of size $2^{O(S(n))}$ and depth $D(n)$.*

Corollary 117 *U_{E^*}-uniform $NC^k =$ ATIME-SPACE$(\log^k n, \log n)$.*

Proof of Corollary 117: U_{E^*}-uniform $NC^k \subseteq$ ATIME-SPACE$(\log^k n, \log n)$ by Lemma 115, and
ATIME-SPACE$(\log^k n, \log n) \subseteq U_E$-uniform NC^k by Lemma 116
$\subseteq$ U_{E^*}-uniform NC^k

So loosely speaking NC^k-uniform NC^k is U_E-uniform NC^k.

Proof of Lemma 116: The circuits for L are based on the execution tree of the ATM for L. The ATM M uses time $D(n)$ and space $S(n)$. So a configuration for M uses $O(S(n))$ symbols (on input of length n). Gate #s for the circuit for L are pairs $\langle t, c\rangle$, where t represents a time between 0 and $D(n)$, and c is a configuration of M. There are $D(n) \cdot (c')^{S(n)} \leq O(1)^{S(n)} \cdot O(1)^{S(n)} = 2^{O(S(n))}$ of these.

Inputs to $\langle t, c\rangle$ are $\langle t+1, c_L\rangle$ and $\langle t+1, c_R\rangle$, where c_L and c_R are successor configurations to c.

The type of $\langle t, c\rangle$ is AND or OR if c is a universal or existential (resp.) configuration, or is of type input if it's a halting configuration. W.l.o.g. we can look at the inputs only in halting configurations since inputs can be guessed and verified during the computation.

The depth is clearly $D(n)$. We need to check that $\langle n, g, p, h\rangle \in L_{EC}$ can be determined in time $O(\log 2^{O(S(n))}) = O(S(n))$. This can be done by simulating the ATM and following the path p through the computation tree.

Note: this circuit will have "useless gates," i.e., ones that are not connected to the output gate.

Homework: Give another proof of Lemma 116 that constructs a circuit without "useless gates."

Definition: SC = DTIME-SPACE(poly, polylog).

It is open whether SC = NC.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: May 19, 1992 **Topic: Uniform Circuit Families**
Instructor: Sam Buss **Notes by: Dave Robinson**

Papers related to the next topic:
Buss, STOC 1987, section 3.
Barrington, Immerman, Straubing "On Uniformity within NC^1," JCSS 41 (1990) 274-306.

Definition: A family $\{C_n\}$ of circuits is in uniform AC^0 iff the C_n are constant depth polynomial size unbounded fanin $AND/OR/NOT$ circuits and the L_{DC} can be recognized in deterministic time $O(\log n)$, where: $L_{DC} = \{\langle n, g, 0, g'\rangle : g' \text{ is input to } g\} \cup \{\langle n, g, 1, t\rangle : t$ is the gate type of $g\}$. (In both of these, n is given in binary notation, and g, g' are gate numbers in C_n.)

Equivalently L_{DC} is recognized in linear time in the length of its inputs.

Definition: A family $\{C_n\}$ of circuits is in *-uniform AC^0 iff the above holds except that L_{DC} is recognizable by an ATM in time $O(\log n)$ with constant alternations.

Equivalently, L_{DC} is in constant alternation linear time.

Definition: (Sipser) The logtime hierarchy (LH) is the set of predicates or languages L such that membership in L can be recognized in constant alternation logtime.

Convention: the ATM for sublinear time has an index tape that allows random access to the input tape. The index tape:

(a) has address written in binary,

(b) tape head position is unimportant during the query,

(c) has a special "input query" state.

Example: the following is a deterministic logtime algorithm for determining the length n of the input string. By convention, tape squares past the end of the the input string contain a special blank character.

(I) determine length of input to within a power of 2
(1) write 1 on the index tape
(2) query the input. If blank goto (4)
(3) append 0 to index tape contents and goto (2)
(4) erase last 0 and return tape head to the most significant 1
(II) do binary search to get actual length
(5) move to the right one square. If blank halt
(6) if it's a 0 then change it to a 1
(7) query the input. If it's a blank then change the 1 back to 0
(8) goto (5)

Aside: in deterministic logtime you can't do OR.

A desirable property (which is not satisfied above) is that input symbols only be queried in halting configurations. To this end we add six special halting states that accept/reject iff the input symbol being read is 1/0/blank.

Now given an ATM M that runs in time $c \log n$ and makes k alternations we can construct a new machine M′ with L(M′) = L(M) such that M′ makes all queries at halting states. M′ does:
(1) Guesses input length in binary. (Some computation paths which will necessarily be nonaccepting are longer than $O(\log n)$.)
(2) (wlog M starts in an existential state)
Do the following for a total of k blocks of alternations:
M′ guesses $c \log n$ bits existentially
then chooses $c \log n$ bits universally
then guesses $c \log n$ bits existentially
. . .
These guesses and choices will fully specify a computation of M. That is, every path in the computation tree of M will be represented by one of these sequences— but not vice versa. The i^{th} round of guesses and choices gives $c \log n$ bits which is enough to specify the (co)nondeterministic moves of M in its i^{th} block.
(3) (wlog k is odd, so we ended with existential choices in 2.)
Guess existentially the sequence of input symbols that will be queried by M on the computation using guesses and choices from (2).

(4) Choose universally $j \leq c \log n + 1$.

If $j = 0$ verify that the guess of the input length is correct.

If $1 \leq j \leq c \log n$, run M deterministically until its j^{th} query to the input and accept iff the guess for that symbol was correct. (Assuming that the guessed values for the first $j - 1$ queries were correct.)

If $j = c \log n + 1$, run M deterministically using the guessed input symbols. Accept iff M does.

M$'$ accepts in time $O(\log n)$ iff M accepts in time $c \log n$. M$'$ has $k+1$ blocks of alternations.

Homework: If we use the convention that the index tape head must be on the left symbol of the address, can a DTM find the length of its input in time $O(\log n)$? Or: can it find the length of its input to within a power of 2 in $O(\log n)$ time?

In summary, we will always make the assumption that a constant alternaton logtime Turing machine begins by guessing the length of its input and that the machine will reject if this guessed length is erroneous. In addition, constant alternation Turing machines will be presumed to access input symbols only in halting configurations.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: May 26 — June 2, 1992 **Topic: Uniform AC^0 Circuits**
Instructor: Sam Buss **Notes by: Elias Koutsoupias**

We are going to prove this theorem:

Theorem 118 . *The following are equivalent:*

- $L \in$ *uniform* AC^0
- $L \in$ **-uniform* AC^0
- $L \in LH$

The above theorem follows from the following two theorems.

Theorem 119 *If L is in LH then L has a family of uniform AC^0 circuits.*

Proof. [*of Theorem 2*] Let M be an ATM for L which makes k alternations and runs in time $\mathcal{O}(\log n)$. Without lost of generality, it operates by making $kc \log n$ universal/existential choices in alternating blocks of length $c \log n$ and queries input symbols only in halting configurations, as discussed in the previous lectures.

Suppose M starts existentially. Gate numbers will have binary representation $\in \{0,1\}^{ic \log n}$ (for a gate at depth i) and the bits of this representation indicate the existential/universal choises made by M, so far. Looking at a gate one can tell the choices we have made so far. At a leaf, the gate type is x_l or $\overline{x}_l$, where the l-th input symbol is queried at the M's execution after the nondeterministic choices given by the gate number. To check if a gate proceeds another gate, just check if the representation of one is a prefix of the other.

Q.E.D.

Theorem 120 *If L has a *-uniform AC^0 circuit then $L \in LH$.*

Proof. [*of Theorem 3*] Let L have polysize circuits $\{c_n\}$ of depth k, starting with an OR gate and having AND's and OR's in alternating layers.

An LH algorithm for L is:

1. Initially $g_0 = 0$ (output gate); $i = 0$.
2. Guess g_{i+1} (intended to be input to g_i).

   Universally do:

   a. Verify g_{i+1} is an input to g_i. Accept if so, reject otherwise.

   b. Continue.

   Universally choose g_{i+2} (if $i + 2 < k$).

   Existentially do:

   a. Check if g_{i+2} is an input to g_{i+1}. Accept if not, otherwise reject.

   b. Continue, loop back to 2.
3. Once $g_0, g_1, \ldots, g_k$ have been picked:

   Guess the type t of g_k and universally do

   a. Accept if g_k has type t, reject otherwise.

   b. Accept if $t = x_l$ or $\overline{x}_l$ and the l-th input symbol is 1 or 0 respectively. Reject otherwise.

Q.E.D.

0.1 Multiplication of n–bit numbers

Theorem 121 *Multiplication is in Linear Time hierarchy.*

Proof. We will have a TM that on input x, y can guess z with $z = xy$, where z has at most $2n$ bits.

The problem is how to verify that $z = xy$. The idea is to check $z = xy(\bmod p)$, for many small primes p. By Chebychev's theorem we know that there exists a constant b such that the product of all primes smaller that u is at least 2^{bu}. So, using Chinese Remainder Theorem we get:

Corollary 122 *If $z \geq x, y$ then $z = xy$ iff $\forall p > 1,\ p \leq \frac{2}{b} \log z :\ z = xy(\bmod p)$.*

Let $m = \log n$.

The algorithm is the following:

1. Universally choose p. $0 < p \leq \frac{2}{b} z \leq \frac{2}{b} 2n$. p has $m + \mathcal{O}(1)$ bits.

   We want to verify $z = xy(\bmod p)$. Consider the binary representation of x. We group the bits of this representation into blocks of m bits each. In other words we consider the representation of x in base 2^m. Thus x can be expressed as:

   $$x = \sum_{j=0}^{\lceil \frac{n}{m} \rceil} \overline{x}_j 2^{jm}$$

   where $\overline{x}_j$ is an m-bit integer, namely $\overline{x}_j$ is bits $j \cdot m$ through $j \cdot m + m - 1$ of x. We do the same for y and z.

2. Guess the following values:

   a. $x \bmod p,\ y \bmod p,\ z \bmod p$.

   b. For all $i < \lceil \frac{n}{m} \rceil,\ \sum_{j=0}^{i-1} \overline{x}_{\lceil \frac{n}{m} \rceil + j + i} \cdot 2^{jm} \bmod p$

   c. The same as b. for y and z.

3. Universally verify (and accept if all are true)

   a. $((x \bmod p) \cdot (y \bmod p) = (z \bmod p)) \bmod p$

   b. For all $i < \lceil \frac{n}{m} \rceil$:

   $$(\sum_{j=0}^{i-1} \overline{x}_{\lceil \frac{n}{m} \rceil + j - i} \cdot 2^{jm} \bmod p) 2^m + \overline{x}_{\lceil \frac{n}{m} \rceil - i - 1} = (\sum_{j=0}^{i} \overline{x}_{\lceil \frac{n}{m} \rceil + j - i} \cdot 2^{jm} \bmod p)$$

   c. The same as b. for y and z.

Q.E.D.

0.2 Division

The rest of the course will treat material from Beame-Cook-Hoover, SIAM J. Comput. 15 (1986) 994-1003. We will start with a discussion of circuits for computing division and

reciprocals; see Knuth's Volume 1 for an overview. Additionally, Reif, SIAM J. Comput. 15 (1986) 138-145, has some of the best results to date on aymptotically efficient circuits for division.

Definition 56 *DIVISION is the following problem:*

Input: x, y, *two n-bit integers*

Output: $\lceil \frac{x}{y} \rceil$, *an n-bit integer.*

Definition 57 *RECIPROCAL is the following problem:*

Input: a, $1/2 \leq a < 1$ *with n significant bits.*

Output: $1/a$ *to the same accuracy.*

How to compute $1/a$? We can use Newton's Method. Consider $f(x) = 1/a - a(1/a - x)^2$. This has $1/a$ as a fixpoint. Equivalently $f(x) = 2x - ax^2$.

Suppose $1/2 < a < 1$. Choose initially $x_0 = 3/2$, so that $|1/a - x_0| \leq 1/2$ and set $x_{i+1} = f(x_i)$.

Let $\epsilon_i = |x_i - 1/a|$. Then $\epsilon_{i+1} = |x_{i+1} - 1/a| = a\epsilon_i^2 < \epsilon_i^2$, and we have quadratic convergence.

So, $x_{\lceil \log n \rceil} \approx 1/a$ with error at most 2^{-n}, i.e. $x_{\lceil \log n \rceil}$ is accurate to n places.

This method yields a polynomial size circuit of depth $\mathcal{O}\left(\log^2 n\right)$.

Another method is to use higher order versions of Newton's Method. In this case we have $f(x) = 1/a - a^{k-1}(1/a - x)^k$, for any $k \geq 2$. The function $f(x)$ has $1/a$ as a fixpoint and by a similar argument as above the order of convergence is k. Although large k means better convergence, the method is not in general better than the simple Newton's method, because as k increases the computational complexity of $f(x_i)$ increases too.

We can also use Taylor series.

$$\frac{1}{1-x} \approx 1 + x + x^2 + \ldots + x^{2^i - 1} = (1+x)(1+x^2)\cdots(1+x^{2^{i-1}})$$

This method also gives a $\mathcal{O}\left(\log^2 n\right)$ depth circuit.

Definition 58 *POWERING is the following problem:*

Input: x *an n-bit integer.*

Output: $x^0, x^1, x^2, \ldots, x^n$, n *many* n^2*-bit integers.*

Definition 59 *ITERATED PRODUCT is the following problem:*

Input: $x_1, x_2, \ldots, x_n$, *n-bit integers.*

Output: $\prod_{i=1}^{n} x_i$, *an* n^2*-bit integer.*

Obviously POWERING is a special case of ITERATED PRODUCT.

Theorem 123 *DIVISION is uniformly* NC^1 *reducible to POWERING.*

Proof. By the following algorithm:

1. Find j such that $2^{j-1} \leq y < 2^j$. Wlog $j \geq 1$
   Let $u = 1 - 2^j y$, so that $0 < u \leq 1/2$.
2. Compute with a POWERING circuit $u^0, u^1, \ldots, u^n$.
3. Let $\overline{u} = u^0 + u^1 + \cdots + u^n$, using Vector Addition.
   The idea is that $\overline{u} \approx \frac{1}{1-u}$. In fact $\overline{u} < \frac{1}{1-u}$ and
   $$\left| \frac{1}{1-u} - \overline{u} \right| = u^{n+1} + u^{n+2} + \cdots \leq 2^{-n}$$
4. Since $y = (1-u)2^j$, approximate $1/y$ by $\overline{y} = \overline{u}2^{-j}$.
5. Compute $t = x\overline{y}$.
   We have:
   $$\frac{x}{y} - t = x(\frac{1}{y} - \overline{y}) < 2^n 2^{-j-n} = 2^{-j} \leq \frac{1}{2}$$
   Set $r = x - \lfloor t \rfloor y$.
   If $r < y$ output $\lfloor t \rfloor$, otherwise output $\lfloor t \rfloor + 1$.

Q.E.D.

Theorem 124 *POWERING is uniformly NC^1 reducible to DIVISION (or to RECIPROCAL).*

Proof. By the following algorithm:

1. Set $u = 2^{2n^3+2n^2}$, $v = 2^{2n^2} - x$.
2. Compute $\lfloor \frac{u}{v} \rfloor$.

$$\frac{u}{v} = 2^{2n^3} \frac{1}{1 - 2^{-2n^2}x} = 2^{2n^3} \sum_{i=0}^{\infty} 2^{-2n^2 i} x^i = \sum_{i=0}^{\infty} 2^{2n^2(n-i)} x^i$$

If $i \leq n$ these terms are integers. If $i > n$ they are < 1.

We estimate the tail:

$$\sum_{i=n+1}^{\infty} 2^{2n^2(n-i)} x^i = \sum_{j=1}^{\infty} 2^{-2n^2 j} x^{j+n} << 1$$

since $x < 2^n$. So,

$$\lfloor \frac{u}{v} \rfloor = \sum_{i=0}^{n} 2^{2n^2(n-i)} x^i$$

and for $i \leq n$ x^i has at most n^2 bits. Thus, for each i, x^i is written in columns $2n^2(n-i)$ through $2n^2(n-i+1)-1$ of the binary representation of $\lfloor \frac{u}{v} \rfloor$.

Q.E.D.

Theorem 125 *Computing $\lfloor \frac{x}{m} \rfloor$, where x is an n-bit integer and $m \leq n$, is in logspace-uniform NC^1.*

Proof. What the theorem asserts is that division by small numbers is in NC^1. It suffices to think of m as being fixed, since we can make n many circuits, one for each value of $m \leq n$.

The logspace circuit constructor precomputes the values $a_i \equiv 2^i \bmod m$. The circuit does the following:

On input $x = (x_{n-1}, \ldots, x_0)_2$

1. It computes $y = \sum a_i x_i$. So, $y \equiv x \bmod m$ and $0 \leq y \leq m(m-1) < m^2$.
2. In parallel compute the m-numbers $y - tm$, for $t = 0, 1, \ldots, m-1$.
3. Output the value $y - tm$ which is between $0, m$, i.e. $0 \leq y - tm < m$

It suffices to note that each step can be done in logarithmic depth. The first by using Vector Addition circuits and the other two because multiplication can be done in logarithmic depth.

Q.E.D.

Theorem 126 *Computing $x \bmod m$, where x is an n-bit integer and $m \leq n$, is in logspace-uniform NC^1.*

Proof. From Long-division algorithm,

if $z = (z_{n-1} \ldots z_0)_2 = \lfloor (x_{n-1} \ldots x_0)_2 / m \rfloor$ then $z_i = 1$ if and only if $2((x_{n-1} \ldots x_{i+1})_2 \bmod m) + x_i \geq m$. We can compute $(x_{n-1} \ldots x_{i+1})_2 \bmod m$, using the previous theorem.

Q.E.D.

Theorem 127 *Computing $x^{-1} \bmod m$, x, m as above, is in logspace-uniform NC^1.*

Proof. $x^{-1} \bmod m \equiv u$ iff $xu \equiv 1 (\bmod m)$ and $0 \leq u < m$. We can in parallel try each value of u and use the previous theorem to check if $xu \equiv 1 (\bmod m)$ or $xu \not\equiv 1 (\bmod m)$.

Q.E.D.

Theorem 128 *On input $x_1, x_2, \ldots, x_n$, n-bit integers, and on input p^l, p a prime, $p^l \leq n$ the value $\prod x_i \bmod p$ can be computed in logspace-uniform NC^1.*

Proof. We will need some facts from number theory. If $Z^*_{p^l}$ is the multiplicative group $\bmod p^l$ then there exists a generator for this group, i.e. there exists g such that $g^0, g^1, \ldots$ contains all the elements of the group. The only exception is when $p = 2$ and $l \geq 3$. In this case 5 and $-1 \equiv 2^l - 1 (\bmod 2^l)$ can generate all the elements of the group, i.e. every element in $Z^*_{2^l}$ can be written uniquely as $5^a(-1)^b$ with $b \in \{0, 1\}$ and $0 \leq a < 2^{l-2}$.

The circuit constructor for the theorem finds a generator g for each p^l (or uses 5,-1 for the special case).

For each $0 \leq x < p^l$, relatively prime to p,

'express x as $g^i \bmod p^l$ or $5^i(-1)^j \bmod 2^l$'.

Hardwire into the circuit look up tables, for finding i (or i and j) form x and for finding x from i (or i and j). These are, of course, essentially the discrete logarithm and anti-logarihtm functions.

The circuit is constructed to do the following:

1. For each x_i, in parallel

   compute all values $(x_i \bmod p^j)$, $(p^j \leq n)$ and let j_i be the greatest value such that $p^{j_i} | x_i$. Compute $y_i = (x_i \bmod p^l)/p^{j_i}$.

2. Compute $j = \sum j_i$ (Vector addition). So,

   $$\prod x_i \equiv p^i \prod y^i (\bmod p^l)$$

   From now on we treat two cases, depending on whether there are one or two generators of the group. The two cases are similar. We give the algorithm for the first case.

3. Find a_i's so that $y_i = g^{a_i}$, where g is the generator of the group, i.e. we take logarithms by table look up. We do this in parallel for all i.

4. Compute $a = \sum a_i$ (Vector addition).

5. Compute $\overline{a} = a \bmod p^l - p^{l-1}$.

6. Compute $y = g^{\overline{a}}$, using table look up (anti-logarithm).

7. Compute $p^j y \bmod p^l$ and output this.

We can do this because if $j \geq l$ then output 0, otherwise $p^j \leq n$ and $p^j y$ is a product of 'small' numbers.

Q.E.D.

As a by-product of the proof method we get the following corollary:

Corollary 129 *On input two n-bit integers x, y and $m \leq n$ the value $x^y \bmod m$ can be computed in logspace-uniform NC^1.*

Proof. The circuit does the following:

1. Finds each $p^l | m$ such that p is prime and p^{l+1} does not divide m (hardwired).
2. For each such p^l

   take $\overline{y} \equiv y(\bmod p^l - p^{l-1})$ and compute $x^{\overline{y}} \bmod p^l$, by the circuit of the previous theorem. This is equal to $x^y \bmod p^l$, since $Z^*_{p^l}$ has order $p^l - p^{l-1}$.
3. Given $x^y \bmod p^l$ for such p^l the Chinese Remainder Theorem says that there exists a unique z, $0 \leq z < m$, such that $z \equiv x^y(\bmod p^l)$ for all the p^l's. This z can be found from the values $x^y \bmod p^l$ by the table look up (hardwired by the logspace circuit constructor).

Alternatively [Dave Robinson's algorithm]:

1. Compute $\overline{y} \equiv y \bmod \phi(m)$, where $\phi(m)$ is the number of relative primes to m and smaller than m.
2. Compute $\overline{x} \equiv x \bmod m$.
3. Compute $\overline{x}^{\overline{y}} \bmod m$ by table lookups.

Q.E.D.

MATH 267A
Circuit Complexity & Computational Complexity

Lecture Date: June 4, 1992 **Topic: Division and Powering**
Instructor: Sam Buss **Notes by: Sam Buss**

This lecture continues the material from Beame, Cook and Hoover. So far, we have established that the following problems have logspace-uniform NC^1 circuits (we say a number is 'small' iff it is $n^{O(1)}$ where n is the input length):

- Division by a small number (Theorem 125),
- Remainder w.r.t a small number (Theorem 126),
- Computing reciprocal modulo a small number (Theorem 127),
- Powering modulo a small number (Theorem 128),
- Exponentiation modulo a small number (Theorem 129).

The CHINESE REMAINDERING problem is defined as follows:

CHINESE REMAINDERING PROBLEM:
INPUTS:
Pairwise relatively prime integers $c_1, \ldots, c_n \leq n^2$ (so each c_i is a $2\lceil \log n \rceil$-bit number).
Integers $x_1, \ldots, x_n$ such that $0 \leq x_i \leq c_i$ for all i.
OUTPUT: An integer x s.t. $0 \leq x \leq \prod_i c_i$ and such that $x \bmod c_j = x_i$ for all i.

Since $c_1, \ldots c_n$ are presumed to be relatively prime, there will always be a unique value x to be output as the solution of the Chinese Remaindering problem. This is, of course, the content of the Chinese Remainder Theorem.

It is useful to recall the proof of the Chinese Remainder Theorem: the main step in the proof is to show that there exists numbers $u_1, \ldots, u_n$ such that each u_s is congruent to 1 mod c_s and is congruent to 0 mod $c_{s'}$ for all $s' \neq s$. Then we let

$$y \;=\; x_1 \cdot u_1 + x_2 \cdot u_2 + \cdots + x_n \cdot u_n.$$

Clearly $y \equiv x_s \bmod c_s$ for all s. Then we let x be y plus an appropriate multiple of $(\prod_s c_s)$ so that $0 \leq x < \prod_x c_s$. That suffices proves the existence part of the Chinese Remainder Theorem, and the uniqueness follows by a pigeonhole argument. It still remains to show that the numbers u_s can be obtained. The usual proof is based on Euclid's algorithm; however, another proof is to let u_s be defined as

$$u_s \;=\; \left(\prod_{i \neq s} c_i\right)^{-1} \bmod c_s.$$

This latter definition of u_s is not as elementary as the Euclid algorithm proof, however, it yields a computationally tractable method of obtaining u_s (with the aid of Theorem 127).

Lemma 130 *Chinese Remaindering modulo $c_1, \ldots, c_n < n^2$ is logspace-uniform-NC^1 reducible to the problem of computing the product $\prod_{i=1}^n c_i$.*

Lemma 130 says that if one has efficient circuits for computing the product of 'small' numbers than one can build efficient circuits for solving Chinese Remaindering.

Proof. Lemma 130 is proved by using the following algorithm for Chinese Remaindering. We leave it to the reader to check that the algorithm can be tranformed into a log-depth circuit.

INPUT: Integers $x_1, \ldots, x_n$ and $c_1, \ldots, c_n$ with $0 \leq x_s < c_s \leq n^2$.

(1) Use oracle to get the product $c = \prod_{i=1}^n c_i$.

(2) Compute $v_s = c/c_s = \prod_{i \neq c_j} c_i$ for all s, (in parallel).
This can be done in log depth since c_s is 'small'.

(3) Compute $w_s = v_s^{-1} \bmod c_s$ for all s (in parallel).
Thus $0 \leq w_s < c_s$ and $v_s \cdot w_s \equiv 1 \bmod c_s$.
This is computable in log depth since c_s is 'small'.

(4) Compute $u_s = v_s \cdot w_s$ for all s in parallel, using log-depth multiplication circuits.
Note $u_s = w_s \cdot \frac{c}{c_s} \leq c$.

(5) Compute $y = \displaystyle\sum_{s=1}^{n} x_s \cdot u_s$ (using log-depth vector addition circuits).
Now $y \bmod c_s = x_s$ for all s.
And $0 \leq y < n(n^2 \cdot c) = n^3 \cdot c$.

(6) Compute, in parallel, the values $y - tc$ for $t = 0, 1, 2, \ldots, n^3 - 1$.
Output the (unique) value $x = y - tc$ such that $0 \leq y - tc < c$.
By construction, $x \bmod c_s = x_s$ for all s.

Q.E.D.

In the previous lemma, the Chinese Remaindering Theorem was reduced to the problem of computing a product of small numbers; indeed, knowing *any* product of small numbers would be enough. This motivates the next definition:

Definition 60 *A* good modulus sequence *is a sequence of integers $M_1, M_2, M_3, \ldots$ such that there is a constant $d \geq 1$ and a polynomial $r(n)$ so that*

(1) $2^n \leq m_n \leq 2^{n^d}$, *and*

(2) *For all prime powers $p^\ell | M_n$, we have $p^\ell \leq r(n)$.*

In other words a good modulus sequence is a sequence of integers (of length polynomial in n) which have only small prime power factors.

Example: The following are good modulus sequences:

(1) M_n equal to the product of the first n primes.

(2) $M_n = \prod\{p^\ell : p^\ell \leq n < p^{\ell+1} \text{ and } p \text{ is prime}\}$.

Theorem 131 *ITERATED PRODUCT is logspace-uniform-NC^1 reducible to the problem of computing any good modulus sequence.*

Proof. We describe an algorithm which has as input n many n-bit integers $x_1, \ldots, x_n$ and which outputs their product $\prod x_i < 2^{n^2}$:

(1) Use oracle to get the value M_{n^2}.

(2) Factor M_{n^2} by checking, in parallel, for each prime power $p^\ell < r(n^2)$ whether $p^\ell | M_{n^2}$ and $p^{\ell+1} \nmid M_{n^2}$. This finds all maximal prime power factors of M_{n^2}, since M_{n^2} belongs to a good modulus sequence.
Let $c_1 = p_1^{\ell_1} \cdot c_2 = p_2^{\ell_2} \cdot \ldots c_k = p_k^{\ell_k}$ be the prime factorization of M_n^2.

(3) Compute $B_s = \prod x_i \bmod c_s$ for all s, in parallel.

(4) Use Chinese Remaindering to find $\prod x_i$. (Using the known factorization of M_{n^2}).

Q.E.D.

We can now prove the main results of Beame-Cook-Hoover:

Theorem 132 *ITERATED PRODUCT is in P-uniform NC^1.*

Proof. In polynomial time we can compute the good modulus sequence from Example (1) above. This involves (on input n), finding the first n primes and computing their product. The straightfoward (highschool) algorithms for this take time polynomial in n.

Now the proof of Theorem 131 shows that there are P-uniform log-depth circuits for ITERATED PRODUCT since the value of M_{n^2} may be hardwired into the circuit by a polynomial time circuit constructor.

Q.E.D.

Corollary 133 *DIVISION and POWERING have P-uniform NC^1 circuits.*

Open Problem 3 *Give a logspace construction of a good modulus sequence. More generally show the division is in logspace.*

Theorem 134 *ITERATED PRODUCT is logspace-uniform-NC^1 reducible to POWERING.*

Corollary 135 *DIVISION, POWERING and ITERATED PRODUCT are equivalent under logspace-uniform-NC^1 reductions.*

Proof. We shall prove Theorem 134 by proving that $M_n = \binom{2n}{n}$ forms a good modulus sequence and is computable by logspace-uniform NC^1 circuits with an oracle for POWERING.

The circuit for computing $\binom{2n}{n}$ is quite simple: form the $2n+1$ bit binary number $2^{2n}+1$, then use a powering circuit to compute $(2^{2n}+1)^2 n$. We need to prove three things:

Claim 1: $2^n \leq \binom{2n}{n} < 2^{2n}$. This is an elementary fact about Pascal's triangle.

Claim 2: The binary representation of $(2^{2n}+1)^2 n$ contains the $2n$-th row of Pascal's triangle with the binary representation of each entry from that row of Pascal's triangle in a block of

$2n$-bits. This is easily proved from the binomial theorem, using the fact that $\binom{2n}{i} < 2^{2n}$ for all i.

From this $M_n = \binom{2n}{n}$ is easily computed.

Claim 3: Finally, we must show that if p is prime and p^ℓ divides $\binom{2n}{n}$, then $p^\ell \leq 2n$.

For this we recall that $\binom{2n}{n} = \frac{(2n)!}{n!n!}$. Now we need the fact that, for p prime, the maximum value ℓ for which p^ℓ divides $n!$ is equal to

$$\sum_{i>0} \left\lfloor \frac{n}{p^i} \right\rfloor .$$

This not too difficult to prove if one notes that $\lfloor n/p \rfloor$ is the number of multiples of p in the product $n(n-1)\cdots 2\cdot 1$, that $\lfloor n/p^2 \rfloor$ is the number of multiples of p^2 in the product, and so forth.

Thus, the maximum value ℓ for which p^ℓ divides $\binom{2n}{n}$ is equal to

$$\sum_{i>0} \left(\left\lfloor \frac{2n}{p^i} \right\rfloor - 2 \cdot \left\lfloor \frac{n}{p^i} \right\rfloor \right) \quad \leq \quad \sum_{p^i \leq 2n} 1 \quad \leq \log_p(2n).$$

Thus, if $p^\ell | M_n$, we have $p^\ell \leq 2n$. That establishes that $\{M_n\}_n$ is a good modulus sequence.

Q.E.D.

It remains an open question whether $\binom{2n}{n}$ can be computed in space $O(\log n)$.

MATH 267A
Circuit Complexity and Computational Complexity

Instructor: Sam Buss
Winter and Spring Quarters 1992, UCSD

Homework Problems

Starred ($\star$) problems are ones that Sam does not know how to solve at the time that he assigns them.

1 Prove that $C_{\{\wedge,\vee,\neg\}}(\oplus) \leq 4$ by explicitly describing a circuit of size 4. Your example should also show $L_{\{\wedge,\vee,\neg\}}(\oplus) \leq 4$. Here $\oplus$ denotes the binary parity function.

2 Prove that $C_{\{\wedge,\vee,\neg\}}(\oplus) = 4$.

3$\star$ Show that there is no $\{\wedge, \vee, \neg\}$-formula of size $2^{2^n} - n$ which computes all the Boolean functions of n variables. Give the best lower bound you can for the size of such a formula.

Best answer so far: Size is $\geq \frac{3}{2} \cdot 2^{2^n}(1 - o(1))$ [F. Bäuerle and D. Robinson]. Similar size bound also obtained by M. Clegg.

4$\star$ Define $g(1) = 1$ and, for $n > 1$, define $g(n) = 2^n + g(\lceil \frac{n}{2} \rceil) + g(\lfloor \frac{n}{2} \rfloor)$. Prove or disprove: $g(n)$ is the optimal size of a circuit computing all the full minterms on n variables. [This problem is still open as of June 1992.]

5 In part (b) of the second Shannon theorem, we showed that for any set of m Boolean functions of n variables, there is a circuit of size $(m+1)\frac{2^n}{n}(1 + o(1))$ computing all m functions. Improve this bound to $m\frac{2^n}{n}(1 + o(1))$.

6 Let m be fixed. Generalize the first Shannon theorem to give a lower bound on the size of a circuit computing m Boolean functions of n variables, for almost all sets of m functions.

The next three problems discuss improvements to part (a) of the Spira-Brent theorem. Recall that this stated that if C is an $\{\wedge, \vee, \neg\}$-circuit of leaf-size m, then there is an equivalent $\{\wedge, \vee, \neg\}$ circuit of depth $\beta \log m$ and of leaf-size m^α, where $\beta = \frac{2}{\log 3 - 1} \approx 3.419$ and $\alpha = 2.1963$ (so α satisfies $\frac{1+2^\alpha}{3^\alpha} \leq \frac{1}{2}$). Problems 7 and 8 both depend on the following construction: If C_0 and C_1 are defined as in class, then either C_0 logically implies C_1 or vice-versa (in symbols $C_1 \geq C_0$ or $C_0 \geq C_1$). In the first case we have the equivalences:

$$C \equiv C_0 \vee (D \wedge C_1) \equiv (C_0 \vee D) \wedge C_1.$$

and in the second case, we have:

$$C \equiv C_1 \vee ((\neg D) \wedge C_0) \equiv (C_1 \vee \neg D) \wedge C_0$$

7 Improve part (a) of the Spira-Brent theorem to have $\alpha = 2$ with the same value for β. [Hint: Use the above constuction and otherwise follow the outline of the proof given in class.]

8 Now improve part (a) of the Spira-Brent theorem to have $\alpha = 1.735$ with $\beta = \frac{1}{2 - \log 3} \approx 2.4094$. Here, α is picked so that

$$\frac{1 + 2^\alpha + 3^\alpha}{4^\alpha} \leq 1.$$

[Hint: Show that the proof for problem 7, can be modified so that the leaf-size of D is $\leq \frac{m}{2}$, so that C_i is equivalent to a formula with leaf size $\leq \frac{m}{2}$ and so that C_{1-i} is equivalent to a formula with leaf size $\leq \frac{3}{4}m$ (for at least one value of $i = 0, 1$). Then use an appropriate equivalence from the hint above.]

9⋆ Further improve the constants for part (a) and/or part (b) of Spira's theorem.

Best answer so far: Based partly on the technique of problem 7, Luisa Bonet has shown that for any $\alpha > 1$ there is a β for which part (a) of Spira's theorem holds. This is also shown by Bshouty-Cleve-Eberly in the 32nd FOCS, 1991.

The next two problems are due to [Paul '77]. Let ISA_n be the n-ary Boolean function defined as follows: Suppose $p = 2^{2^\ell}$ and $b = \log p$ and $k = \log p - \log\log p$ and that $n = 2p + k$. For $a_1, \ldots, a_r \in \{0,1\}$, let $(a_1 \cdots a_k)_2$ be the integer with binary representation given by the a_i's. Then $ISA_n(z_1, \ldots, z_k, x_0, \ldots, x_{p-1}, y_0, \ldots, y_{p-1})$ is defined by letting $i = (z_1 \cdots z_k)_2$ and letting

$$j = (x_{ib} x_{ib+1} \cdots x_{ib+b-1})_2$$

and then setting

$$ISA_n(z_1, \ldots, z_k, x_0, \ldots, x_{p-1}, y_0, \ldots, y_{p-1}) = y_j.$$

Note that ISA_n implements an indirect addressing function.

(10) Show that $L^*_{B_2}(ISA_n) = \Omega\left(\frac{n^2}{\log n}\right)$. [Hint: Use Neciporuk's theorem, letting the blocks Y_i be the i-th block of the $\vec{x}$ inputs, namely, $Y_i = \{x_{ib}, \ldots, x_{ib+b-1}\}$.]

(11) Show that $C_{\{\wedge,\vee,\neg\}}(ISA_n) = O(n)$. [Hint: Recall the circuits for computing all full minterms.]

(12) Here is an improvement of Krapchenko's theorem due to [Elias Koutsoupias, 1992]. If x is a truth assignment and A is a set of truth assignments, define $d_A(x)$ to be equal to the number of neighbors of x in the set A. Let f be a Boolean function. Let A and B disjoint, nonempty sets of truth assignments such that $A \subseteq f^{-1}(0)$ and $B \subseteq f^{-1}(1)$, or vice-versa.

(a) $L^*_{\{\wedge,\vee,\neg\}}(f) + 1$ is greater than or equal to

$$K'_{A,B}(f) \;=\; \frac{1}{|A|}\sum_{x \in B} (d_A(x))^2.$$

(b) Letting $K_{A,B}(f)$ be the lower bound from Krapchenko's theorem as discussed in class, prove that

$$K_{A,B}(f) \;\leq\; K'_{A,B}(f) \;\leq\; n^2.$$

(c) Apply the lower bound from (a) to the functions defined by $f_k(\vec{x}) = 1$ iff exactly k of $\vec{x}$ have value True (for fixed k). Compare this lower bound to the lower bound Krapchenko's theorem gives. (Try $k = 2$, for instance.)

13⋆ In class, we gave a $\{\wedge, \vee, \neg\}$ formula for the fulladder with 9 gates. Is this size optimal? This fulladder had seven AND and OR gates. Is this optimal?

14⋆ Find (and try to improve) the claimed $O(n \log n)$-size, depth $O(\log n)$ circuits for the comp.theory problem.

15 (Cumulative Counting) Define $CC(x_1, \ldots, x_n)$ to be the function with value the vector of $n \log n$ bits y_i, j, for $1 \leq i \leq n$ and $1 \leq j \leq \lceil \log(n+1) \rceil$, such that for each value of i, the signals $y_{i,j}$ code (in binary notation) the number of 1's among $x_1, \ldots, x_i$. Find $O(n \log n)$-size, $O(\log n)$-depth circuits for $CC(x_1, \ldots, x_n)$. [Hint: combine techniques used for Vector Addition and for Parallel Prefix.]

16 (Unary Sorting) Let $T_n^k(x_1, \ldots, x_n)$ be the k-threshhold function which has value 1 iff at least k of its inputs equal 1. Consider the n-ary unary sorting function

$$\langle x_1, \ldots, x_n \rangle \;\mapsto\; \langle T_n^n(\vec{x}), T_n^{n-1}(\vec{x}), \ldots, T_n^2(\vec{x}), T_n^1(\vec{x}) \rangle.$$

Show that this function has circuits of size $O(n)$ and depth $O(\log n)$.

17 (due to W. Paul). Show that the storage access function

$$SA(a_1, \ldots, a_{\log n}, x_0, \ldots, x_{n-1}) = x_{|\vec{a}|}$$

has $\{\wedge, \vee, \neg\}$-circuits of size $2n + o(n)$.

18 (due to W. Paul). Show that any B_2-circuit for the storage access function SA has size $\geq 2n - 2$. [Hint: Given a circuit C for the storage access function, let P be the set of gates that are (explicitly) the parity or negated parity of some set of inputs. Write X or A to mean the parity (or negation of the parity) of the nonempty subset X or A of the inputs $\vec{x}$ or $\vec{a}$, respectively. Find an $\wedge$-type gate which has at least one input from P of the form X or $X \oplus A$ or a negation of one of these. By setting one of the conditions $X = c$ or $X = c \oplus A$, where c is a constant, the $\wedge$-type gate can be forced to a constant — this eliminates this gate and any gate it feeds into. Iterate this process, with the proviso that subsequent sets X must be linearly independent of the prior X's. Either the process continues for $n-1$ rounds, so $\geq 2n-2$ gates are eliminated from the circuit; or we arrive at a point where there is a particular x_j such that for *any* setting of the values of $\vec{a}$, there are two settings of values for the x_i's for which the circuit has the same value but with x_j receiving different values under the two settings. This gives a contradiction when $|\vec{a}| = j$.] This hint is Buss's method; Paul (1977) has a somewhat simpler proof.

19 Prove or disprove: If a Boolean function can be written as an OR of ANDs of fanin $\leq s$ then every minterm of f has size $\leq s$.

20 From results proved in class, we know that $Parity_n$ has polynomial size, unbounded fanin circuits of depth $(\log n)/(\log \log n)$. Prove that, for $c > 0$, it has polynomial size unbounded fanin circuits of depth $(\log n)/(c + \log \log n)$.

21 Is the $\leq_{cd-tt}$ reduction of *Vector Addition* to *Binary Count* logtime or log hierarchy uniform? (It is fairly easy to see that it is logspace uniform.)

22 Show $Binary\,Count \leq_{cd-tt} Majority$ (Note that it suffices to show $Unary\,Count \leq_{cd-tt} Majority$).

23 Show, for $q \in N$, that $Mod\text{-}q \leq_{cd-tt} Binary\,Count$.

24 Is $\{Y_1, \ldots, Y_n\}$ U_F^n-complete for $h \in F \setminus \{0, 1\}$?

25 Is $\prod_{i=1}^{n} X_i$ U_F^n-complete?

26 Suppose f_n is a family of functions which are not computable by circuits of depth k of at most $2^{o(n^{\frac{1}{2k}})}$ many nearly F-easy gates for any k. Further suppose $\{f_n\} \leq_{cd} \{g_n\}$. Prove that there is a constant r such that it is not the case that $\{g_n\}$ has circuits of depth k with $2^{o(n^{\frac{1}{rk}})}$ many nearly F-easy gates.

27 (Smolensky-Barrington). Prove that depth k circuits for *Majority* require $2^{\Omega(n^{\frac{1}{2k}})}$ many nearly F-easy gates.

28 Prove the claim made at the end of the proof of the Claim 100 (which was part of the proof of Valiant-Vazirani theorem, Theorem 97).

29⋆ Try to make improvements to the constants in the Valiant-Vazirani theorem. Especially, can the probability that there is some i with $|S_i|$ is odd be made as large as $1 - \delta^n$ or $1 - \left(\frac{1}{n}\right)$? (Where δ is a constant.)

30 This question concerns alternatives to the choice of Toda polynomials (which were based on $g(n) = 3n^4 + 4n^3$).

(a) Suppose we set $g(n) = 3n^2 - 2n^3$. State and prove analogues of Lemmas 103-105. [Hint: use $x \equiv 1 \bmod N$ in place of $x \equiv -1 \bmod N$.]

(b) Why wouldn't a second-degree polynomial, say $g(n) = x^2 - 2x$, work?

(c⋆) How can Corollary 106 or Theorem 107 be proved using $g(n) = 3n^2 - 2n^3$ in place of the Toda polynomial used in class?

31 Prove that if $D(n) \geq \Omega(\log n \log\log n))$, then if $\{C_n\}$ is a family of U_D-uniform circuits of depth $D(n)$, then $\{C_n\}$ is also U_{E^*}-uniform. [Hint: binary search.]

32⋆ Work out the theory of uniform AC^k for $k \geq 1$.

33⋆ Rework the proof of Ruzzo's Theorem 116 so that every gate in the (encoded) circuit s connected (possibly, indirectly) to the output gate. This might increase the depth a little.

34⋆ Recall that deterministic log time Turing machines can compute the length n of their input. Suppose that the definition of Turing machines which have an address tape for accessing input symbols is changed, so that the tape head is required to be at the most significant (i.e., leftmost) bit of the address of the input tape in order to read an input symbol. Can a deterministic Turing machine (with this restriction) find the length of its input in $O(\log n)$ time? Can it approximate the length of its input to within a factor of two? [The problem of whether the precise length can be computed is open.]

35 Let $r, s \geq 2$. The *r-s-Base-Conversion* problem is the problem of, given an integer x in base r, to compute x's base s representation. Show that *r-s-Base-Conversion* is in constant alternation linear time.

As a corollary, prove that, the log time hierarchy could have equivalently been defined by using the convention that the address tape of a Turing machine used to access input symbols have addresses written in an arbitrary (fixed) base r. For instance, we could have used base 3 instead of base 2.