JASS'06 Report Summary

Circuit Complexity

Konstantin S. Ushakov

May 14, 2006

Abstract

Computer science deals with many computational models. In real life we have normal computers that are constructed using, so called, silicon chips. Boolean circuits model may be considered a formalization of the silicon chip. That's the reason why it is very interesting to determine a set of problems which can be solved by boolean circuits.

Contents

1 Introduction

Boolean circuits model is a formalization of the silicon chip - the basic block of computers in their todays state. *Boolean circuit* with n inputs is a directed acyclic graph with n leaves *(input wires).* Each inner node(gate) of the graph is labeled with logical operation ($\vee$, $\wedge$, $\neg$). The $\vee$ and $\wedge$ nodes have indegree of 2, and $\neg$ nodes have indegree of 1. There is at least one output node. The input n bits are fed to the input gates. The circuit works as follows: each internal gate applies the operation it is labeled with on it's incoming edges and produces a single bit. As the graph is acyclic there are no feedback effects, so, a single output bit emerges in the *output* node (see Figure 1). Hence, the circuit with one output gate implements a boolean function from $\{0,1\}^n \to \{0,1\}$. The size of the circuit $S(C_n)$ is the number of edges in it. The depth of the circuit is the length of the longest path from input to output excluding NOT gates.

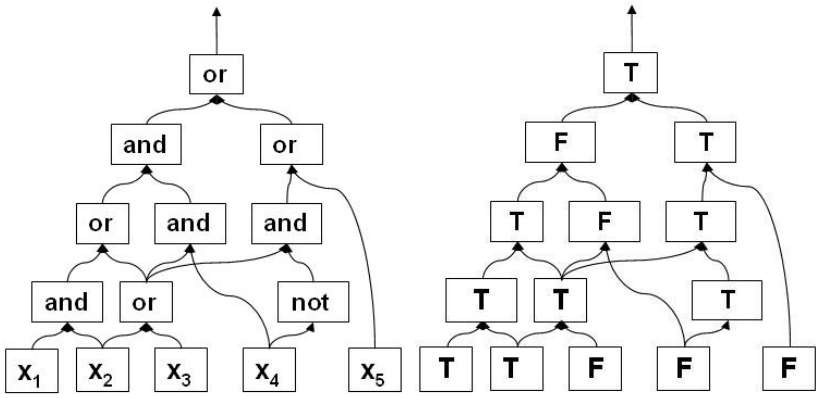

Figure 1: Circuit Work Example

Boolean functions OR, AND and NOT form a *universal basis*, that means every boolean function of n-bit inputs can be implemented by a boolean circuit we have introduced.

Since in general we don't know the length of the input, we introduce a *circuit family* concept.

Definition 1.1 *Circuit family* $\{W_n\}_{n\geq 1}$ *is a sequence of circuits, where* W_n *has* n *inputs.*

Definition 1.2 *We say* $\{W_n\}$ *is a circuit family of size* $O(f(n))$*, if for every* n *size of circuit* W_n *is* $O(f(n))$*.*

Definition 1.3 *A circuit family decides a language* L *if for every input* x*,*

$$x \in L \Leftrightarrow W_{|x|}(x) = 1.$$

It is pretty obvious, that any language can be decided by a circuit family of size $O(n2^n)$ (simply encoding decision result for any possible input in the circuit). This means that even algorithmically undecidable languages can be decided by boolean circuits. This leads us to the question of circuit families construction. There are two possible approaches.

1. A circuit family must be generated by a computer, such circuits are called *uniform.* Uniform circuits can be used in real life, and so this approach seems to be logical. However, in this case the following theorem holds:

   Theorem 1 *A language* $L \subset \{0,1\}^*$ *has uniform polynomial circuits iff* L *lies in* $\mathbf{P}$*.*

2. Circuits can be defined in abstract way, such circuits are called *non-uniform*. This extends the number of problems which can be solved by circuits, but makes this approach unrealistic.

The positive thing is that in real life we don't need the whole circuit family. If, for instance, we create a circuit of a small size that solves **SAT** for input of size 1024, created circuit will solve lots of vital problems and help us to break some of todays cryptography systems. Another important fact is that we need to perform the construction of the circuit only once and then it could be placed in the hardware and it could be reused.

2 Polynomial Circuit Families

In real life we try to archive polynomial work time. For a circuit polynomial time of work means polynomial size of the circuit. There are two equivalent definitions of *polynomial size circuits.*

Definition 2.1 $L \in \mathbf{P/poly}$ *iff there exists* $\{C_i\}_{i\in\mathbb{N}}$ *and polynomial* p *that*

- $\forall i |C_i| \leq p(i)$
- $x \in L \Leftrightarrow C_{|x|}(x) = 1$ *(*$\{C_i\}_{i\in\mathbb{N}}$ *correctly decides L)*

Definition 2.2 $L \in \mathbf{P/poly}$ *iff there exists a polynomial time computable relation R, that*

$$\exists \{y_i\}_{i\in\mathbb{N}} \forall x \left(x \in L \Leftrightarrow R(x, y_{|x|}) = 1\right),$$

the advice string for the Turing machine does not depend on the input x but only on the length of the input.

Now, as we introduced a new complexity class, we are interested in it's relations with other known classes. It is obvious that $\mathbf{P} \in \mathbf{P/poly}$ and that $\mathbf{P}$ is not equal to $\mathbf{P/poly}$, because $\mathbf{P/poly}$ contains algorithmically undecidable languages. The interesting question is if $\mathbf{NP} \nsubseteq \mathbf{P/poly}$. This problem is still open, but it is not independent.

Theorem 2 (Karp-Lipton) *If* **SAT** *has polynomial circuits then the polynomial hierarchy collapses to the second level.*

Lemma 2.3 *The following language is* $\Sigma_3\mathbf{P}$*-complete*

$$L = \{x | \exists y \forall z \, (x, y, z) \in R\},$$

where R is a polynomially balanced relation decidable in NP.

Proof[Lemma] We know that is language $L \in \Sigma_{k+1}\mathbf{P} \Leftrightarrow \exists$ polynomially balanced relation $R \in \Pi_k\mathbf{P}$, that

$$\forall x \, (x \in L \Leftrightarrow \exists y R \, (x, y)).$$

Therefore, $L \in \Sigma_3\mathbf{P}$ iff there exists such relation R_1 from $\Pi_2\mathbf{P}$, that

$$\forall x \, (x \in L \Leftrightarrow \exists y R_1 \, (x, y)).$$

As $\Pi_2\mathbf{P} = co - \Sigma_2\mathbf{P}$, applying the rule one more time we get

$$L = \{x | \exists y \forall z \, (x, y, z) \in R_2\},$$

where $R_2 \in \Sigma_1\mathbf{P} = \mathbf{NP}$. □

Proof[Theorem] To prove that polynomial hierarchy collapses to the second level we prove that $\Sigma_2\mathbf{P} = \Sigma_3\mathbf{P}$. From this we immediately get the hierarchy collapse:

$$\Sigma_4\mathbf{P} = \mathbf{NP}^{\Sigma_3\mathbf{P}} = \mathbf{NP}^{\Sigma_2\mathbf{P}} = \Sigma_3.$$

To prove this equality we consider a $\Sigma_3\mathbf{P}$ -complete language

$$L = \{x | \exists y \forall z \, (x, y, z) \in R\},$$

where R is a polynomially balanced relation decidable in **NP**. Now we should show that L lies in $\Sigma_2\mathbf{P}$. For this we need to replace somehow relation R with another relation Q, which is polynomially balanced and decidable in **P**. The main idea is to hide the **NP** under the circuits, which can be computed in polynomial time.

1. If R lies in **NP** it can be reduced to **SAT**, because **SAT** is **NP**-complete problem, let F be the reduction. That means that $(x, y, z) \in R \Leftrightarrow F_{(x,y,z)}$ is satisfiable.
2. Let's denote $\mathbf{C} = \{C_0, \ldots\}$ the polynomial circuit family that solves **SAT** and $\mathbf{C_n} = \{C_0, \ldots, C_n\}$ the *initial segment* of this family. We say that $\mathbf{C_n}$ is a *correct initial segment* iff $\mathbf{C_n}$ correctly decides **SAT** for formulas of size $\leq m$.
3. We know that **SAT** is self-reducible, this means that for every formula G

   $$G = G[x := true] \vee G[x := false],$$

   where $G[x := true]$ is formula G, where all occurrences of x are substituted with *true* value.
4. Therefore, family of circuits $\{C_i\}_{i=1}^{n}$ it works correctly for formula ω of length n, if
   - for every variable x contained in ω

     $$C_{|\omega|}(\omega) = C_{|\omega[x:=true]|}(\omega[x := true]) \vee C_{|\omega[x:=false]}(\omega[x := false])$$

   - if ω contains no variables, then it is *true* or $false$, hence the circuit should work correctly on constants: $C_{true}(true) = true$ and $C_{false}(false) = false$
5. The initial segment $\mathbf{C_n}$ is a correct initial segment if and only if it works correctly $\forall\omega\,(|\omega| \leq n)$.

Now we can replace functionality of the relation R in the definition of language L by polynomial circuits. These circuits will check the formula, corresponding to our triple (x, y, z).

$x \in L \Leftrightarrow \exists \mathbf{C_m} \exists y \forall z (all\ of\ length\ at\ most\ m)$

- $\mathbf{C_m}$ *is a correct initial segment of length* m
- $\mathbf{C_m}(F_{(x,y,z)}) = true$

The question is: what m should we choose? In other words, how can we bind the length of formula $F_{(x,y,z)}$ depending on length of x. We need m to be polynomially bound, as the relation Q should be polynomially balanced. Let us suppose that on input x the largest expression $F_{(x,y,z)}$ that could be constructed is of length at most $p(|x|)$. Since R is polynomially balanced and F is polynomial-time, $p(n)$ is polynomial and we can take $m = p(|x|)$.

Now the final version of the definition of language L is as follows:

$x \in L \Leftrightarrow \exists \mathbf{C_{p(|x|)}} \exists y \forall z (all\ of\ length\ at\ most\ p(|x|))$

- $\mathbf{C_{p(|x|)}}$ *is a correct initial segment of length* $p(|x|)$
- $\mathbf{C_{p(|x|)}}(F_{(x,y,z)}) = true$

This representation of language L is correct.

$$x \in L \Rightarrow \exists y \forall z R\,(x, y, z) \Rightarrow \exists y \forall z F_{(x,y,z)} \in \mathbf{SAT} \Rightarrow \exists\, \{C_i\}_{i=1}^{m}\ correct\ initial\ segment$$

$$\{C_i\}_{i=1}^{m}\ correct\ initial\ segment \Rightarrow F_{(x,y,z)} \in \mathbf{SAT} \Rightarrow \exists y \forall z R\,(x, y, z) \Rightarrow x \in L$$

So, finally, we rewrote the definition of language L in the form

$$L = \{x | \exists y \forall z\,(x, y, z) \in R\},$$

where Q is polynomially balanced and polynomial-time decidable, and then the language L lies in $\Sigma_2\mathbf{P}$. □

Corollary 1 *If* **NP** *has polynomial circuits, then*

$$\mathbf{PH} = \Sigma_2\mathbf{P} \cap \Pi_2\mathbf{P}.$$

Proof This corollary is obvious as **PH** is closed under complement. So, if $\mathbf{PH} \subset \Sigma_2\mathbf{P}$, then co$-\mathbf{PH} = \mathbf{PH} \subset \Pi_2\mathbf{P}$. □

Corollary 2 *If* $\mathbf{NP} \subset \mathbf{P/poly}$ *iff there exists a sparce* **NP***-hard language in terms of Cook reduction.*

Proof Out of the coverage of this paper. □

3 Circuit families of size $O(n^k)$

Definition 3.1 **Size**$[f(n)]$ *is a class of languages accepted by boolean circuit families of size* $O(f(n))$.

Definition 3.2 **Size**$[n^k]$ *is a class of languages accepted by boolean circuit families of size* $O(n^k)$.

Theorem 3 $\Sigma_2\mathbf{P} \cap \Pi_2\mathbf{P} \not\subseteq$ **Size**$[n^k]$ *for any k.*

Lemma 3.3 $\Sigma_4\mathbf{P} \not\subseteq$ **Size**$[n^k]$ *for any k.*

Proof[Lemma] Let's consider all functions f that depend only on the first $c \cdot k \cdot \log(n)$ bits. It is obvious that such functions could be encoded in polynomial number of bits (for exapmle by truth table). Number of possible functions f is $2^{2^{c \cdot k \cdot \log(n)}} = 2^{n^{c \cdot k}}$. Number of possible boolean circuits of size n^k is at most $2^{\frac{k}{2}+n}$, which is less than number of possible functions f. Consider

$$M = \left\{ f \mid \forall c \left(circuit\ of\ size\ n^k \right) \exists x \left(input\ of\ length\ n \right) : f(x) \neq c(x) \right\}.$$

The set M is obviously not empty, since not all functions f can be encoded by a boolean circuit of size n^k. Impose some fast computable order "$\leq$" on set M, for instance lexicographical. Let F be one of the smallest functions in M. Take

$$L = \{x | F(x) = 1\}.$$

Now we will describe language L in the form $\exists\forall\exists\forall R$, where R is polynomially balanced and polynomial-time decidable.

$$y \in L \Leftrightarrow \begin{cases} F(y) = 1 \\ \forall c \exists x : F(x) \neq c(x) \\ \forall F' : (\forall c \exists x F'(x) \neq c(x)) \Rightarrow F \leq F' \end{cases}$$

Rewriting this in one line:

$$y \in L \Leftrightarrow \exists F' \forall c \forall F' \exists x \exists c' \forall x' : F(x) \neq c(x) \vee \left(\left(F \leq F' \right) \wedge F'(x') = c'(x') \right) \wedge F(y) = 1.$$

So, we have constructed the language L from $\Sigma_4\mathbf{P}$ which can't be accepted by a circuit of size n^k.

The proof is not completed. We proved that L can't be accepted by a circuit of size n^k, but we need to prove that it can't be accepted by a circuit of size $O(n^k)$. It is obvious how to complete the proof, because n^k grows faster than $c \cdot n^{k-1}$ and, hence, L can't be accepted by circuits of size $O(n^{k-1})$. □

Proof[Theorem] Assume that for some k $\Sigma_2\mathbf{P} \cap \Pi_2\mathbf{P} \subseteq$ **Size**$[n^k]$. Then there exist a polynomial circuits that decide **NP**. This leads us to collapse of the polynomial hierarchy to $\Sigma_2\mathbf{P} \cap \Pi_2\mathbf{P}$. But then

$$\mathbf{PH} = \Sigma_2\mathbf{P} \cap \Pi_2\mathbf{P} \subseteq \textbf{Size}[n^k]?!$$

We have a contradiction, and so our assumption was wrong. □

Corollary 3 **PH** $\not\subseteq$ **Size**$[n^k]$ *for every k.*

4 Circuit Complexity of PP

First of all let's remind what Merlin-Arthur protocols are.

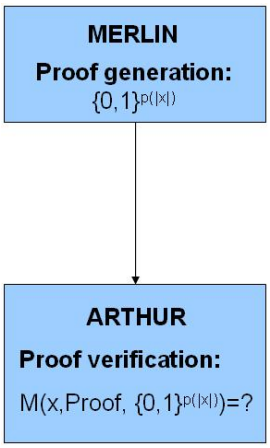

Figure 2: Merlin Arthur Protocol

$L \in \mathbf{MA}$ iff

- $x \in L \Leftrightarrow$ Merlin can think of a proof that Arthur will accept with high probability
- $x \notin L \Leftrightarrow$ every proof created by Merlin will be rejected by Arthur with high probability

As usual, Merlin is very clever and has infinite computational power, and Arthur is honest, but has polynomial computational power. Writing strictly $L \in \mathbf{MA}$ if there exist polynomials p and q and Turing machine M, working polynomial time on all inputs, that for every x:

$$x \in L \Leftrightarrow \exists y \in \{0,1\}^{p(|x|)} : \Pr_{z \in \{0,1\}^{q(|x|)}} \{M(x,y,z) = 1\} > \frac{3}{4}$$

$$x \notin L \Leftrightarrow \forall y \in \{0,1\}^{p(|x|)} : \Pr_{z \in \{0,1\}^{q(|x|)}} \{M(x,y,z) = 1\} < \frac{1}{4}$$

Additional knowledge that we will need.

Theorem 4 (Toda) $\mathbf{PH} \subset \mathbf{P}^{\mathbf{PP}}$.

Lemma 4.1 (Part of the proof of the Toda theorem) $\mathbf{P}^{\mathbf{PP}} = \mathbf{P}^{\#\mathbf{P}}$.

Theorem 5 $\mathbf{P}^{\#\mathbf{P}}$ *has interactive protocol with prover from* $\mathbf{P}^{\#\mathbf{P}}$.

We won't proof both theorems and lemma. Now we come to the main theorem of this section:

Theorem 6 $\mathbf{PP} \not\subseteq \mathbf{Size}[n^k]$ *for every* k.

Lemma 4.2 *If* $\mathbf{PP} \subset \mathbf{P/poly}$ *we have* $\mathbf{P}^{\mathbf{PP}} \subset \mathbf{MA}$.

Proof[Lemma] We know that $\mathbf{P}^{\#\mathbf{P}} = \mathbf{P}^{\mathbf{PP}}$. And by Theorem 5 we know that there exists an interactive protocol for $\mathbf{P}^{\#\mathbf{P}}$ with oracle from the same class. The problem is that during the communications between the prover and the verifier the verifier may ask questions to the prover more than once. And this conflicts with the definition of **MA** protocols. The main idea is that if the prover can encode itself in polynomial length and send to the verifier, this reduces required number of communication rounds to one.

Speaking strictly, $\mathbf{P}^{\#\mathbf{P}} = \mathbf{P}^{\mathbf{PP}} \subset \mathbf{P/poly}$, where the advice string for $\mathbf{P/poly}$ are polynomial circuits, which calculate the answers of the oracle from the class **PP** on inputs of length $poly(n)$. Let's take an interactive prover for $\mathbf{P}^{\#\mathbf{P}}$ from Theorem 5. We modify the interaction protocol in the following way (see Figure 3):

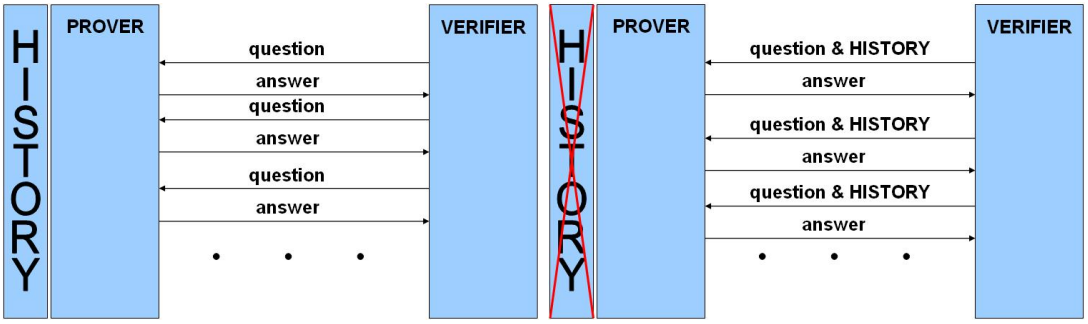

Figure 3: Initial and Modified protocol

- the prover does not remember the communication history
- the verifier remembers the communication history and sends it with every request to the prover (the communication history has polynimial size).

The modified prover acts as a simple $\mathbf{P}^{\#\mathbf{P}}$ machine. So, as $\mathbf{P}^{\#\mathbf{P}} \subset \mathbf{P/poly}$, there exist polynomial circuits for $\mathbf{P}^{\#\mathbf{P}}$. Now Arthur and Merlin appear, Arthur simulates verifier and in the first request to Merlin (prover) he asks to send him the polynomial circuits that encode the prover. As all requests to the prover had length $poly(n)$, the circuits are valid replacement for the prover. So, we constructed a Merlin-Arthur protocol for $\mathbf{P}^{\mathbf{PP}}$. □

Lemma 4.3 (without proof) $\mathbf{MA} \subseteq \mathbf{PP}$.

Proof[Theorem] Assume, that there exists k, that $\mathbf{PP} \subseteq \mathbf{Size}[n^k] \Rightarrow \mathbf{PP} \subseteq \mathbf{P/poly}$.

$$\begin{aligned} \mathbf{PH} \quad \subseteq \quad & \mathbf{P}^{\mathbf{PP}} \ (\textit{by Theorem } 4) \\ & \mathbf{MA} \ (\textit{by Lemma } 4.2) \\ & \mathbf{PP} \ (\textit{by Lemma } 4.3)\,. \end{aligned} \tag{1}$$

Since by Lemma 3 we know that $\mathbf{PH}$ is not a subset of $\mathbf{Size}[n^k]$, $\mathbf{PP} \not\subseteq \mathbf{Size}[n^k]$. □

References

[1] Aurora. *Computational Complexity: A Modern Approach*, chapter 6. 2006.

[2] Hirsh. Introduction to complexity theory. 2002.

[3] Papadimitriou. *Computational Complexity*, chapter 17. Addison-Weseley Publishing Company, 1994.