Parity $\notin AC^0$ using Hastad's Switching Lemma

Utkarsh Patange

Indian Institute of Technology Kanpur

April 5, 2014

Motivation for a weaker class

- To prove $P \neq NP$, knowing $P \subseteq NP$, we must find a language $L \in NP$ and $L \notin P$. That is a lower bound on the resources required to decide a language must be obtained.
- Since our conventional models of computation are very powerful, it becomes difficult to comment on the lower bounds. An indirect line of attack, to quote Johan Hastad [H] would be:

"We want to prove that the computer cannot do something quickly. We cannot do this. But if we tie the hands and feet of the computer together maybe we will have better luck. The hope being of course that we eventually will be able to remove the ropes and prove that the full powered computer needs a long time"

Boolean Circuits

- We define a boolean circuit as a Directed Acyclic Graph where internal nodes are labeled with one of $\vee, \wedge$ and $\neg$ representing that the "output of the node" is OR, AND or NEGATION of it's input(s)
- An edge (u, v) in the DAG represents that the output of node u is given as an input to node v

Boolean Circuits

- We define a boolean circuit as a Directed Acyclic Graph where internal nodes are labeled with one of $\vee, \wedge$ and $\neg$ representing that the "output of the node" is OR, AND or NEGATION of it's input(s)
- An edge (u, v) in the DAG represents that the output of node u is given as an input to node v
- Leaf nodes (nodes with in-degree $= 0$) are bits of the input string.
- Output of the circuit is defined to be the output of the root node (node with out-degree $= 0$). There is only one root node in any circuit
- We define some terms here:

Boolean Circuits

- We define a boolean circuit as a Directed Acyclic Graph where internal nodes are labeled with one of $\vee, \wedge$ and $\neg$ representing that the "output of the node" is OR, AND or NEGATION of it's input(s)
- An edge (u, v) in the DAG represents that the output of node u is given as an input to node v
- Leaf nodes (nodes with in-degree $= 0$) are bits of the input string.
- Output of the circuit is defined to be the output of the root node (node with out-degree $= 0$). There is only one root node in any circuit
- We define some terms here:
  - Depth of a circuit:= Longest path from root to any leaf node.
  - Fan in of a node:= The in-degree of the node
- Note that fan in of a node labeled $\neg$ can only be 1. Also, we have not restricted fan-in's of any other node, neither have we restricted the depth of the circuit

Boolean Circuits

- We define a *T(n)-size circuit family* to be a sequence $\{C_n\}_{n\in\mathbb{N}}$ of Boolean circuits, where C_n has n inputs, single output and $|C_n| \leq T(n)\,\forall n \in \mathbb{N}$ [AB]
- A language L is said to be *recognized* by a circuit family $\{C_n\}_{n\in\mathbb{N}}$ if $\forall x \in \{0,1\}^n$, $x \in L \iff C_n(x) = 1$ [AB]

Boolean Circuits

- We define a *T(n)-size circuit family* to be a sequence $\{C_n\}_{n\in\mathbb{N}}$ of Boolean circuits, where C_n has n inputs, single output and $|C_n| \leq T(n)\, \forall n \in \mathbb{N}$ [AB]
- A language L is said to be *recognized* by a circuit family $\{C_n\}_{n\in\mathbb{N}}$ if $\forall x \in \{0,1\}^n,\ x \in L \iff C_n(x) = 1$ [AB]
- With these definitions, if we allow a circuit of polynomial size and unbounded depth, we get a class $P_{/Poly}$ which is found to be "recognizing" languages that are undecidable.
- Due to this, we try to restrict the class of boolean circuits even further to get AC and NC.

The class AC

- For every d, $L \in AC^d$ if L can be decided by a family of circuits $\{C_n\}$ where C_n has
  - size polynomial in n
  - depth $O(\log^d n)$
  - unbounded fan-in. i.e., any node (except input nodes, of course) can have arbitrarily many inputs.
- $AC = \cup_{i \geq 0} AC^i$
- We can also define another class NC which is same as AC except that the circuits are allowed a fan-in of two only.

The class AC

- For every d, $L \in AC^d$ if L can be decided by a family of circuits $\{C_n\}$ where C_n has
  - size polynomial in n
  - depth $O(\log^d n)$
  - unbounded fan-in. i.e., any node (except input nodes, of course) can have arbitrarily many inputs.
- $AC = \cup_{i \geq 0} AC^i$
- We can also define another class NC which is same as AC except that the circuits are allowed a fan-in of two only.
- Since unbounded fan-in can be simulated using a tree of ORs/ANDs of depth $O(\log n)$, $NC^i \subseteq AC^i \subseteq NC^{i+1}$. For $i = 0$, the inclusion is known to be strict. Clearly, NC^0 is very limited, while due to unbounded fan-in, AC^0 is not.
- We get $AC^0 \neq NC^1$ by studying the parity function $\oplus$. We will prove that this function is not in AC^0

Proof Sketch

- We will first prove Hastad's Switching Lemma and using that prove $\oplus \notin AC^0$
- The original proof given by Johan Hastad [H] is rather complicated. We present here an alternate proof by Razborov [RA].
- Before we state the lemma, we need to define a few terms:

Proof Sketch

- We will first prove Hastad's Switching Lemma and using that prove $\oplus \notin AC^0$
- The original proof given by Johan Hastad [H] is rather complicated. We present here an alternate proof by Razborov [RA].
- Before we state the lemma, we need to define a few terms:
  - $k-$CNF:= A boolean formula that is in conjunctive normal form and every clause of which has at most k literals. Similarly, we define $k-$DNF.
  - Restriction on a function:= Assigning some value to some of the input variables to the function
  - $f|_\rho$:= A function f under restriction ρ. That is, $f|_\rho$ takes an assignment τ to variables not assigned any value by ρ and outputs f applied to ρ and τ.
  - $f|_{\pi\rho} := f|_\pi|_\rho$ for restrictions π and ρ that are on disjoint sets of variables

Switching Lemma: Statement

If f is a function that is expressible as a $k-$DNF and ρ is a random restriction that assigns random values to t randomly selected input bits, then $\forall s \geq 2$

$$\Pr_{\rho}[f|_{\rho} \text{ is not expressible as } s\text{-}CNF] \leq \left(\frac{(n-t)k^{10}}{n}\right)^{s/2} \tag{1}$$

Terms Used

- min-term of $f :=$ A partial assignment to f's variables that makes f output $\mathbf{1}$ regardless of what value is assigned to the rest of the variables
- max-term of $f :=$ A partial assignment to f's variables that makes f output $\mathbf{0}$ regardless of what value is assigned to the rest of the variables
  For example, Consider a function that is expressible as a $k-$DNF. Every clause in this formula can yield a size-k min-term of f. Similarly in a function that is expressible as a $k-$CNF, every clause can yield a size-k max-term.

Terms Used

- min-term of f:= A partial assignment to f's variables that makes f output $\mathbf{1}$ regardless of what value is assigned to the rest of the variables
- max-term of f:= A partial assignment to f's variables that makes f output $\mathbf{0}$ regardless of what value is assigned to the rest of the variables
  For example, Consider a function that is expressible as a $k-$DNF. Every clause in this formula can yield a size-k min-term of f. Similarly in a function that is expressible as a $k-$CNF, every clause can yield a size-k max-term.
- We will assume henceforth that the min-terms (respectively max-terms) are minimal. That is, no assignment to a proper subset of the term's variables would make the function 1(respectively 0).

Size of max-term

Theorem

If all max-terms of a function are of size at most s, then the function is expressible as an $s-CNF$.

- It is a known result from boolean algebra (circuit minimization) that if two functions have same set of max-terms then they are equivalent. This can be proved by representing the functions as product of sums for which there is a unique representation
- In the function f, we consider each of its max-terms σ_i one-by-one and construct a clause C_i corresponding to it

Size of max-term

Theorem

If all max-terms of a function are of size at most s, then the function is expressible as an $s-CNF$.

- It is a known result from boolean algebra (circuit minimization) that if two functions have same set of max-terms then they are equivalent. This can be proved by representing the functions as product of sums for which there is a unique representation
- In the function f, we consider each of its max-terms σ_i one-by-one and construct a clause C_i corresponding to it
- For any variable x that is assigned 1 in σ_i, we include $\overline{x}$ in C_i and for $y = 0$ in σ_i, we include y in C_i.
- Taking OR of all of the literals in C_i, we get a clause.
- Doing this for every max-term gives us a set of clauses each having at most s literals. Thus, the theorem stands proved.

Size of max-term

Theorem

If all max-terms of a function are of size at most s, then the function is expressible as an $s-$CNF.

Due to the above theorem we can say something about the size of the max-term of a function which *cannot* be expressed as an $s-$CNF

If a function f cannot be expressed as an $s-$CNF, then there must be at least one max-term of f of size $\geq s+1$

Finding the probability

- Let R_t be the set of all restrictions of $t \geq n/2$ variables. We have $|R_t| = \binom{n}{t} 2^t$
- Let B be the set of *bad restrictions* – those $\rho \in R_t$ for which $f|_\rho$ is not expressible as an $s-$CNF
- To prove the switching lemma, we must prove a specific bound on probability of a random restriction being bad $= \frac{|B|}{|R_t|}$
- To compute $|B|$, we establish a one-to-one mapping $G : B \rightarrow R_{t+s} \times \{0,1\}^\ell$ for some $\ell = O(s \log k)$
- This will give us $|B| = \binom{n}{t+s} 2^{t+s+\ell} = \binom{n}{t+s} 2^t 2^s k^{O(s)}$

Finding the probability

- Let R_t be the set of all restrictions of $t \geq n/2$ variables. We have $|R_t| = \binom{n}{t}2^t$
- Let B be the set of *bad restrictions* – those $\rho \in R_t$ for which $f|_\rho$ is not expressible as an $s-$CNF
- To prove the switching lemma, we must prove a specific bound on probability of a random restriction being bad $= \frac{|B|}{|R_t|}$
- To compute $|B|$, we establish a one-to-one mapping $G : B \rightarrow R_{t+s} \times \{0,1\}^\ell$ for some $\ell = O(s \log k)$
- This will give us $|B| = \binom{n}{t+s}2^{t+s+\ell} = \binom{n}{t+s}2^t 2^s k^{O(s)}$

If such G exists, we get

$$\frac{|B|}{|R_t|} = \frac{\binom{n}{t+s}2^t 2^s k^{O(s)}}{\binom{n}{t}2^t} = \frac{\binom{n}{t+s}2^s k^{O(s)}}{\binom{n}{t}} \qquad (2)$$

To prove the lemma, it suffices to prove $\binom{n}{t+s} \leq \binom{n}{t}\left(\frac{e(n-t)}{n}\right)^s$ since this will imply a bound that is stronger than the lemma

Proving $\binom{n}{t+s} \leq \binom{n}{t}\left(\frac{e(n-t)}{n}\right)^s$

It suffices if we prove,

$$\frac{t!(n-t)!}{(t+s)!(n-t-s)!} \leq e^s \cdot \frac{(n-t)^s}{n^s}$$

Consider

$$\underbrace{\left(\frac{n-t}{n-t} \cdot \frac{n-t-1}{n-t} \cdot \ldots \cdot \frac{n-t-s+1}{n-t}\right)}_{\leq 1} \cdot \underbrace{\left(\underbrace{\frac{n}{t+s}}_{\leq 2} \cdot \underbrace{\frac{n}{t+s-1}}_{\leq 2} \cdot \ldots \cdot \underbrace{\frac{n}{t+1}}_{\leq 2}\right)}_{\leq 2^s \leq e^s \text{ Since } t \geq n/2}$$

Hence proved

Proving upper bound on probability

Thus we get,

$$\frac{|B|}{|R_t|} = \frac{\binom{n}{t+s} 2^s k^{O(s)}}{\binom{n}{t}} \leq \left(\frac{2e(n-t)}{n}\right)^s k^{O(s)}$$

$$\Pr_{\rho}[f|_{\rho} \text{ is not expressible as } s\text{-}CNF] \leq \left(\frac{2e(n-t)}{n}\right)^s k^{O(s)} \leq \left(\frac{(n-t)k^{10}}{n}\right)^{s/2}$$

Which is the statement of the switching lemma

We will see later that the term $O(s)$ is actually $\approx 2s$. Thus, we are relaxing the upper bound in the sense that power of a term less than 1 is being decreased and that of a term greater than 1 is increased

Proving upper bound on probability

Thus we get,

$$\frac{|B|}{|R_t|} = \frac{\binom{n}{t+s}2^s k^{O(s)}}{\binom{n}{t}} \leq \left(\frac{2e(n-t)}{n}\right)^s k^{O(s)}$$

$$\Pr_{\rho}[f|_\rho \text{ is not expressible as } s\text{-}CNF] \leq \left(\frac{2e(n-t)}{n}\right)^s k^{O(s)} \leq \left(\frac{(n-t)k^{10}}{n}\right)^{s/2}$$

Which is the statement of the switching lemma

We will see later that the term $O(s)$ is actually $\approx 2s$. Thus, we are relaxing the upper bound in the sense that power of a term less than 1 is being decreased and that of a term greater than 1 is increased Also, we are assuming here that $k \geq 2 > (2e)^{\frac{1}{3}}$ since for $k = 1$, we can always build an $s-$CNF $\forall s \geq 2$.

Constructing the Mapping

- Thus to prove the switching lemma, it is sufficient to describe the one-to-one mapping $G : B \to R_{t+s} \times \{0,1\}^{\ell}$.
- Note that a bad restriction ρ cannot make any clause of f true, since that will give $f|_{\rho} = 1$. Similarly, not all clauses are made false by ρ
- Since $f|_{\rho}$ is not expressible as an $s-$CNF, it has some max term π of size greater than s.
- Assuming π to be maximal, we can say that $f|_{\rho\pi} = 0$ but $\forall \pi' \subset \pi \; f|_{\rho\pi'} \neq 0$
- We will define the mapping $G(\rho) = (\rho\sigma, c)$ where σ is a suitably defined restriction on *exactly* s of π's variables and $c \in \{0,1\}^{\ell}$
- Thus, $\rho\sigma$ restricts at least $t + s$ variables and the proof follows

Constructing the Mapping

- Let the clauses be ordered in an arbitrary fashion: $t_1, t_2, \ldots, t_i, \ldots, t_n$. Within the clauses, the variables are also ordered in an arbitrary way.
- By definition, $\rho\pi$ is a restriction that sets all the clauses to zero
- We split π into $m \leq s$ sub-restrictions $\pi_1, \pi_2, \ldots, \pi_m$ inductively as follows:
    - Assume we already have $\pi_1, \pi_2, \ldots, \pi_{i-1}$ such that $\pi_1\pi_2\ldots\pi_{i-1} \neq \pi$
    - Let t_{l_i} be the first clause in our ordering of terms that is not 0 under $\rho\pi_1\pi_2\ldots\pi_{i-1}$

Constructing the Mapping

- Let the clauses be ordered in an arbitrary fashion: $t_1, t_2, \ldots, t_i, \ldots, t_n$. Within the clauses, the variables are also ordered in an arbitrary way.
- By definition, $\rho\pi$ is a restriction that sets all the clauses to zero
- We split π into $m \leq s$ sub-restrictions $\pi_1, \pi_2, \ldots, \pi_m$ inductively as follows:
  - Assume we already have $\pi_1, \pi_2, \ldots, \pi_{i-1}$ such that $\pi_1\pi_2\ldots\pi_{i-1} \neq \pi$
  - Let t_{l_i} be the first clause in our ordering of terms that is not 0 under $\rho\pi_1\pi_2\ldots\pi_{i-1}$
  - Let Y_i be the set of all variables set by π but not by $\rho\pi_1\pi_2\ldots\pi_{i-1}$. Since t_{l_i} is 0 under π, $Y_i \neq \phi$
  - We define π_i to be the restriction that is induced by π on the variables of Y_i (and hence sets t_{l_i} to 0)
  - We define σ_i to be the restriction of Y_i that *keeps* t_{l_i} *from* being 0. Such a restriction must exist since otherwise, t_{l_i} would be 0 under $\rho\pi_1\pi_2\ldots\pi_{i-1}$ itself
- Note that σ_i *does not* ensure $t_{l_i} = 1$ since the clause is an $\wedge$ of literals

Constructing the Mapping

- This process is continued until at least s variables are assigned due to the restriction $\pi_1\pi_2\ldots\pi_m$
- We *trim* π_m in an arbitrary way so as to make these restrictions assign exactly s variables
- Note that π assigned more than s variables. So, $\pi_1\pi_2\ldots\pi_m \neq \pi$

Constructing the Mapping

- This process is continued until at least s variables are assigned due to the restriction $\pi_1\pi_2\ldots\pi_m$
- We *trim* π_m in an arbitrary way so as to make these restrictions assign exactly s variables
- Note that π assigned more than s variables. So, $\pi_1\pi_2\ldots\pi_m \neq \pi$
- Also note that $\forall i < j$, π_j does not assign values to variables that are already assigned by π_i
- We define $G(\rho) = (\rho\sigma_1\sigma_2\ldots\sigma_m, c)$. To prove that this is one-to-one mapping, we must prove that we can invert it uniquely
- Since there is no way to identify ρ from $\sigma_1, \sigma_2, \ldots, \sigma_m$ we keep some additional information in c to help us

Uniquely inverting the mapping

- Given the assignment $\rho\sigma_1\sigma_2\ldots\sigma_m$, we plug this into f and try to infer t_{l_1} from it
- t_{l_1} is the first clause that is not fixed to 0 by ρ. This property is maintained by σ_1.
- Also, $\sigma_2\ldots\sigma_m$ does not assign values to any variables in t_{l_1}.

Uniquely inverting the mapping

- Given the assignment $\rho\sigma_1\sigma_2\ldots\sigma_m$, we plug this into f and try to infer t_{l_1} from it
- t_{l_1} is the first clause that is not fixed to 0 by ρ. This property is maintained by σ_1.
- Also, $\sigma_2\ldots\sigma_m$ does not assign values to any variables in t_{l_1}.This is because all variables of t_{l1} restricted by π were already there in π_1 and therefore cannot be in later π_i's and hence σ_i's

Uniquely inverting the mapping

- Given the assignment $\rho\sigma_1\sigma_2\ldots\sigma_m$, we plug this into f and try to infer t_{l_1} from it
- t_{l_1} is the first clause that is not fixed to 0 by ρ. This property is maintained by σ_1.
- Also, $\sigma_2\ldots\sigma_m$ does not assign values to any variables in t_{l_1}.This is because all variables of t_{l1} restricted by π were already there in π_1 and therefore cannot be in later π_i's and hence σ_i's
- In string c, we include
  - $s_1 :=$ number of variables in π_1
  - The indices in t_{l_1} of the variables in π_1
  - And the values that π_1 assigns to these variables
  - We will need $O(s_1 \log k)$ bits to store this information since each term has at most k variables
- Once we know t_{l_1}, using the information in c, it is easy to reconstruct π_1

Uniquely inverting the mapping

- Knowing π_1, we can change the restriction to $\rho\pi_1\sigma_2\ldots\sigma_m$

Uniquely inverting the mapping

- Knowing π_1, we can change the restriction to $\rho\pi_1\sigma_2 \ldots \sigma_m$
  - Since we know the variables to which π_1 assigns values, we only need to change them to the exact assignment that π_1 assigns
  - This information is there in c

Uniquely inverting the mapping

- Knowing π_1, we can change the restriction to $\rho\pi_1\sigma_2 \ldots \sigma_m$
    - Since we know the variables to which π_1 assigns values, we only need to change them to the exact assignment that π_1 assigns
    - This information is there in c
- Now, we can similarly work out which clause is t_{l_2}. It is again the first non-zero clause under the new restriction
- Here, we will need to use the next $O(s_2 \log k)$ bits of c to reconstruct π_2
- We continue this process until we have processed all m clauses and figured out $\pi_1, \pi_2, \ldots, \pi_m$. We can then simply remove the assignments by π_i's to get ρ
- We get, $|c| = O((s_1 + s + 2 + \ldots + s_m) \log k) = O(s \log k)$

Summary

- We first proved a theorem stating that if a function cannot be expressed as an $s-$CNF, then it must have at least one max-term of size $> s$
- We called a restriction ρ a bad restriction if $f|_\rho$ cannot be expressed as an $s-$CNF
- For a function $f|_\rho$ that cannot be expressed as an $s-$CNF, we assumed one such max-term π
- Using π we constructed σ and defined $G(\rho) = (\rho\sigma, c)$ where c contained information about π that helped us reconstruct ρ when we are given $\rho\sigma$. This proved G to be a one-to-one mapping
- Since the one-to-one mapping was from a set of bad restrictions to another set that could be easily counted, we got an idea of the cardinality of the set of bad restrictions
- Using this cardinality, we were able to bound the probability of choosing a bad restriction among all possible restrictions of a particular size

References

Hastad, Johan (1987)
Computational limitations of small depth circuits
Ph.D. thesis, Massachusetts Institute of Technology.

Arora Barak (2009)
Computational Complexity: A Modern Approach
Published on April 2009 by Cambridge University Press

Razborov, Alexander A. (1993)
An equivalence between second order bounded domain bounded arithmetic and first order bounded arithmetic
Arithmetic, proof theory and computational complexity 23: 247277

The End