# CSC 2429 – Approaches to the P versus NP Question

## Lecture #7: 5 March 2014

**Lecturer: Joshua Grochow**

**Scribe Notes by: Mika Göös**

## 1 Algebraic complexity theory

This week we begin the study of computational complexity from the perspective of *algebraic computation*. That is, instead of boolean logic ($\vee$, $\wedge$, $\neg$, etc.) our model of computation uses the usual algebraic operations ($+$, $-$, $\times$, $/$, $\sqrt{\cdot}$, etc.) to manipulate elements in some field $\mathbb{F}$. Here we focus on the case $\mathbb{F} = \mathbb{R}$, the field of real numbers.

The study of algebraic computation is motivated, among other things, by the fact that the algebraic models typically have more structure than their boolean counterparts. In particlar, in many cases lower-bound results for boolean models imply lower bounds in the analogous algebraic models, so we might as well start by understanding the algebraic setting first.

The purpose of these scribe notes is to present a lower bound result due to Ben-Or [BO83].

## 2 Model: Algebraic computation trees

An *algebraic computation tree* over $\mathbb{R}$ is a binary tree $T$ where each node $v$ is assigned one of the following three roles:

1. **Computation node:** Here $v$ has exactly one child. The node computes a value $f_v$ using one of the following operations:

$$f_v := f_{v_1} \circ f_{v_2}, \quad f_v := c \circ f_{v_1}, \quad f_v := \sqrt{f_{v_1}},$$

   where $v_i$ is an ancestor of $v$ in $T$ or $f_{v_i} \in \{x_1, \ldots, x_n\}$, and $\circ \in \{+, -, \times, /\}$, and $c \in \mathbb{R}$.
2. **Comparison node:** Here $v$ has exactly two children (labelled *true* and *false*). The node performs one of the following tests:

$$f_{v_1} > 0, \quad f_{v_1} \geq 0, \quad f_{v_1} = 0,$$

   where $v_1$ is an ancestor of $v$.
3. **Output node:** $v$ is a leaf labelled with either *accept* or *reject*.

An algebraic computation tree $T$ computes a boolean valued function $\mathbb{R}^n \to \{0, 1\}$ as follows. On input $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ we start traversing the tree from the root and take steps towards the leaves. When we reach a computation node, we perform the computation associated with it and proceed to its unique child. When we reach a comparison node, we perform the associated

comparison and depending on its outcome (*true*/*false*) we proceed to the appropriately labelled child. When we reach a leaf we accept or reject according to its label.

Thus, an algebraic computation tree naturally solves the membership problem associated with the set $D \subseteq \mathbb{R}^n$ of points that it accepts.

## 3 Lower bounds for linear decision trees

An often studied problem is the element distinctness problem.

**Definition 1.** In the **Element Distinctness** problem we are given $(x_1, \ldots, x_n) \in \mathbb{R}^n$ as input and we need to decide if $x_i \neq x_j$ for all $i \neq j$.

The set $\mathrm{ED} \subseteq \mathbb{R}^n$ corresponding to the element distinctness problem is

$$\mathrm{ED} = \bigcup_{\pi \in S_n} A_\pi$$

where $S_n$ is the set of all permutations $[n] \to [n]$ and

$$A_\pi = \{x \in \mathbb{R}^n : x_{\pi(1)} < x_{\pi(2)} < \cdots < x_{\pi(n)}\}.$$

In fact, the $A_\pi$'s are precisely the *connected components* of ED. Thus, if we let #ED denote the number of connected components of ED, then

$$\#\mathrm{ED} = |S_n| = n!.$$

More generally, for any decision problem $D \subseteq \mathbb{R}^n$, the quantity $\#D$ is a measure of the complexity of $D$. Dobkin and Lipton [DL75] made this intuition formal.

**Definition 2** (Linear computation trees)**.** An algebraic computation tree is *linear* if all the values $f_v$ computed at the nodes are degree-1 polynomials of $x_1, \ldots, x_n$.

**Theorem 1** ([DL75])**.** *Let $D \subseteq \mathbb{R}^n$. Any linear computation tree for $D$ requires height $\log \#D$.*

*Proof sketch.* Let $\ell$ be a leaf of a linear decision tree. The set of inputs $X_\ell \subseteq \mathbb{R}$ that reach $\ell$ is defined by some set of linear inequalities in the variables $x_i$ determined by the comparison nodes along the path from the root to $\ell$. Thus $X_\ell$ is convex, and in particular *connected*. The set computed by a linear decision tree is a union over all $X_\ell$ where $\ell$ is accepting. But this implies that we need at least one leaf for each component of $D$. This proves the theorem. □

**Corollary 2.** *Any linear computation tree for* ED *requires height* $\Omega(n \log n)$. □

## 4 Lower bounds for non-linear decision trees

The proof of Theorem 1 relied on the property that the sets $X_\ell$ are connected, that is $\#X_\ell \leq 1$. However, if we do not restrict the degree of the polynomials $f_v$ used by the algebraic computation tree, then this property no longer holds. For example, the quadratic inequality

$$x^2 - 1 \geq 0$$

has a solution set in the reals with two connected components.

Fortunately, there is a known upper bound on the number of components of $X$ in case $X$ is defined as the set of common zeroes of a family of polynomials.

**Theorem 3** (Milnor–Thom theorem; see, e.g., [Wal96])**.** *Any $X \subseteq \mathbb{R}^m$ defined as the set of common zeroes of a set of polynomials $\{p_i\}$ where each $p_i$ has degree at most $d$ satisfies*

$$\#X \leq d(2d-1)^{m-1}.$$

Using this estimate Ben-Or [BO83] generalized the result of Dobkin and Lipton.

**Theorem 4** ([BO83])**.** *Any algebraic computation tree for* ED *requires height $\Omega(n \log n)$.*

Note that in an algebraic computation tree of height $k$ we can build up polynomials of degree $2^k$ by repeated squaring. Thus, a direct application of Theorem 3 would not give useful upper bounds on $\#X_\ell$.

The key idea in [BO83] is to control the degree of the polynomial constraints in the algebraic computation tree by *introducing auxiliary variables*. This is the take-home message:

**Key idea.** Increase the number of variables to keep the degree of polynomial constraints low.

Indeed, for each node $v$ we now start regarding its associated value $f_v$ as a real variable—we no longer think of $f_v$ as a polynomial in the original input variables $x_i$. That is, the new variables we consider are

$$\{x_1, \ldots, x_n\} \cup \{f_v : v \text{ is a node}\}.$$

For each operation of a computation node, we associate an accompanying degree-2 equation:

$$\begin{array}{ll} \text{Operation} & \text{Equation} \\ f_v := x_i & f_v = x_i \\ f_v := f_{v_1} \pm f_{v_1} & f_v = f_{v_1} \pm f_{v_2} \\ f_v := f_{v_1} \times f_{v_2} & f_v = f_{v_1} f_{v_2} \\ f_v := f_{v_1} / f_{v_2} & f_v f_{v_2} = f_{v_1} \\ f_v := \sqrt{f_{v_1}} & f_v^2 = f_{v_1} \end{array} \tag{1}$$

Consider some leaf $\ell$ and the associated set $X_\ell \subseteq \mathbb{R}^n$ of inputs that reach it. To involve the new variables $f_v$ we can express $X_\ell$ as

$$X_\ell = \big\{ x \in \mathbb{R}^n : \text{ There exists an assignment to the variables } f_v \text{ that satisfy (1)} \\ \text{and all the comparisons on the path from the root to } \ell. \big\}$$

In other words, $X_\ell$ is a *projection* of some higher dimensional algebraic set $Y_\ell \subseteq \mathbb{R}^m$ where

- the coordinates of $Y_\ell$ are indexed by $m = n + h$ variables

  $$\{x_1, \ldots, x_n\} \cup \{f_v : v \text{ appears on the path from the root to } \ell\},$$

  and $h$ is the height of the tree.

- the constraints for $Y_\ell$ are the degree-2 equations from (1) and the $\leq h$ inequalities of the form $f_v > 0$, $f_v \geq 0$, or $f_v = 0$, that correspond to the comparison nodes on the path from the root to $\ell$.

We can now apply the Milnor–Thom theorem (Theorem 3):

$$\#Y_\ell \leq 2 \cdot 3^{n+h-1}.$$

(Strictly speaking, we should convert the *in*equality constraints of the comparison nodes to equality constraints, which can be achieved by introducing still new variables; see [BO83] for full details.)

Since $X_\ell$ is a projection of $Y_\ell$ we have that $\#X_\ell \leq \#Y_\ell$. Thus, in order for the algebraic computation tree to give rise to the set ED we need to have

$$\#\mathrm{ED} = n! \leq (\#\text{ leaves}) \cdot (\max_\ell \#X_\ell) \leq 2^h \cdot 2 \cdot 3^{n+h-1}.$$

This implies $h = \Omega(n \log n)$ proving Theorem 4.

## References

[BO83] Michael Ben-Or. Lower bounds for algebraic computation trees. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC, pages 80–86. ACM, 1983. doi:10.1145/800061.808735.

[DL75] David P. Dobkin and Richard J. Lipton. On the complexity of computations under varying sets of primitives. In *Automata Theory and Formal Languages*, volume 33 of *Lecture Notes in Computer Science*, pages 110–117. Springer, 1975. doi:10.1007/3-540-07407-4_14.

[Wal96] Nolan R. Wallach. On a theorem of Milnor and Thom. In Simon Gindikin, editor, *Topics in Geometry*, volume 20 of *Progress in Nonlinear Differential Equations and Their Applications*, pages 331–348. Birkhauser Boston, 1996. doi:10.1007/978-1-4612-2432-7_13.