1 Notation

Let $\mathbb{P}_2(n)$ denote the set of all Boolean functions with n bits on input. Let

$$\mathbb{P}_2 = \bigcup_{n \in \mathbb{N}} \mathbb{P}_2(n)$$

denote the set of all Boolean functions.

2 Query models

A *query model* $\mathcal{M}$ is a subset of $\mathbb{P}_2$.

By a *determinsitic complexity* of a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ in a query model $\mathcal{M}$ we mean the following. There are Gollum and Bilbo. Gollum chooses $x \in \{0,1\}^n$. Bilbo wants to find $f(x)$. At the unit cost Bilbo can ask Gollum the value of $q(x)$ for any $q \in \mathcal{M} \cap \mathbb{P}_2(n)$. The minimal d for which there exists a Bilbo's algorithm that always finds $f(x)$ in at most d questions is called determinsitic complexity of f in $\mathcal{M}$ (denoted by $D^{\mathcal{M}}(f)$).

There is a standard way of representing Bilbo's algorithms in some query model $\mathcal{M}$ as binary trees. Nodes of a tree will represent possible positions of the game (and also states of an algorithm). Namely, we always start in the root of the tree. If in the current node Gollum answers YES, we go to the left child, if NO – to the right child. For every non-leaf v there is an n-bits function $q_v \in \mathcal{M}$, associated with this node. Bilbo will ask $q_v(x)$ if the current node is v. In the leaves of the tree are outputs of Bilbo.

We will call such trees (deterministic) $\mathcal{M}$-trees (or simply trees if $\mathcal{M}$ is clear from the context). For an $\mathcal{M}$-tree T we will denote its depth by $\mathrm{depth}(T)$. Note that $\mathrm{depth}(T)$ is just the complexity of the corresponding Bilbo's algorithm, i.e., the maximal number of questions Bilbo asks in T. We will also view an $\mathcal{M}$-tree T as a Boolean function $T\colon \{0,1\}^n \to \{0,1\}$. Namely, $T(x)$ will be the output of the corresponding Bilbo's algorithm on x.

3 Examples of query models

In this formalism we obtain

- the decision tree complexity by allowing Bilbo to ask the value of any selector function
- the parity decision tree complexity by allowing Bilbo to ask the value of any linear (over $\mathbb{F}_2$) function of x;
- the communication complexity by setting Q_n allowing Bilbo to ask any YES/NO question about the first half of x, and any YES/NO question about the second half of x.

We will use the notation $\mathcal{M} = dt$ for the decision tree complexity, $\mathcal{M} = \oplus dt$ for the parity decision tree complexity, and $\mathcal{M} = cc$ for the communication complexity.

4 Non-deterministic trees

First version. Non-deterministic $\mathcal{M}$-trees are expressions of the form:

$$N = \bigvee_{i=1}^{m} T_i.$$

where $T_1, T_2, \ldots, T_m$ are deterministic $\mathcal{M}$-trees. With such an expression we associate a Boolean function it computes:

$$N\colon x \mapsto N(x) = \bigvee_{i=1}^{m} T_i(x).$$

By the depth of N we mean $\operatorname{depth}(N) = \lceil \log_2(m) \rceil + \max_{i=1,\ldots,m} \operatorname{depth}(T_i)$. This will be just the depth of a tree obtained by presenting the OR in $\bigvee_{i=1}^{m} T_i$ via a binary $\vee$-tree.

Definition 1. *The non-deterministic complexity of f in a computational model $\mathcal{M}$ is the minimal d for which there exists a non-deterministic $\mathcal{M}$-tree N such that* $\operatorname{depth}(N) \leqslant d$ *and $N(x) = f(x)$ for all $x \in \{0,1\}^n$. This quantity will be denoted by $N^{\mathcal{M}}(f)$.*

Here all the Bilbo's non-deterministic choice is done in advance. One might try to get an advantage by allowing Bilbo to make deterministic choices during the talk with Gollum. It is not hard to see that up to a constant factor this leads to the same definition of $N^{\mathcal{M}}(f)$.

5 Polynomial hierarchy

Similarly we can define co-non-deterministic trees, Σ_2-trees and so on, and also the corresponding complexities of f:

$$coN^{\mathcal{M}}(f), \Sigma_2^{\mathcal{M}}(f), \ldots$$

For example, Π_2-$\mathcal{M}$-trees are expressions of the form:

$$\Pi = \bigwedge_{i=1}^{a} \bigvee_{i=1}^{b} T_{i,j}.$$

Here $T_{i,j}$ are deterministic $\mathcal{M}$-trees. Again, to define $\operatorname{depth}(\Pi)$ we represent this expression in a natural way as a binary AND-OR tree, attach $T_{i,j}$ to the leaves, and take the depth of the result.

6 Alternation

By an alternating $\mathcal{M}$-tree A we just mean a binary AND-OR tree with some deterministic $\mathcal{M}$-trees attached to its leaves. We define depth(A) as before – we draw the underlying AND-OR tree, then draw the deterministic protocols attached to it, and then take the depth of a tree on the resulting picture.

The function computed by an alternating tree A is defined as follows. We take $x \in \{0,1\}^n$, plug-in x into deterministic trees attached to the leaves of A, compute the outputs of these trees on x, put the results into the leaves, and then evaluate the underling AND-OR tree.

Definition 2. *The alternating complexity of f in a query model $\mathcal{M}$ is a minimal d such that there exists an alternating $\mathcal{M}$-tree of depth at most d that computes f. This quantity is denoted by $A^{\mathcal{M}}(f)$.*

Non-deterministic trees, Π_2-trees and so on can be seen as special cases of alternating trees. For example, a non-deterministic tree is an alternating tree where we just have a single layer of $\vee$'s. A Π_2-tree is an alternating tree where we first have a layer of $\wedge$'s, and then a layer of $\vee$'s. Of course, in general alternating trees the number of such layers is not bounded.

Remark. *WLOG we may assume that the deterministic trees in the leaves of an alternating tree are trivial. By "trivial " we mean that Bilbo asks only one question in them. This is because we can represent a deterministic tree is an OR over its leaves with output* 1*, and then each such leaf as an AND over questions Bilbo asks on the path to the leaf. This construction can only increase the depth of our initial alternating tree by a constant factor.*

So, e.g., NC^1 is just the set of all (sequences) of functions with logarithmic A^{dt}.

7 PSPACE

We now define "space complexity" of f in the computational model $\mathcal{M} = \{Q_n\}_{n\in\mathbb{N}}$. For that we turn back to the Bilbo-Gollum games. Recall that Gollum gets $x \in \{0,1\}^n$, and Bilbo's goal is to find $f(x)$. Before Bilbo wanted to minimize the number of questions he asks to Gollum. Now he will want to minimize the amount of *memory* he uses.

To formalize it we assume that Bilbo's actions are governed by a finite automaton $\mathcal{A}$. The automaton $\mathcal{A}$ has a finite set of states S with 3 designated stated s_{init}, s_0 and s_1. The state s_{init} will be the initial state, the states s_0 and s_1 will be two terminating states (the subscript indicates the output of Bilbo). For each $s \in S \setminus \{s_0, s_1\}$ there is a question $q_s \in Q_n$ that Bilbo asks to Gollum when in the state s. Finally, there is a transition function $\delta\colon S \setminus \{s_0, s_1\} \times \{YES, NO\} \to S$ that takes a current state of Bilbo and a Gollum's answer, and outputs the state of Bilbo upon receiving this answer.

We say that such an automaton computes f if whenever Bilbo plays with Gollum according to this automaton, the automaton comes into the state $s_{f(x)}$ after a finite number of question. Note that in principle we can come into a loop without ever reaching a terminal state.

Definition 3. ***The space complexity*** *of f in a query model $\mathcal{M}$ is the minimum of $\log_2(s)$ over all s for which there exists an s-state Bilbo's automaton that computes f. This quantity is denoted by $SPACE^{\mathcal{M}}(f)$.*

In other words, $SPACE^{\mathcal{M}}(f)$ is the minimal number of bits Bilbo needs to store a state of the smallest automaton computing f.

Equivalently, we can represent such automata as *branching programs*. These will be directed graphs where each node has out-degree 2, except for the two terminal nodes (one labeled by 0 and the other labeled by 1). There will also be one initial node. With each non-terminal node there is a question from Q_n associated with this node. This will be a question Bilbo asks Gollum in this node. For every non-terminal node v there is an ordering on its two out-going edges. If the answer is YES, we transit by the first out-going edge of our current node, if NO – by the second.

A branching program computes f if we come into a terminal, labeled by $f(x)$, whenever we play with Gollum having $x \in \{0,1\}^n$. The quantity $BP^{\mathcal{M}}(f)$ corresponds to the minimal size (the number of nodes) of a branching program, computing f.

Branching programs can be seen as transition graphs of the automata from the the definition of $SPACE^{\mathcal{M}}(f)$. So $SPACE^{\mathcal{M}}(f) = \log_2 BP^{\mathcal{M}}(f)$.

In a special case when these branching programs are trees we obtain just the standard deterministic complexity of f.

Theorem 1. *For any computational model $\mathcal{M}$ we have that $A^{\mathcal{M}}(f)$ and $SPACE^{\mathcal{M}}(f)$ are polynomially equivalent.*

8 AM

Let us define a notion of max $\frac{1}{2}$*-trees*. This is a rooted binary tree, where the non-leaf nodes are partitioned into the max-nodes and the *random* nodes. Now, take an max $\frac{1}{2}$-tree R and consider an assignment of the leaves of R to Boolean values. Let us define *the output* of R on such an assignment (this will be a rational number from $[0,1]$). We define it recursively. For a leaf, its output is just a Boolean value assigned to it. For a max-node, its output is just the maximum of the outputs of its children. For a random node, its output is the arithmetic mean of the outputs of its children.

Now, an *Arthur-Merlin $\mathcal{M}$-tree* (AM $\mathcal{M}$-tree for short) I is a max $\frac{1}{2}$-tree, equipped with an assignment of its leaves to deterministic $\mathcal{M}$-trees. We let depth(I) be the depth of a tree obtained by attaching these deterministic $\mathcal{M}$-trees to the underlying max $\frac{1}{2}$-tree.

For $x \in \{0,1\}^n$, we define the output of I on x as follows. First, we obtain an assignment of the leaves of the underlying max $\frac{1}{2}$-tree to Boolean values. We do so by evaluating the bottom deterministic $\mathcal{M}$-trees on x. Then we just let $I(x)$ to be the output of our max $\frac{1}{2}$-tree on this assignment.

We say that an AM $\mathcal{M}$-tree I computes $f\colon \{0,1\}^n \to \{0,1\}$ if for any $x \in \{0,1\}^n$ we have $|f(x) - I(x)| \leqslant 1/3$.

Definition 4. *The AM complexity of f in a computation model $\mathcal{M}$ is the minimal d such that there exists an AM $\mathcal{M}$-tree of depth at most d that computes f. This quantity is denoted by $AM^{\mathcal{M}}(f)$.*

Remark. *Here Merlin and Arthur are communicating by descending in the tree. Arthur communicates in the random nodes, where he descends to a random child (so he just sends random bits to Merlin). Merlin communicates in the* max*-nodes, where he is allowed to choose any child to descend. Finally, they reach some leaf. The verdict is made by evaluating a deterministic tree, attached to this leaf, on x. If the verdict is 1, Arthur is convinced that $f(x) = 1$. If the verdict is 0, he is not convinced.*

Clearly, $I(x)$ equals the maximum over all Merlin's strategies of the probability that Merlin convinces Arthur that $f(x) = 1$.

Theorem 2. *For any query model $\mathcal{M}$ we have*

$$SPACE^{\mathcal{M}}(f) = \left(AM^{\mathcal{M}}(f)\right)^{O(1)}.$$

9 Criteria for PSPACE=AM

Let $\mathcal{M}$ be a query model. Fix $n \in \mathbb{N}$. By an *elementary parity of size w* we mean the point-wise XOR of w functions from $\mathcal{M} \cap \mathbb{P}_2(n)$.

Theorem 3. *Let $\mathcal{M}$ be a query model. Then $SPACE^{\mathcal{M}}(f)$ and $AM^{\mathcal{M}}(f)$ are polynomially equivalent if and only if the Arthur-Merlin complexity of any elementary parity of size w is at most $(\log_2(w))^C$ for some absolute constant $C > 0$.*

The space complexity of any w-size elementary parity is $O(\log w)$. Indeed, ask the terms one by one, for that we should only remember the index of the current term and the current parity. So one direction of Theorem 3 is obvious: if the AM complexity of an w-size elementary parity can be super-polynomial in $\log(w)$, then $SPACE^{\mathcal{M}}(f)$ and $AM^{\mathcal{M}}(f)$ are not polynomially equivalent.

From Theorem 3 one can get the following sufficient condition, which is enough to establish $AM = PSPACE$ for the parity decision tree complexity, and for the communication complexity.

Corollary 4 (Sufficient condition for $AM = PSPACE$)**.** *Let $\mathcal{M}$ be a query model. Assume that there exists a constant $C > 0$ such that for any n the set $\mathcal{M} \cap \mathbb{P}_2(n)$ (represented as a set of 2^n-bit vectors) is a union of C linear subspaces of $\mathbb{F}_2^{2^n}$. Then $SPACE^{\mathcal{M}}(f)$ and $AM^{\mathcal{M}}(f)$ are polynomially equivalent.*

For $\oplus dt$ we just have $C = 1$, and for cc we have $C = 2$.

One can also formulate the following structural complexity version of Theorem 3.

Theorem 5. *We have $IP^A = PSPACE^A$ for any oracle A with $\oplus P^A \subseteq IP^A$.*

9.1 Proof of the hard direction of Theorem 3

Take any $f\colon \{0,1\}^n \to \{0,1\}$ with $SPACE^{\mathcal{M}}(f) = s$. By the equivalence of space and alternation, we can represent f as a binary AND-OR tree T of depth $d = s^{O(1)}$ having in the leaves functions from $\mathcal{M} \cap \mathbb{P}_2(n)$ and their negations. In a standard way one can identify the leaves of T with d-bit binary strings. For $u \in \{0,1\}^d$ we let $q_u \in \mathcal{M} \cap \mathbb{P}_2(n)$ be the function from the leaf u. We define $\delta_u = 1$ if this function is taken with the negation and $\delta_u = 0$ otherwise.

Assuming WLOG that the levels of AND's and OR's in T alternate (starting and ending with an OR), we can write:

$$f(x) = \exists u_1 \forall u_2 \ldots \exists u_d \;\; (q_{u_1 \ldots u_d}(x) \oplus \delta_u)\,.$$

We will start with just performing the Shamir's protocol. For that we algebrize $(q_{u_1 \ldots u_d}(x) \oplus \delta_u)$. For any fixed $x \in \{0,1\}^n$ this is just a function of $u_1, \ldots, u_d$ that can be written as a multi-linear polynomial $p_x \in \mathbb{F}_2[u_1, \ldots, u_d]$. In fact, by the standard interpolation trick we can write:

$$p_x(u_1, \ldots, u_d) = \sum_{v \in \{0,1\}^d} (q_v(x) \oplus \delta_v) \cdot u^v. \tag{1}$$

Here for two d-bit vectors f and g we write $f^g = (f_1 \oplus g_1) \cdot \ldots \cdot (f_d \oplus g_d)$.

Fix some $d^{O(1)}$-size field $\mathbb{F} \supset \mathbb{F}_2$. To perform the Shamir's protocol we only have to be able to do the following: for a given $r_1, r_2, \ldots, r_d, a \in \mathbb{F}$ check, whether:

$$p_x(r_1, \ldots, r_d) = a.$$

By substituting $r_1, \ldots, r_d$ into (1), we see that our goal is to check, whether in the field $\mathbb{F}$ we have the following equality:

$$\sum_{v \in \{0,1\}^d} (q_v(x) \oplus \delta_v) \cdot r^v = a.$$

Here by r^v we abbreviate $(r_1 + v_1) \cdot \ldots \cdot (r_d + v_d)$. Now, the last equality can be understood as $O(\log d)$ equalities over $\mathbb{F}_2$, by recalling that $\mathbb{F}$ is a linear space over $\mathbb{F}_2$. Each of these $O(\log d)$ equalities will be just an equality of some 2^d-size elementary parity to a constant. By our assumptions such an equality can be decided by a $d^{O(1)}$-depth AM-tree.