Entropy and Complexity Lower Bounds[*]

Luc Pellissier[1] and Thomas Seiller[2]

1 **LIP, École normale supérieure de Lyon, 46 allée d'Italie, 69364 Lyon, France**
`luc.pellissier@ens-lyon.fr`

2 **Department of Computer Science, University of Copenhagen, Emil Holms Kanal 6 – bygning 24, 2300 København S, Denmark**
`seiller@di.ku.dk`

Abstract

Finding lower bounds in complexity theory has proven to be an extremely difficult task. Nowadays, only one research direction is commonly acknowledged to have the ability to solve current open problems, namely Mulmuley's Geometric Complexity Theory programme. Relying on heavy techniques from algebraic geometry, the latter stemmed from a first lower bound result for a variant of Parallel Random Access Machines (PRAMs).

We analyse this original proof from a semantics point of view, interpreting programs as graphings – generalizations of dynamical systems. We show that Mulmuley's method can be abstracted to a more general setting, exploiting the classic notion of topological entropy. This reformulation recentres the proof around dynamical aspects, relegating the use of algebraic geometry to a model-driven choice rather than a key concept of the method.

1998 ACM Subject Classification F.1.1 Models of Computation; F.1.3 Complexity Measures and Classes; F.3.2 Semantics of Programming Languages

Keywords and phrases Semantics, Complexity Theory, Dynamical Systems, Algebraic Geometry

Digital Object Identifier 10.4230/LIPIcs.DICE.2016.

1 Introduction

The question of *classifying* the complexity classes remains one of the most important question in the field of Complexity Theory. As part of the classification problem, complexity theory has traditionally been concerned with proving *separation results*.

Proving that two classes $B \subset A$ are not equal can be reduced to finding lower bounds for problems in A: by proving that certain problems cannot be solved with less than certain resources on a specific model of computation, one can show that two classes are not equal. Conversely, proving a separation result $B \subsetneq A$ provides a lower bound for the problems that are *A-complete* [6] – i.e. problems that are in some way *universal* for the class A.

Alas, the proven lower bound results are very few, and most separation problems remain as generally accepted conjectures.

The failure of most techniques of proof has been studied in itself, which lead to the proof of the existence of negative results that are commonly called *barriers*. Altogether, these results show that all proof methods we know are ineffective with respect to proving interesting lower bounds. To this day, only one research program aimed at proving new separation results is commonly believed to have the ability to bypass all barriers: Mulmuley's Geometric Complexity Theory (GCT) program [12].

[*] This work was partially supported by the ANR project ELICA (ANR-14-CE25-0005) and the European Commission's Marie Skłodowska-Curie Individual Fellowship (H2020-MSCA-IF-2014) 659920 - ReACT.

1.1 Geometric Complexity Theory (GCT)

Geometric Complexity Theory is widely considered to be a promising research program that might lead to interesting results. It is also widely believed to necessitate new and extremely sophisticated pieces of mathematics in order to achieve its goal. The research program aims to prove the Ptime $\neq$ NPtime lower bound by showing that certain algebraic surfaces (representing the permanent and the discriminant, which are believed [21] to have different complexity if Ptime $\neq$ NPtime) cannot be embedded one into the other. Recently, some negative results [10] have closed the easiest path towards it promised by GCT.

The GCT program was inspired, according to its creators, by a lower bound result obtained by Mulmuley [11] for a specific parallel model (the PRAM without bit operations). This result, despite being the main inspiration of the well-known GCT program, remains seldom cited and has not led to variations applied to other problems. At first sight it relies a lot on algebraic geometric techniques and results, such as the Milnor-Thom theorem[1].

1.2 Semantic ICC

1.2.1 Dynamic Semantics

Geometry of interaction (GOI) is a research program proposed by Girard [8] shortly after the inception of linear logic. In opposition to traditional denotational semantics – e.g. domains –, the GOI program aims at giving an account of programs which also interprets their dynamics, i.e. their execution. This program is well-suited for tackling problems involving computational complexity, and indeed, geometry of interaction's first model was used to prove the optimality of Lamping's reduction in λ-calculus [9]. More recently, a series of characterisations of complexity classes were obtained using GOI techniques [3, 4, 1, 2].

Among the most recent and full-fledged embodiement of this program lie the second author's Interaction Graphs (IG) models [13, 15, 16, 18]. These models, in which programs are interpreted as *graphings* – generalisations of dynamical systems –, encompass all previous GOI models introduced by Girard [18]. In particular, IG models allow for modelling quantitative features of programs [16].

1.2.2 Graphings and Complexity

Based on a study of several Interaction Graphs models characterising complexity classes [17, 19], the second author has proposed a semantic approach to complexity theory [14]. The main intuition behind this proposition is to model and study programs as dynamical systems that acts on a space – thought of as the space of configurations. As dynamical systems are inherently deterministic, the use of graphings is needed to extend the approach with, e.g. probabilities, non-determinism. One can then study a program through the geometry of the associated graphing.

1.3 Graphings and Lower Bounds

The present work reports on a study of Mulmuley's geometric proof of lower bounds for the model of PRAM without bit operations [11] through the prism of graphings.

[1] Let us here notice that, even though this is not mentionned by Mulmuley, the Milnor-Thom theorem was already used to prove lower bounds, c.f. papers by Dobkin and Lipton [7], Steele and Yao [20], Ben-Or [5], and references therein.

- $\alpha(\texttt{plus}(i))$ is the map $f \mapsto P_i \circ f$, with $P : (x_1, \dots, x_d) \mapsto (x_1, \dots, x_{i-1}, x_i + 1, x_{i+1} \dots, x_d)$;
- $\alpha(\texttt{minus}(i))$ is the map $f \mapsto M_i \circ f$, with $M : (x_1, \dots, x_d) \mapsto (x_1, \dots, x_{i-1}, x_i - 1, x_{i+1} \dots, x_d)$;
- $\alpha(\texttt{copy}(i, j))$ is the map $f \mapsto C_{i,j} \circ f$, with $C_{i,j} : (x_1, \dots, x_d) \mapsto (x_1, \dots, x_{i-1}, x_j, x_{i+1} \dots, x_d)$;
- $\alpha(\texttt{copy}(i, \sharp j))$ is the map $f \mapsto CR_{i,j} \circ f$, with $CR_{i,j} : (x_1, \dots, x_d) \mapsto (x_1, \dots, x_{i-1}, x_{x_j}, x_{i+1} \dots, x_d)$;
- $\alpha(\texttt{copy}(\sharp i, j))$ is the map $f \mapsto RC_{i,j} \circ f$, with $RC_{i,j} : (x_1, \dots, x_d) \mapsto (x_1, \dots, x_{x_i - 1}, x_j, x_{x_i + 1} \dots, x_d)$.

Figure 1 Generators for the AMC of Random Access Machines.

When studying Mulmuley's proof, the authors quickly realised that the techniques involved are somehow of a semantic nature and that algebraic geometry does not play the essential role Mulmuley seems to believe it did (based on the fact that GCT is heavily founded upon algebraic geometry). More importantly, it became clear that the proof could be seen as a study of the geometry of the graphing interpretation of PRAM machines, compared to the geometry of the maxflow problem.

Viewed in this light, a recast of Mulmuley's proof in the language of graphings provides a first useful insight about its geometric contents. It provides a more general and flexible method for interpreting programs geometrically, as most of the method can be followed for arbitrary dynamical systems (equivalently, deterministic graphings). Lastly, it shows the use of methods from algebraic geometry is but an implicit choice taken by Mulmuley (guided by the specific PRAM model considered).Other choices could allow for the use of methods from e.g. differential geometry, greatly widening the scope of geometric methods one could hope to use to tackle lower bound results.

2 Contributions.

2.1 Abstract Programs

▶ **Definition 1.** An *abstract model of computation* (AMC) is defined as a triple (G, R, α), where $\langle G, \mathrm{R}\rangle$ is a presentation of a monoid $\mathrm{M}\langle G, \mathrm{R}\rangle$ and α is a monoid action $\mathrm{M}\langle G, \mathrm{R}\rangle \curvearrowright \mathbf{X}$.

▶ **Definition 2.** An *abstract program A* within an AMC $\alpha : \langle G, \mathrm{R}\rangle \curvearrowright \mathbf{X}$ is defined as a finite set $\mathbf{S^A}$ of *control states* and a graphing [18] G^A w.r.t. the monoid action $\mathrm{M}\langle G, \mathrm{R}\rangle \times \mathfrak{S}_k \curvearrowright \mathbf{X} \times \mathbf{S^A}$. An abstract program is *deterministic* if its underlying graphing is deterministic.

First, we show how the second author's notion of *graphing* can adequately interpret random access machines (RAM). For this, we consider the AMC induced by the action α on the space $\theta : \mathbf{Z}^d \to \mathbf{Z}^\omega \times \mathbf{Z}^\omega$ defined on a set of generators as shown in Figure 1. The intuition here is that a map $\theta : \mathbf{Z}^d \to \mathbf{Z}^\omega \times \mathbf{Z}^\omega$ is a *reparametrisation of the input*: for each element $\vec{a} \in \mathbf{Z}^d$, $\theta(\vec{a})$ represents a configuration of the memory blocks. This technical artefact should not hide that the interpretation of a RAM machine acts in fact on $\mathbf{Z}^\omega \times \mathbf{Z}^\omega$ and this action is then lifted to the space of maps by precomposition. The fact that we consider two copies of $\mathbf{Z}^\omega$ eases the definition of PRAMs: the left-hand copy represents shared registers while the right-hand copy represents private registers.

2.2 The crew operation

We then introduce a purely algebraic technique, related to the notion of *amalgamated free product of groups* to deal with parallelisation of abstract models of computation.

▶ **Definition 3** (Noncommutative Product). Let $\mathrm{M}\langle G, \mathrm{R}\rangle$ and $\mathrm{M}\langle G', \mathrm{R}'\rangle$ be representations of monoids, and let $H \subseteq G$ and $H' \subseteq G'$ be subsets of generators. We define the noncommutative product of $\mathrm{M}\langle G, \mathrm{R}\rangle$ and $\mathrm{M}\langle G', \mathrm{R}'\rangle$ above H, H' as the monoid $\mathrm{M}\langle G \times$

$G', \mathrm{R}\times\mathrm{Id}\cup\mathrm{Id}\times\mathrm{R}'\cup\mathrm{Q}\rangle$ where Q is defined as $\{(a,a')(b,b')=(aa',bb')\mid a,a'\not\in H \text{ or } b,b'\not\in H'\}$. The resulting monoid is denoted $\mathrm{M}\langle G,\mathrm{R}\rangle\ {}_{H}\!\times_{H'}\ \mathrm{M}\langle G',\mathrm{R}'\rangle$.

Given a monoid action α of a monoid M on a space $\mathbf{X}\times\mathbf{Y}$, we define the set $Z_\alpha(M)$ of *central elements* of M, i.e. elements that act as the identity on $\mathbf{X}$: $\alpha(m);\pi_X=\alpha(m)$. We write $\bar{Z}_\alpha(M)$ the set of non-central elements.

By considering the noncommutative product of AMCs above the sets of non-central elements, one can define the CREW operation, which represents the parallelisation of computational models w.r.t. the the *concurrent read, parallel write* discipline.

▶ **Definition 4** (The CREW operation)**.** Let $\alpha:\mathrm{M}\langle G,\mathrm{R}\rangle\curvearrowright\mathbf{X}\times\mathbf{Y}$ and $\beta:\mathrm{M}\langle H,\mathrm{Q}\rangle\curvearrowright\mathbf{X}\times\mathbf{Z}$ be AMCs. We define the AMC $\mathrm{CREW}(\alpha,\beta):\mathrm{M}\langle G,\mathrm{R}\rangle\ {}_{\bar{Z}_\alpha(G)}\!\times_{\bar{Z}_\beta(G')}\ \mathrm{M}\langle G',\mathrm{R}'\rangle\curvearrowright\mathbf{X}\times\mathbf{Y}\times\mathbf{Z}$. by letting $\mathrm{CREW}(\alpha,\beta)(m,m')=\alpha(m)*\beta(m')$ on elements of $G\times G'$, where $\alpha(m)*\beta(m')$ is defined as (here $\Delta:(x,y,z)\mapsto(x,y,x,z):\mathbf{X}\times\mathbf{Y}\times\mathbf{Z}\to\mathbf{X}\times\mathbf{Y}\times\mathbf{X}\times\mathbf{Z}$):

$$\alpha(m)*\beta(m') \quad = \quad \begin{cases} \Delta;[\alpha(m);\pi_Y,\beta(m')] & \text{if } m\not\in\bar{Z}_\alpha(G), m'\in\bar{Z}_\beta(G');\\ \Delta;[\alpha(m),\beta(m');\pi_Z] & \text{otherwise}\end{cases}$$

▶ **Definition 5** (The AMC of PRAMs)**.** Let $\alpha: M\curvearrowright X\times X$ be the AMC of RAMs. The AMC of PRAMs is defined as $\varinjlim \mathrm{CREW}^k(\alpha)$, where $\mathrm{CREW}^{k-1}(\alpha)$ is identified with a restriction of $\mathrm{CREW}^k(\alpha)$ through $\mathrm{CREW}^{k-1}(\alpha)(m_1,\dots,m_{k-1})\mapsto\mathrm{CREW}^k(\alpha)(m_1,\dots,m_{k-1},1)$.

▶ **Theorem 6.** *The representation of* PRAMs *as graphings is sound.*

2.3 Entropy

We then explain how a graphing induces geometric decompositions of the space it acts on, describing the orbits – the computational traces – of a point through iterations of the graphing. The geometric decomposition obtained after k iterations is called the *k-th cell decomposition* $\mathrm{CELL}(k)$ of the space w.r.t. the graphing. We generalise the notion of topological entropy, which quantifies the exponential growth of the number of orbits of a dynamical system, to define the entropy of a graphing. We then give bounds on the number of cells of the decomposition as well as on the number of varieties delimiting them (under additional hypotheses).

▶ **Theorem 7.** *Let G be a deterministic graphing with entropy $h(G)$. The cardinality $c(k)$ of $\mathrm{CELL}(k)$ is asymptotically bounded by $g(k)=2^k2^{h([G])}$, i.e. $c(k)=O(g(k))$.*

▶ **Theorem 8.** *Let G be a regular*[2] *deterministic graphing interpreting a* PRAM *with p processors. Then $\mathrm{CELL}(k)$ is determined by at most 2^kp^k algebraic varieties.*

2.4 Lower Bounds

After that, we show how to associate a partition of a space to a decision problem obtained from a parametrization of an optimization problem. In particular, the maxflow problem can be described in such a way. Although the geometry of the problem is fairly simple, we show that a machine deciding it would need to induce a decomposition of the space whose *complexity*[3] satisfy a certain relation w.r.t. the complexity of the problem, the number of processors, and the computation time.

[2] Regular graphings are those that preserve algebraic varieties under reverse images.
[3] The complexity of a decomposition is here measured by the variations of the varieties that delimitates it.

After having defined a variation on the PRAM model allowing to have finer complexity bounds, we finally show Mulmuley's result, i.e. a lower bound for the MAXFLOW problem.

▶ **Theorem 9.** *Let G be a deterministic graphing interpreting a* PRAM *without bit operations with $2^{O(N^c)}$ processors, where N is the length of the inputs and c any positive integer. Then G does not decide maxflow in $O(N^c)$ steps.*

3 Conclusion

This reformulation of Mulmuley's proof shows that tools from algebraic geometry are only used to get some bounds on the number of intersections of the surfaces. The essence of the proof lies elsewhere, in the study of the geometry of a dynamical system. The use of methods from algebraic geometry is dictated by the presence of polynomials, a consequence of the fact that PRAMs without bit operations manipulate numbers and not strings of bits.

Our approach is robust in that the cells decomposition induced by a graphing can be studied, and bounds on its cardinality computed, even when this decomposition is not naturally delimited by algebraic surfaces.

References

1 C. Aubert, M. Bagnol, P. Pistone, and T. Seiller. Logic programming and logarithmic space. In *APLAS*, 2014.
2 C. Aubert, M. Bagnol, and T. Seiller. Unary resolution: Characterizing ptime. In *FOSSACS*, 2016.
3 C. Aubert and T. Seiller. Characterizing co-nl by a group action. *Mathematical Structures in Computer Science*, 26, 2016.
4 C. Aubert and T. Seiller. Logarithmic space and permutations. *Inf. Comp.*, 248, 2016.
5 M. Ben-Or. Lower bounds for algebraic computation trees. In *STOC*, 1983.
6 S. Cook. The complexity of theorem-proving procedures. In *STOC*, 1971.
7 D. Dobkin and R. Lipton. Multidimensional searching problems. *SIAM J. Comp.*, 5, 1976.
8 J.-Y. Girard. Towards a Geometry of Interaction. *Contemporary Mathematics*, 92, 1989.
9 G. Gonthier, M. Abadi, and J.-J. Lévy. The geometry of optimal lambda reduction. In *Proc. 19th ACM Symposium on Principles of Programming Languages*, 1992.
10 C. Ikenmeyer and G. Panova. Rectangular kronecker coefficients and plethysms in geometric complexity theory. *Advances in Mathematics*, 319, 2017.
11 K. Mulmuley. Lower bounds in a parallel model without bit operations. *SIAM J. Comp.*, 28, 1999.
12 K. D. Mulmuley. The gct program toward the p vs. np problem. *Commun. ACM*, 55, 2012.
13 T. Seiller. Interaction graphs: Multiplicatives. *Ann. of Pure and Applied Logic*, 163, 2012.
14 T. Seiller. Towards a *Complexity-through-Realizability* theory. Arxiv:1502.01257, 2015.
15 T. Seiller. Interaction graphs: Additives. *Ann. of Pure and Applied Logic*, 167, 2016.
16 T. Seiller. Interaction graphs: Full linear logic. In *IEEE/ACM LICS*, 2016.
17 T. Seiller. Interaction graphs: Nondeterministic automata. Under revision, 2016.
18 T. Seiller. Interaction graphs: Graphings. *Ann. of Pure and Applied Logic*, 168, 2017.
19 T. Seiller. Interaction graphs: Probabilistic automata. In preparation, 2018.
20 J. M. Steele and A. Yao. Lower bounds for algebraic decision trees. *J. Algorithms*, 3, 1982.
21 L. G. Valiant. The complexity of computing the permanent. *Th. Comp. Sci.*, 8, 1979.