Semantics, Entropy and Complexity Lower Bounds

Luc Pellissier
LIP
École normale supérieure de Lyon
Lyon, France
luc.pellissier@ens-lyon.fr

Thomas Seiller†
Department of Computer Science
University of Copenhagen
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 – generalisations 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 recovers the proof around dynamical aspects, relegating the use of algebraic geometry to a model-driven choice rather than a key concept of the method.

Keywords Semantics, Complexity Theory, Dynamical Systems, Algebraic Geometry

1 Introduction
While the general theory of computability focused on studying what a computable function is, computer scientists quickly realised that this notion was not meaningful in practice. Indeed, one can always define a computable function \( f \) such that no computer could compute the values of \( f \) for two-digits inputs within the next, say, ten years.

This lead researchers to work on the definition and understanding of the notion of feasible computation, i.e. characterise a set of functions which can be effectively computed. Within the span of a single year three different papers [12, 16, 23] tackled this question, and all of them provided the same answer, namely feasible functions are those functions for which there exists a program whose running time is asymptotically bounded by a polynomial in the input. This is how the first complexity class was born: the class of polynomial-time computable functions.

Very quickly, other classes were defined, some of them considering constraints on space rather than time. The question of classifying the complexity classes became one of the main question in the field, and a number of important results were obtained within the first years.

Lower bounds. As part of the classification problem, complexity theory has traditionally been concerned with proving separation results. Among the numerous open separation problems lies the much advertised \( \text{Ptime} \) vs. \( \text{Nptime} \) problem of showing that some problems considered hard to solve but efficient to verify do not have a polynomial time algorithm solving them.

Proving that two classes \( B \subseteq A \) 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 \( B \subseteq A \) provides a lower bound for the problems that are \( A \)-complete [14] – 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. For instance, a proof that the class of non-deterministic exponential problems is not included in what is thought of as a very small class of circuits was not achieved until very recently [45].

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. Indeed, there are three barriers: relativisation [8], natural proofs [34] and algebraization [1], and every known proof method hits at least one of them. This shows the need for new methods. However, 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 and Sohoni’s Geometric Complexity Theory (gct) program [32].

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 \( \text{Ptime} \neq \text{Nptime} \) lower bound by showing that certain algebraic surfaces (representing the permanent and the discriminant, which are believed [44] to have different complexity if \( \text{Ptime} \neq \text{Nptime} \)) cannot be embedded one into the other. Although this program has lead to interesting developments as far as pure mathematics is concerned, it has not enhanced our understanding of complexity lower bounds for the time being (actually, even for Mulmuley himself, such understanding will not be achieved in our lifetimes [18]). Recently, some negative results [26] 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 [31]. Specifically, it was proved that the maxflow problem (deciding whether a certain quantity can flow from a source to a target in a weighted graph) is not solvable efficiently in a specific parallel model (the \( \text{pram} \) without bit operations). The maxflow problem is quite interesting as it is known to be \( \text{NP} \)-hard [17], but there are no known

\[ \text{P} = \text{NP} \]
efficiently parallel algorithm solving it. This lower bound proof, despite being the main inspiration of the well-known GCT research 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.

**Implicit Computational Complexity.** Another approach to complexity theory that emerged in the recent years is Implicit Computational Complexity (ICC). Related to logical approaches of computational complexity such as Descriptive Complexity, the aim of ICC is to study algorithmic complexity only in terms of restrictions of languages and computational principles. It has been established since Bellantoni and Cook’s landmark paper [9], and following work by Leivant and Marion [29, 30].

As part of ICC techniques, some approaches derive from the proofs-as-programs (or Curry–Howard) correspondence. At its core, this correspondence allows one to view the execution of a program as the cut-elimination procedure of a corresponding proof in a formal deductive system (e.g. sequent calculus). Initially stated for intuitionistic logic [25], the correspondence extends to resource-aware logics such as linear logic (LL), which is well-suited to study computation. This approach to ICC therefore relies on restrictions on the deductive system considered to characterise complexity classes. In particular, several variants of LL were shown to characterise \( \text{FPtime}^3 \): BLL [21], SLL [28], DLAL [7] and LLL [20].

**Dynamic Semantics** The geometry of interaction program was proposed by Girard [19] 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 the proofs and programs which also interprets their dynamical features, i.e. cut-elimination/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 \( \lambda \)-calculus [22]. More recently, a series of characterisations of complexity classes were obtained using goi techniques [3–6].

Among the most recent and full-fledged embodiment of this program lie the second author’s Interaction Graphs models [35, 37–39]. These models, in which proofs/programs are interpreted as graphings — generalisations of dynamical systems —, encompass all previous goi models introduced by Girard [39]. In particular, Interaction Graphs allow for modelling quantitative features of programs/proofs [38].

**Semantic Approach to Complexity** Based on a study of several Interaction Graphs models characterising complexity classes [40, 42], the second author has proposed to use graphings to develop a semantic approach to complexity theory [36]. The basic idea behind this program 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 to probabilistic and/or non-deterministic programs. One can then study a program through the geometry of the associated graphing (for instance, a configuration caught in a loop is represented as a point of the space of finite orbit).

The second author conjectures that advanced methods developed within the theory of dynamical systems, in particular methods specific to the study of ergodic theory using techniques from operator algebras, could enable new proof techniques for separation. It can be argued that such techniques should be able to bypass barriers [41].

**Contributions.** The present work reports on the first investigations into how the interpretation of programs as graphings could lead to separation techniques. As the second author’s program shares with Mulmuley’s GCT the involved geometric nature of the techniques, we turned to Mulmuley’s geometric proof of lower bounds for the model of PRAM without bit operations. 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 action of programs expressible in the PRAM model, compared to the geometry of the maxflow problem. Viewed in this light, a recast of this 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.

**Plan of the paper.** After a first section recalling some definitions and establishing notations, we propose a quick introduction to the second author’s program of representing programs as graphings. We then explain how random access machines (RAM) can be interpreted in this way. The next section introduces a purely algebraic technique to deal with parallelisation of abstract models of computation, which is then used to produce interpretations of PRAMS as graphings.

We then show how a graphing induces a geometric decomposition of the space it acts on, that describes the final state a point of the space is mapped to by the graphing. After introducing the notion of entropy of a deterministic graphing (which is but the generalization of the entropy of a dynamical system), we give bounds on the number of cells of the decomposition as well as on the number of varieties delimiting them (under additional hypotheses).

After that, we show how to associate a partition of a space to a decision problem obtained from a parametrization of an optimisation problem. The maxflow problem can be described in such a way. Although the geometry of the problem is fairly simple, we show in Section 7 that a machine deciding this problem would need to induce a decomposition of the space whose complexity satisfies a

---

2 Let us here notice that, even though this is not mentioned by Mulmuley, the Milnor-Thom theorem was already used to prove lower bounds, c.f. papers by Dokkin and Lipton [15], Steele and Yao [43], Ben-Or [10], and references therein.

3 \( \text{FPtime} \) is a variant of \( \text{Ptime} \) that computes a function and not just a boolean predicate.

4 The complexity of a decomposition is here measured by the variations of the varieties that delimitates it.
certain relation w.r.t. the complexity of the problem, the number of processors, and the computation time.

After having defined a variation on the PRAM model allowing to have finer complexity bounds, we finally show and discuss Mulumley’s result.

2 Preliminaries and Notations

Monoid and Actions. Given a set $G$, we denote by $M(G)$ the free monoid on $G$, i.e. the set of finite sequences of elements of $G$.

Definition 1. We recall that a presentation $(G, R)$ of a monoid $M$ is given by a set $G$ of generators and a set $R$ of relations such that $M$ is isomorphic to $M(G)/R$.

Definition 2. Let $M$ be a monoid and $X$ be a space. An action of $M$ on $X$ is a monoid morphism $\alpha$ from $M$ to the set of endomorphisms on $X$. We denote actions by $\alpha: M \acts X$, sometimes omitting the morphism $\alpha$.

In this definition, we purposely chose not to specify the kind of space considered. As a consequence, if one considers a discrete space $X$ (i.e. sets), the set of endomorphisms will simply be the set of functions $X \rightarrow X$. Similarly, if $X$ is a topological space, the set of endomorphisms will be continuous maps (hence $\alpha$ will be a continuous action). Etc.

Random Access Machines (SRAMs). In this paper, we will consider parallel random access machines. In order to define those properly, we first define the notion of (sequential) random access machine (RAM) before considering their parallelisation.

A RAM command is a pair $(\ell, I)$ of a label (or line) $\ell \in N^*$ and a command $I$ among the following:

- Command $\text{exec}$: $X_i := X_j$,
- Command $\text{if}$: if $X_i = 0$ goto $\ell$ else $\ell'$,
- Command $\text{goto}$: $\ell'$,

where $i, j \in N$ and $\ell, \ell' \in N^*$ are labels.

A RAM machine $M$ is then a finite set of commands such that the set of labels is in bijection with an initial segment of $N^*$. The cardinality of the set of commands is called the length of $M$ and denoted $|M|$.

Without loss of generality, we can suppose that the set of labels is equal to $\{1, 2, \ldots, |M|\}$. We will denote the commands in $M$ by $(i, \text{Inst}_M(i))$. $\text{Inst}_M(i)$ denotes the $i$-labelled command.

The semantics of SRAM machines is quite straightforward: a configuration is represented as a pair $(i, \sigma)$ where $i$ is the current label and $\sigma: Z \rightarrow Z$ an eventually null function $\sigma$ represents the state of the registers. Then the commands above are easily interpreted as a transition to the label $i + 1$ (except in case of the unconditional and an updated function $\sigma$. E.g. a $\text{exec}$ command induces the following transition: $(i, \sigma) \rightarrow (i + 1, \sigma[\sigma(j)/\sigma(i)])), while a $\text{if}$ command induces the transition $(i, \sigma) \rightarrow (i + 1, \sigma[\sigma(j)/i])$.

Parallel Random Access Machines (PRAMS). Based on the notion of SRAM, we are now able to consider their parallelisation, namely PRAMS. A PRAM machine $M$ is simply given as a finite sequence of SRAM machines $M_1, \ldots, M_p$, where $p$ is the number of processors of $M$. Each processor $M_i$ has access to its own, private, set of registers $\{X_k^i\}_{k \geq 0}$ and a shared memory represented as a set of registers $\{Y_k\}_{k \geq 0}$.

As usual, one has to deal with conflicts when several processors try to access the shared memory simultaneously. We here chose to work with the CREW discipline, i.e. Concurrent Read, Exclusive Write, implemented as follows: at a given step at which several processors try to write in the shared memory, only the processor with the smallest index will be allowed to do so.

In order to express complexity results, we will be led to introduce PRAMs without bit operations, machines akin to PRAM but that act on strings of bits and not integers. They deserve the moniker “without bit operations” as they can not access arbitrary bits stored in their memories (actually, this is an hybrid model, where numbers are stored as binary strings, but manipulated as numbers).

Additional notations. We also fix a number of notations that we will use throughout the paper. First, we will write composition of functions as $f \circ g$ instead of $g \circ f$.

3 The SRAM action

In this section, we will define a specific monoid action $\text{sramp}$ and show that SRAM machines correspond to (deterministic) graphings w.r.t. this monoid action. We first recall what graphings are and how they are used here as mathematical models of computation. Then we introduce the SRAM action and show how machines are represented as graphings.

3.1 Abstract Models of Computation

The basic intuitions here can be summarised by the following slogan: “Computation, as a dynamical process, can be modelled as a dynamical system”. Of course, the above affirmation cannot be true of all computational processes; for instance the traditional notion of dynamical system is deterministic. In practice, one works with a generalisation of dynamical systems named graphings; introduced as part of a family of models of linear logic, graphings have been shown to model non-deterministic and probabilistic computation.

Definition 3. An abstract model of computation (AMC) is defined as a triple $(G, R, \alpha)$, where $(G, R)$ is a presentation of a monoid $M(G, R)$ and $\alpha$ is a monoid action $M(G, R) \acts X$. We denote an AMC as $\alpha: (G, R) \acts X$.

Remark. Although it might seem enough to define an abstract model of computation solely as a monoid action, the choice of a presentation of the monoid by generators and relations is important. First, when considering several models of computation, one wants to consider a notion of compilation: an element $f \in \text{End}(Y)$ is compilable in the action $A \acts X$ when there is an automorphism $\Psi: X \rightarrow Y$ such that one can write $X = \psi_1 \circ \ldots \circ \psi_n X_1$ and the restriction of $\Psi^{-1} \circ f \circ \Psi$ to $X_1$ is the restriction of an element $g_1$ of $M$ to $X_1$. To use this notion of compilation in a meaningful way, one would want to quantify the complexity of compilation. This can be done only by considering a definition of the monoid $M$ as generators and relations $(G, R)$, allowing one to consider the degree of $g_1$ – the length of the smallest word in $G^*$ representing $g_1$ – and therefore the degree of the compilation of $f$ into $\alpha: (G, R) \acts X$.

\(^1\)Let us notice that this notion has already been considered in relation to dynamical systems, used to define what is called the algebraic entropy and the fundamental group entropy [27, Section 3.1].
Although we will not consider the notion of compiliation in this work, it is remarkable that the representation of the monoid as generators and relations is needed in the definition of the parallelisation of actions – defined as the crew operation.

**Definition 4.** A graphing representative \( G \) w.r.t. a monoid action \( M \rightrightarrows X \) is defined as a set of edges \( E^G \) and for each element \( e \in E^G \) a pair \( (S^G_e, m^G_e) \) of a subspace \( S^G_e \) of \( X \) – the source of \( e \) – and an element \( m^G_e \in M \) – the realiser of \( e \).

Graphings come in different flavours (discrete, topological, measurable), depending on the type of space \( X \) one wishes to consider. If \( X \) is a topological space, the action will be continuous, if \( X \) is a measure space, the action will be measurable. While the notion of graphing representative does not depend on this choice, the notion of graphing is defined as a quotient of the space of graphing representatives w.r.t. an adequate notion of equivalence. We will here consider the notion of topological graphing \cite{PellissierSeiller2016}, which we will simply call graphings. In this case, the notion of equivalence is easier to define than in the case of measurable graphings as the latter requires one to consider almost-everywhere equality.

**Definition 5 (Refinement).** Let \( F, G \) be graphing representatives. Then \( F \) is a refinement of \( G \), noted \( F \leq G \), when there exists a bijection \( \theta : S^F \to S^G \) and an \( E^G \)-indexed partition \( \{E^F_g\}_{g \in E^G} \) of \( E^F \) such that for all \( g \in E^G \), the family \( \{S^F_f\}_{f \in E^F_g} \) is a partition of \( S^G_g \) and for all \( f \in E^F_g \) we have \( m^G_g = m^F_f \).

This notion defines an equivalence relation defined by \( F \sim G \) if and only if there exists \( H \) with \( H \leq F \) and \( H \leq G \).

**Definition 6.** A graphing is an equivalence class of graphing representatives w.r.t. the equivalence relation generated by refinements. A graphing is deterministic if its representatives are deterministic, i.e. if any representative \( G \) is such that for all \( x \in X \) there is at most one \( e \in E^G \) such that \( x \in S^G_e \).

**Definition 7.** An abstract program \( A \) within an AMC \( \alpha : (G, R) \rightrightarrows X \) is defined as a finite set \( S^A \) of control states and a graphing \( G^A \) w.r.t. the monoid action \( M(G, R) \times \mathbb{S}_A \rightrightarrows X \times S^A \).

An abstract program is deterministic if its underlying graphing is deterministic.

### 3.2 The Crew action

We now define the crew action. The underlying space is the space of functions \( X = \mathbb{Z}^d \to \mathbb{Z}^{\omega} \times \mathbb{Z}^{\omega} \); it is thought of as the result of applying a sequence of instructions to an initial configuration defined from \( \mathbb{Z}^d \). The set of generators is defined following the possible actions of a crew action on the memory: \( \text{plus}(i), \text{minus}(i), \text{copy}(i,j), \text{copy}(j,i) \). The action \( \alpha \) is defined on the generators as follows:

- \( \alpha(\text{plus}(i)) \) is the map \( f \mapsto P_i \circ f \), with \( P_i : (x_1, \ldots, x_d) \mapsto (x_1, \ldots, x_{i-1}, x_i+1, x_{i+1}, \ldots, x_d) \);
- \( \alpha(\text{minus}(i)) \) is the map \( f \mapsto M_i \circ f \), with \( M_i : (x_1, \ldots, x_d) \mapsto (x_1, \ldots, x_{i-1}, x_i-1, x_{i+1}, \ldots, x_d) \);
- \( \alpha(\text{copy}(i,j)) \) is the map \( f \mapsto C_{i,j} \circ f \), with \( C_{i,j} : (x_1, \ldots, x_d) \mapsto (x_1, \ldots, x_{i-1}, x_j, x_{i+1}, \ldots, x_d) \);
- \( \alpha(\text{copy}(j,i)) \) is the map \( f \mapsto R_{i,j} \circ f \), with \( R_{i,j} : (x_1, \ldots, x_d) \mapsto (x_1, \ldots, x_{j-1}, x_j, x_{j+1}, \ldots, x_d) \);
- \( \alpha(\text{copy}(i,j)) \) is the map \( f \mapsto M_{i,j} \circ f \), with \( M_{i,j} : (x_1, \ldots, x_d) \mapsto (x_1, \ldots, x_{i-1}, x_j, x_{i+1}, \ldots, x_d) \);
Definition 10. Let $\alpha : M \to X \times Y$ be a monoid action. We say that an element $m \in M$ is central relatively to $\alpha$ (or just central) if the action of $m$ commutes with the first projection $\pi_X : X \times Y \to X$, i.e. $\alpha(m) \cdot \pi_X = \alpha(m)$; in other words $m$ acts as the identity on $X$.

Intuitively, central elements are those that will not affect the shared memory. As such, they do not raise any issues when the processes are put in parallel. On the other hand, non-central elements need to be dealt with care.

Definition 11. Let $M(G, R) \to X \times Y$ be an AMC. We note $Z_{\alpha}$ the set of central elements and $Z_{\alpha}(G)$ the set $\{m \in G \mid n \not\in Z_{\alpha}\}$.

We will now define the crew operation on AMC. We first define an operation on monoids that will allow for the definition of the crew action later on.

Definition 12 (Noncommutative Product). Let $M(G, R)$ and $M(G', R')$ be representations of monoids, and let $H \subseteq G$ and $H' \subseteq G'$ be subsets of generators. We define the noncommutative product of $M(G, R)$ and $M(G', R')$ above $H, H'$ as the monoid $M(G \times G', R \times \text{Id} \cup \text{Id} \times R' \cup \Omega)$ where $Q$ is defined as:

$$Q = \{(a, a')(b, b') = (aa', bb') \mid a, a' \not\in H \text{ or } b, b' \not\in H'\}.$$

The resulting monoid is denoted $M(G, R) \rtimes_{H, H'} M(G', R')$.

Remark. When both $H, H'$ are empty, this defines the usual product of monoids. This was to be expected. Indeed, one should think of $H, H'$ as representing the set of elements that are not central, i.e. that write on the shared memory. As a consequence, when $H, H'$ are empty no conflicts can arise w.r.t. the shared memory. In other words, the product of monoids corresponds to the parallelisation of processes without shared memory.

Definition 13 (The crew operation). Let $\alpha : M(G, R) \to X \times Y$ and $\beta : M(H, Q) \to X \times Z$ be AMCs. We define the AMC

$$\text{crew}(\alpha, \beta) : M(G, R) \rtimes_{H, H'} M(G', R') \to X \times Y \times Z$$

by letting $\text{crew}(\alpha, \beta)(m, m') = \alpha(m) \cdot \beta(m')$ on elements of $G \times G'$, where $\alpha(m) \cdot \beta(m')$ is defined as:

$$\alpha(m) \cdot \beta(m') = \begin{cases} \Delta; \{\alpha(m), \beta(m')\} & \text{if } m \not\in Z_{\alpha}(G), m' \in Z_{\beta}(G'); \\ \Delta; \{\alpha(m), \beta(m'); \pi_Z\} & \text{otherwise} \end{cases}$$

with $\Delta : (x, y, z) \mapsto (x, y, x, z) : X \times Y \times Z \to X \times Y \times X \times Z$.

We now need to check that we defined the operation on monoids and the action coherently. In other words, that the previous operation is compatible with the quotient by the adequate relations, i.e. it does define a monoid action.

Lemma 14. The crew operation on AMCs is well-defined.

Proof. We check that the definition above extends to a monoid action, i.e. the mapping just defined is coherent with the relations introduced in Definition 12. To see this, it is sufficient to notice that only the relations in the set $Q$ may lead to incoherences. But those are exactly the relations of the type $(m, m')(n, n') = (mm', nn')$ for which $(\alpha(m) \cdot \beta(m'))(n, n') = (\alpha(mn) \cdot \beta(m'n'))$. For instance, one can easily check that if $m', n' \not\in Z_{\beta}(G')$ the maps $\Delta; \{\alpha(m), \beta(m'); \pi_Z\}$, $\Delta; \{\alpha(n), \beta(n'); \pi_Z\}$ and $\Delta; \{\alpha(mn), \beta(m'n'); \pi_Z\}$ are equal. \qed

4.2 The crew of the SRAM action

We can now define the interpretations of PRAMS as abstract programs. For each integer $p$, one can define the AMC crew$^k(\alpha)$ for $\alpha$ is the AMC for SRAMS defined in the previous section. This allows the consideration of up to $p$ parallel SRAMS. The interpretation of such a SRAM with $p$ processors is then defined by considering a set of states equal to $L_1 \times L_2 \times \cdots \times L_p$ where for all $i$ the set $L_i$ is the set of labels of the $i$-th processor.

Now, to deal with arbitrary large PRAMS, i.e. with arbitrarily large number of processors, one considers the following AMC defined as a direct limit.

Definition 15 (The AMC of PRAMS). Let $\alpha : M \to X \times X$ be the SRAMAMC$^p$. The AMC of PRAMS is defined as $\lim \text{crew}^{k}(\alpha)$, where crew$^{k-1}(\alpha)$ is identified with a restriction of crew$^k(\alpha)$ through crew$^{k-1}(\alpha)(m_1, \ldots, m_{k-1}) \mapsto \text{crew}^k(\alpha)(m_1, \ldots, m_{k-1})$.

The fact that the crew operation on actions does define the semantics of the CREW discipline for concurrent read/write should be clear from the definitions. As a consequence, Theorem 9 the interpretation of PRAMS as graphings is sound.

Theorem 16. The representation of PRAMS as graphings is sound.

5 Entropy and Cells

5.1 Topological Entropy

Topological Entropy was introduced in the context of dynamical systems in an attempt to classify the latter with respect to conjugacy. The topological entropy of a dynamical system is a value representing the average exponential growth rate of the number of orbit segments distinguishable with a finite (but arbitrarily fine) precision. The definition is based on the notion of open covers.

Open covers. Given a topological space $X$, an open cover of $X$ is a family $U = (U_i)_{i \in I}$ of open subsets of $X$ such that $\bigcup_{i \in I} U_i = X$. A finite cover $U$ is a cover whose indexing set is finite. A subcover of a cover $U = (U_i)_{i \in I}$ is a sub-family $S = (U_j)_{j \in J}$ for $J \subseteq I$ such that $S$ is a cover, i.e. such that $\bigcup_{j \in J} U_j = X$.

We will denote by Cov$(X)$ (resp. FCov$(X)$) the set of all open covers (resp. all finite open covers) of the space $X$.

We now define two operations on open covers that are essential to the definition of entropy. An open cover $U = (U_i)_{i \in I}$, together with a continuous function $f : X \to X$, defines the inverse image open cover $f^{-1}(U) = (f^{-1}(U_i))_{i \in I}$. Note that if $U$ is finite, $f^{-1}(U)$ is finite as well. Given two open covers $U = (U_i)_{i \in I}$ and $V = (V_j)_{j \in J}$, we define their join $U \vee V$ as the family $(U_i \cap V_j)_{(i, j) \in I \times J}$. Once again, if both initial covers are finite, their join is finite.

Entropy. Usually, entropy is defined for continuous maps on a compact set, following the original definition by Adler, Konheim and McAndrews [2]. Using the fact that arbitrary open covers have a finite subcover, this allows one to ensure that the smallest subcover of any cover is finite. I.e. given an arbitrary cover $U$, one can consider the smallest in terms of cardinality subcover $S$ and associate to $U$ the finite quantity $\log_2(\text{Card}(S))$. This quantity, obviously, need not be finite in the general case of an arbitrary cover on a non-compact set.

Note that to use the crew operation, we need to split the space between public and private registers; we do not provide the full definition of this AMC since it is not necessary for this definition and the full definition is provided in the previous section.
However, a generalisation of entropy to non-compact sets can easily be defined by restricting the usual definition to finite covers\textsuperscript{6}. This is the definition we will use here.

**Definition 17.** Let $X$ be a topological space, and $\mathcal{U} = (U_i)_{i \in I}$ be a finite cover of $X$. We define the quantity $H^U_X(f)$ as

$$\min \{\log_2|\text{Card}(J)| \mid J \subset I, \cup_{j \in J} U_j = X\}.$$  

In other words, if $k$ is the cardinality of the smallest subcover of $\mathcal{U}$, $H^U_X(\mathcal{O}) = \log_2(k)$.  

**Definition 18.** Let $X$ be a topological space and $f : X \to X$ be a continuous map. For any finite open cover $\mathcal{U}$ of $X$, we define:

$$H^U_X(f, \mathcal{U}) = \frac{1}{k} H^U_X(\mathcal{U} \vee f^{-1}(\mathcal{U}) \vee \cdots \vee f^{-(k-1)}(\mathcal{U})).$$

One can show that the limit $\lim_{n \to \infty} H^U_X(f, \mathcal{U})$ exists and is finite; it will be noted $h(f, \mathcal{U})$. The topological entropy of $f$ is then defined as the supremum of these values, when $\mathcal{U}$ ranges over the set of all finite covers $\text{FCov}(X)$.  

**Definition 19.** Let $X$ be a topological space and $f : X \to X$ be a continuous map. The **topological entropy of $f$** is defined as $h(f) = \sup_{\mathcal{U} \in \text{FCov}(X)} h(f, \mathcal{U})$.

### 5.2 Graphings and Entropy

We now need to define the entropy of deterministic graphing. As mentioned briefly already, deterministic graphings on a space $X$ are in one-to-one correspondence with partial dynamical systems on $X$. To convince oneself of this, it suffices to notice that any partial dynamical system can be represented as a graphing with a single edge, and that if the graphing $G$ is deterministic its edges can be glued together to define a partial continuous function $G$. Thus, we only need to extend the notion of entropy to partial maps, and we can then define the entropy of a graphing $G$ as the entropy of its corresponding map $G$.

Given a finite cover $\mathcal{U}$, the only issue with partial continuous maps is that $f^{-1}(\mathcal{U})$ is not in general a cover. Indeed, $\{f^{-1}(U) \mid U \in \mathcal{U}\}$ is a family of open sets by continuity of $f$ but the union $\cup_{U \in \mathcal{U}} f^{-1}(U)$ is a strict subspace of $X$ (namely, the domain of $f$). It turns out the solution to this problem is quite simple: we notice that $f^{-1}(\mathcal{U})$ is a cover of $f^{-1}(X)$ and now work with covers of subspaces of $X$. Indeed, $\mathcal{U} \vee f^{-1}(\mathcal{U})$ is itself a cover of $f^{-1}(X)$ and therefore the quantity $H^U_X(f, \mathcal{U})$ can be defined as $(1/2) H^U_X(f^{-1}(X) \vee f^{-1}(\mathcal{U})).$  

We now generalise this definition to arbitrary iterations of $f$ by extending Definitions 18 and 19 to partial maps as follows.

**Definition 20.** Let $X$ be a topological space and $f : X \to X$ be a continuous partial map. For any finite open cover $\mathcal{U}$ of $X$, we define:

$$H^U_X(f, \mathcal{U}) = \frac{1}{k} H^U_X(\mathcal{U} \vee f^{-1}(\mathcal{U}) \vee \cdots \vee f^{-(k-1)}(\mathcal{U})).$$

The **entropy of $f$** is then defined as $h(f) = \sup_{\mathcal{U} \in \text{FCov}(X)} h(f, \mathcal{U})$, where $h(f, \mathcal{U})$ is again defined as the limit $\lim_{n \to \infty} H^U_X(f, \mathcal{U})$.

Now, let us consider the special case of a graphing $G$ with set of control states $S^G$. For an intuitive understanding, one can think of $G$ as the representation of a PRAM machine. We focus on the specific open cover indexed by the set of control states, i.e. $S = (XX\{s\}_{s \in S^G})$, and call it the states cover. We will now show how the partial entropy $H^U_X(G, \mathcal{S})$ is related to the set of admissible sequence of states. Let us define those first.

**Definition 21.** Let $G$ be a graphing, with set of control states $S^G$. An admissible sequence of states is a sequence $s = s_1s_2 \ldots s_n$ of elements of $S^G$ such that for all $i \in \{1, 2, \ldots, n-1\}$ there exists a subset $C$ of $X$ – i.e. a set of configurations – such that $G$ contains an edge from $C \times \{s_i\}$ to a subspace of $X \times \{s_{i+1}\}$.

**Example 22.** As an example, let us consider the very simple graphing with four control states a, b, c, d and edges from $X \times \{a\}$ to $X \times \{b\}$, from $X \times \{b\}$ to $X \times \{c\}$, from $X \times \{c\}$ to $X \times \{b\}$ and from $X \times \{c\}$ to $X \times \{d\}$. Then the sequences abcd and abcde are admissible, but the sequences aba, abcd, and abcba are not.

**Lemma 23.** Let $G$ be a graphing, and $S$ its states cover. Then for all integer $k$, the set $\text{Adm}_k(G)$ of admissible sequences of states of length $k > 1$ is of cardinality $2^k H^U_X(G, S)$.  

**Lemma 24.** Let $G$ be a graphing, and let $c : k \mapsto \text{Card}(\text{Adm}_k(G))$. Then $c(k) = (2^k 2^{h(G)})$ as $k$ goes to infinity.

The proofs of these two lemmas are in Appendix ??.

### 5.3 Cells Decomposition

Now, let us consider a deterministic graphing $G$, with its state cover $S$. We fix a length $k > 2$ and reconsider the sets $C[s] = C([s_1s_2 \ldots s_{k-1} s_k])$ (for a sequence of states $s = s_1s_2 \ldots s_k$) that appear in the proof of Lemma 23. The set $\{C[s]\}_{s \in \text{Adm}_k(G)}$ is a partition of the space $[G]^{-k+1}(X)$.

This decomposition splits the set of initial configurations into cells satisfying the following property: for any two initial configurations contained in the same cell $C[s]$, the $k$-th first iterations of $G$ goes through the same admissible sequence of states $s$.

**Definition 25.** Let $G$ be a deterministic graphing, with its state cover $S$. Given an integer $k$, we define the $k$-fold decomposition of $X$ along $G$ as the partition $\{C[s] \mid s \in \text{Adm}_k(G)\}$.

Then Lemma 23 provides a bound on the cardinality of the $k$-th cell decomposition. Using the results in the previous section, we can then obtain the following proposition.

**Proposition 26.** Let $G$ be a deterministic graphing, with entropy $h(G)$. The cardinality of the $k$-th cell decomposition of $X$ w.r.t. $G$, as a function $c(k)$ of $k$, is asymptotically bounded by $g(k) = 2^k 2^{h(G)}$, i.e. $c(k) = O(g(k))$.

The bound on the number of cells is important, but being able to say how the cells are defined is of greater value. We will now suppose that the machines we are working with satisfy an additional condition, namely that given a state if several transitions are possible depending on the current configuration, the different sets of configurations leading to different transitions can be delimited along some geometric object. For the sake of discussion, we will here consider the delimitations to be algebraic varieties since this is the case in Mullmuley’s proof; it should be clear however that other types of objects may be considered here. Let us notice that in the PRAM model, since such branchings occur only with conditionals the delimiting varieties are hyperplanes defined by $X_i = 0$.

With this assumption, one can try to look at the inverse image of these delimitations, say after $k$ steps of computation, through...
the iterated map \([G]^k\). We denote by \(\mathcal{V}_k\) the delimitations after \(k\) steps of computation following the sequence \(s\) of states, and by \(p\mathcal{V}_k\) the preimage of \(\mathcal{V}_k\) through \([G]^k\). Then the collection of all \(p\mathcal{V}_k\) where \(s\) ranges over the set of admissible sequences of length less than \(k\) defines the delimitations of the cells in the complete \(k\)-th cell decomposition of \(X\). Since the number of states is finite, the number of possible delimitations at each step is finite (and equal to the number of states minus 1).

In the general case, nothing insures us that the inverse image of a variety is a variety; let us say that \(G\) is regular. If we moreover assume that the graphing \(G\) is nice enough so that preimages of algebraic surfaces are algebraic surfaces, we can obtain the following proposition.

**Proposition 27.** Let \(G\) be a regular deterministic graphing with set of states \(S\). The \(k\)-th cell decomposition of \(X\) w.r.t. \(G\) is determined by at most \(\text{Card}(S)^k - 1\) algebraic varieties.

**Proof.** At each step, one has at most \(\text{Card}(S)\) different transitions delimited by \(\text{Card}(S) - 1\) varieties. This means that after \(k\) steps, the number of varieties is bounded by
\[
(\text{Card}(S) - 1)(1 + \text{Card}(S) + \text{Card}(S)^2 + \cdots + \text{Card}(S)^{k-1}) = \text{Card}(S)^k - 1.
\]
\[\square\]

In general, this bound is not very meaningful, as the set of states may be arbitrarily large. However, in the case of graphings interpreting PRAMS, it can be refined. Indeed, the value \(\text{Card}(S)\) appearing in the sums can be lowered in the case of PRAMS. Indeed, PRAMS are only allowed binary branchings (only conditionals may create branchings); thus, the number of branchings is bounded by \(2p\) where \(p\) denotes the number of processors of the PRAM. Moreover, as explained above, each conditional gives rise to a single delimitation variety – a hyperplan –, providing the much smaller bound of \(p\) delimiting varieties at each step. The sum can then be rewritten as
\[
p(1 + 2p + (2p)^2 + \cdots + (2p)^{k-1}) = p\left(\frac{(2p)^k - 1}{2p - 1}\right) \leq 2^kp^k.
\]

**Lemma 28.** Let \(G\) be a regular deterministic graphing interpreting a PRAM with \(p\) processors. The \(k\)-th cell decomposition of \(X\) w.r.t. \(G\) is determined by at most \(2^kp^k\) algebraic varieties.

### 6 Lower Bounds Abstractly

While the previous section dealt with arbitrary graphings defined from arbitrary AMCS, we now restrict our attention to the AMCS of PRAMS. As a consequence, the partitions into cells obtained in the previous section can be pulled back to partitions of the space \(Z^d\).

#### 6.1 Geometric Interpretation of Optimization Problems

We will start by showing how decision problems of a particular form induce a binary partition of the same space: the points that are accepted and those that are rejected. Intuitively, the machine decides the problem if the partition it induces refines the one of the problem.

We will consider problems of a very specific form: decision problems in \(Z^3\) associated to optimization problems. Let \(\mathcal{P}_{\text{opt}}\) be an optimization problem on \(R^d\). Solving \(\mathcal{P}_{\text{opt}}\) on an instance \(t\) amounts to optimizing a function \(f(t, \cdot)\) over a space of parameters. We note \(\text{Max}\mathcal{P}_{\text{opt}}(t)\) this optimal value. An affine function \(\text{Param} : [p, q] \rightarrow R^d\) is called a parametrization of \(\mathcal{P}_{\text{opt}}\). Such a parametrization defines naturally a decision problem \(\mathcal{P}_{\text{dec}} : \mathcal{P}_{\text{opt}}\) for all \((x, y, z) \in Z^3, (x, y, z) \in \mathcal{P}_{\text{dec}}\) iff \(z > 0, x/z \in [p, q]\) and \(y/z \leq \text{Param}\); \(\text{Max}\mathcal{P}_{\text{opt}}(x/z)\).

In order to study the geometry of \(\mathcal{P}_{\text{dec}}\) in a way that makes its connection with \(\mathcal{P}_{\text{opt}}\) clear, we consider the ambient space to be \(R^3\), and we define the ray \([p]\) of a point \(p\) as the half-line starting at the origin and containing \(p\). The projection \(\Pi(p)\) of a point \(p\) on a plane is the intersection of \([p]\) and the plane affine \(A_1\) of equation \(z = 1\). For any point \(p \in A_1\), and all \(p_1 \in [p]\), \(\Pi(p_1) = p\). It is clear that for \((p, q, q) \in Z^3 \times N^\ast\), \(\Pi((p, q, q)) = (p, q, q'/q, 1)\).

The cone \([C]\) of a curve \(C\) is the set of rays of points of the curve. The projection \(\Pi(C)\) of a surface or a curve \(C\) is the set of projections of points in \(C\). We note \(\text{Front}\) the frontier set
\[
\text{Front} = \{(x, y, 1) \in R^3 | y = \text{Param}; \text{Max}\mathcal{P}_{\text{opt}}(x/z)\}.
\]

and remark that \([\text{Front}]\) = \((x, y, z) \in R^2 \times R^+ | y/z = \text{Param}; \text{Max}\mathcal{P}_{\text{opt}}(x/z)\).

Finally, a machine \(M\) decides the problem \(\mathcal{P}_{\text{dec}}\) if the sub-partition of accepting cells in \(Z^3\) induced by the machine is finer than the one defined by the problem’s frontier \([\text{Front}]\) (which is defined by the equation \(y/z \leq \text{Param}; \text{Max}\mathcal{P}_{\text{opt}}(x/z)\)).

#### 6.2 Parametric Complexity

We now further restrict the class of problems we are interested in: we will only consider \(\mathcal{P}_{\text{opt}}\) such that \(\text{Front}\) is simple. Precisely:

**Definition 29.** We say that \(\text{Param}\) is an affine parametrization of \(\mathcal{P}_{\text{opt}}\) if \(\text{Param}\); \(\text{Max}\mathcal{P}_{\text{opt}}\) is

- convex
- piecewise linear, with breakpoints \(\lambda_1 < \cdots < \lambda_p\)
- such that the \((\lambda_1)\) and the \((\text{Param}; \text{Max}\mathcal{P}_{\text{opt}}(\lambda_i))\) are all rational.

The complexity \(\rho(\text{Param})\) is defined as the number of breakpoints of \(\text{Param}\); \(\text{Max}\mathcal{P}_{\text{opt}}\).

An optimization problem that admits an affine parametrization of complexity \(\rho\) is thus represented by a surface \([\text{Front}]\) that is quite simple: the cone of the graph of a piecewise affine function, constituted of \(\rho\) segments. We say that such a surface is a \(\rho\)-fan. This restriction seems quite serious when viewed geometrically. Nonetheless, many optimization problems admit such a parametrization. Before giving examples, we introduce another measure of the complexity of a parametrization.

**Definition 30.** Let \(\mathcal{P}_{\text{opt}}\) be an optimization problem and \(\text{Param}\) be an affine parametrization of it. The \(\text{bitsize}\) of the parametrization is the maximum of the bitsizes of the numerators and denominators of the coordinates of the breakpoints of \(\text{Param}\); \(\text{Max}\mathcal{P}_{\text{opt}}\).

In the same way, we say that a \(\rho\)-fan is of bitsize \(\beta\) if all its breakpoints are rational and the bitsize of their coordinates is lesser than \(\beta\).

**Theorem 31 (Murty [33], Carstensen [11]).**

1. there exists an affine parametrization of bitsize \(O(n)\) and complexity \(2^{O(n)}\) of combinatorial linear programming, where \(n\) is the total number of variables and constraints of the problem.

2. there exists an affine parametrization of bitsize \(O(n^3)\) and complexity \(2^{O(n)}\) of the maxflow problem for directed and undirected networks, where \(n\) is the number of nodes in a network.
whose degree is moreover bounded. This choice of representation

As hinted in the last section, we will consider algebraic surfaces. A feasible set of algebraic surfaces is the collection of all algebraic surfaces, of total degree $S$.

Let $K$ be a compact of $R^3$ and $P = (P_1, \ldots, P_m)$ be a partition of $Z^3 \cap K$. $P$ can be extended to a partition of the whole of $K$ in a number of ways, as pictured in Fig. 1. In particular, $P$ can always be extended to a partition $P_{\text{alg}}$ (resp. $P_{\text{smooth}}, P_{\text{ana}}$) of $K$ such that all the cells are compact, and the boundaries of the cells are all algebraic (resp. smooth, analytic) surfaces.

In general, such surfaces have no reason to be easy to compute and the more they are endowed with structure, the more complicated to compute they are to be. In the specific case of PRAMS, the decomposition can naturally be represented with algebraic surfaces whose degree is moreover bounded. This choice of representation might not hold for any other model of computation, for which it might be more interesting to consider surfaces of a different kind.

Mulmuley’s result is based on the following strategy. Firstly, one proves that the algebraic extension of any partition that refines a partition decided by a problem contains surfaces whose complexity (measured by its variations) is linear in the complexity of the problem. Secondly, one proves that the complexity of the surfaces of a partition induced by a PRAM is bounded by a polynomial of the number of processors.

The proof uses algebraic geometry machinery, but it turns out this approach is not rooted into it: a different model of computation might suggest a different choice of separating surfaces (such as smooth surfaces, or Lie groups) which then would be studied by tools coming from an other branch of mathematics.

7 Algebraic Surfaces

As hinted in the last section, we will consider algebraic surfaces.

An algebraic surface in $R^3$ is a surface defined by an equation of the form $p(x, y, z) = 0$ where $p$ is a polynomial. If $S$ is a set of surfaces, each defined by a polynomial, the total degree of $S$ is defined as the sum of the degrees of polynomials defining the surfaces in $S$.

Let $K$ be a compact of $R^3$ delimited by algebraic surfaces and $S$ be a finite set of algebraic surfaces, of total degree $\delta$. We can assume that $K$ is actually delimited by two affine planes of equation $\mu x + y = 0$, by taking any such compact containing $K$ and adding the surfaces bounding $K$ to $S$. $S$ defines a partition of $K$ by considering maximal compact subspaces of $K$ whose boundaries are included in surfaces of $S$. Such elements are called the cells of the decomposition associated to $S$.

The cell of this partition can have complicated shapes: in particular, a cell can have a arbitrarily high number of surfaces of $S$ as boundaries. We are going to refine this partition into a partition $Col_S$ whose cells are all bounded by cones of curves and at most two surfaces in $S$.

7.1 Collins’ decomposition

We define the silhouette of a surface defined by the equation $p(x, y, z) = 0$ by:

$$\begin{align*}
    p(x, y, z) &= 0 \\
    x \frac{\partial p}{\partial x} + y \frac{\partial p}{\partial y} + z \frac{\partial p}{\partial z} &= 0.
\end{align*}$$

The silhouette of a surface is the curve on the surface such that all points $(x, y, z)$ of the silhouette are such that the ray $[(x, y, z)]$ is orthogonal to the tangent plane of the surface on $(x, y, z)$.

Up to infinitesimal perturbation of the coefficients of the polynomials, we may assume that the surfaces of $S$ have no integer points in $K$.

$\Pi(K) = \{\Pi(x) \mid x \in K\}$ is a compact of the affine plane $A_1$. Let us consider the set $\Pi(S)$ of curves in $\Pi(K)$ containing:

- the projection of the silhouettes of surfaces in $S$;
- the projection of the intersections of surfaces in $S$ and of the intersection of surfaces of $S$ with the planes $z = \mu(1 + \frac{n \beta}{\delta})$, $n \in \{1, \ldots, 6\delta - 1\}$, where $\delta$ is the total degree of $S$;
- vertical lines of the form $\{(x, a) \mid |x| \leq 2^{\beta+1}\}$ for $a$ a constant such that such lines pass through:
  - all intersections among the curves;
  - all singular points of the curves;
  - all critical points of the curves with a tangent supported by $\partial x$.

$\Pi(S)$ defines a Collins decomposition $[13]$ of $\Pi(K)$. The intersection of any affine line supported by $\partial x$ of the plane with a region of this decomposition is connected if nonempty.

Let $c$ be a cell in $\Pi(S)$. It is enclosed by two curves in $\Pi(K)$ and at most two vertical lines. The curves can be parametrized by $c_{\text{max}} : x \mapsto \max\{y \in R \mid (x, y, 1) \in c\}$ and $c_{\text{min}} : x \mapsto \min\{y \in R \mid (x, y, 1) \in c\}$, which are both smooth functions. The volatility of $c$ is defined as the number of extrema of the second derivatives $c_{\text{max}}''$ and $c_{\text{min}}''$ on their domains of definition.

This set of curves can be lifted to a set of surfaces $Col_S(K)$ of $K$ that contains:

- the surfaces of $S$;
- the cones $[s]$ of every curve $s$ in $\Pi(S)$;
- the planes bounding $K$;
- $6\delta - 2$ dividing planes of equation $z = \mu(1 + \frac{n \beta}{\delta})$, $n \in \{1, \ldots, 6\delta - 1\}$.

The projection of a cell of $Col_S$ is a cell of $\Pi(S)$. We say that a cell of $Col_S(K)$ is flat if none of its boundaries are included in surfaces of $S$.

Let us call $d(S)$ the number of cells in $Col_S(K)$.

Let $c$ be a cell in $Col_S(K)$. Its volatility is defined as the volatility of its projection in $\Pi(S)$.
7.2 Volatility and Separation

Definition 32. Let \( K \) be a compact of \( \mathbb{R}^3 \). A finite set of surfaces \( S \) on \( K \) separates a \( \rho \)-fan \( F \) on \( K \) if the partition on \( \mathbb{R}^3 \cap K \) induced by \( S \) is finer than the one induced by \( F \).

Theorem 33. Let \( S \) be a finite set of algebraic surfaces of total degree \( \delta \), and \( F \) a \( \rho \)-fan of bitsize \( \beta \).

If \( S \) separates \( F \), there exists a compact \( K \) and a cell of \( \text{Col}_S(K) \) with volatility greater than \( \rho/d(S) \).

In order to prove this theorem, we will build explicitly the compact \( K \) and this cell by considering sample points on \( F \) and show in Lemma 35 a bound on the volatility of this cell.

Let \( K \) be a compact delimited by the cone of a rectangle \( \{(x, y, 1) \mid |x|, |y| \leq 2^{\beta+1}\} \) and two planes of equation \( z = \mu \) and \( z = 2\mu \), with \( \mu > (6\delta + 1)2^{\beta} \). We first remark that all affine segments of \( F \) are in the rectangle base of \( K \).

For each affine segment of \( F \) with endpoints \((x_i, y_i, 1)\) and \((x_{i+1}, y_{i+1}, 1)\), let, for \( 0 < k < 10d(S) \), \( g^k \) be such that \((x_i^k, y_i^k, 1)\) is in the affine segment, where \( x_i^k = \frac{10d(S)-k}{10d(S)} x_i + \frac{k}{10d(S)} x_{i+1} \). We remark that, as \(|x_i-x_{i+1}| > 2^{-\beta} \), we have, for \( k, k' \), \( |x_i^k-x_{i}^{k'}| > 2^{-\beta}/10d(S) \).

Lemma 34. For all sample points \((x_i^k, y_i^k, 1)\), there exists a flat cell in \( \text{Col}_S \) that contains an integer point of \( \{(x_i^k, y_i^k, 1)\} \).

Proof. Let \((x_i^k, y_i^k, 1)\) be a sample point. \( \{(x_i^k, y_i^k, 1)\} \) is divided in \( N+1 \) intervals by the dividing planes. On the other hand, \( \{(x_i^k, y_i^k, 1)\} \) intersects surfaces of \( S \) in at most \( \delta \) points, by Bézout theorem.

So, there exists an interval \( e \) of \( \{(x_i^k, y_i^k, 1)\} \) that is bounded by the dividing planes and that do not intersect any surface in \( S \). By construction, \( e \) is included in a flat cell, and its projection on the \( z \)-axis has length \( \mu/(6\delta + 1) \), so, as \( (x_i^k, y_i^k, 1) \) is of bitsize \( \beta \), \((n2^\beta x_i^k, n2^\beta y_i^k, n2^\beta)\) is, for all \( n \in \mathbb{N} \) an integer point of the ray, so, as \( \mu > (6\delta + 1)2^\beta \), \( e \) contains an integer point.

So, for each affine segment of \( F \), there exists a flat cell in \( \text{Col}_S \) that contains integer points in the ray of at least 10 sample points of the affine segment. Going further, there exists a cell \( c \) of \( \text{Col}_S \) that contains integer points in the ray of at least 10 sample points of \( \rho/d(S) \) affine segments of \( F \).

Lemma 35. The volatility of \( c \) is at least \( \rho/d(S) \).

This is achieved by applying the mean value theorem on the function \( \Pi(c)_{\max} \) on pairs of sample points. In particular, this proof uses no algebraic geometry. The details are in Appendix ??.

7.3 Volatility and Degree

Mulmuley’s result follows from Thm. 33 and the two lemmas:

Lemma 36. Let \( S \) be a finite set of curves of total degree \( \delta \), and \( K \) be a compact. The cells of the decomposition \( \text{Col}_S \) of \( K \) have a volatility bounded by a polynomial in \( \delta \).

Proof. Let \( c \) be a cell in \( \text{Col}_S \) and \( g(x, y) = 0 \) be the equation of one of the boundaries of \( \Pi(c) \) in the affine plane. The degree of \( g \) is bounded by the degree of the intersection of surfaces in \( S \). Any extrema \( x \) of \( f'' \), where \( f \) is a parametrization \( y = f(x) \) of this boundary, can be represented as a point \((x, y(y(1), y(2), y(3)))\) in the 5-dimensional phase space that satisfy polynomial equations of the form:

\[
g(x, y) = 0, \quad g_1(x, y(y(1), y(2))) = 0, \quad g_2(x, y(y(1), y(2), y(3))) = 0, \quad y(3) = 0,
\]

where all the polynomials’ degrees are all bounded by the degree of the intersection of surfaces in \( S \) (as they are the derivatives of \( g \)). So, by the Milnor–Thom theorem, such points are in number polynomial in the total degree of the surfaces of \( S \).

Lemma 37. The number of cells \( d(S) \) of the Collins decomposition of \( S \) is polynomial in \( \delta \).

Proof. The intersection of the surfaces in \( S \) are algebraic varieties of number bounded by \( \delta \), by the Milnor–Thom theorem. Moreover, so are the silhouettes of the surfaces, as they are the intersection of two algebraic varieties of total degree smaller than \( \delta \). So, the number of cells in \( \text{Col}_S \) is bounded by the number of cells of \( S \) times the number of dividing planes times the number of intersections, silhouettes and vertical lines they engender.

Theorem 38 (Mulmuley). Let \( S \) be a finite set of algebraic surfaces of total degree \( \delta \).

There exists a polynomial \( P \) such that, for all \( \rho > P(\delta), S \) does not separate \( \rho \)-fans.

8 Mulmuley’s result

We remark that a graphing \( G \) corresponding to a PRAM is more than regular. Indeed, not only are the preimages of algebraic surfaces through \( G \) algebraic surfaces, but we know that all the branching occur with conditionals the delimitating varieties are delimited by hyperplans defined by \( X_i = 0 \). and that \( X_i \) can be expressed as a polynomial in the inputs, with degree bounded in function of the number of steps. So we have the following stronger version of Prop. 27:

Lemma 39. Let \( G \) be a deterministic graphing interpreting a PRAM with \( p \) processors. The \( k \)-th cell decomposition of \( X \) w.r.t. \( G \) is determined by at most \( 2^k \) algebraic varieties of degree at most \( 2^k \).

Theorem 40. Let \( G \) be a deterministic graphing interpreting a PRAM with \( p \) processors.

There exists a polynomial \( P \) such that, for all \( \rho > P(2^k+1)^p \), \( G \) does not separate \( \rho \)-fans in \( k \) steps.
8.1 PRAMs without bit operations

The definition of PRAM we considered is quite ideal in that the complexities are stated without references to the size of the inputs. We can consider the length of an input to be the minimal length of a binary word representing it. Mulmuley defines this model as the pram model without bit operations, by taking the point of view that the integers stored in the memory really are binary words, whose individual bits can not be accessed easily. "

Corollary 41. Let $G$ be a deterministic graphing interpreting a PRAM without bit operations with $3^{\Omega(N^4)}$ processors, where $N$ is the length of the inputs and $c$ any positive integer. $G$ does not decide maxflow in $O(N^c)$ steps.

Proof. Let $n = N^{2c}$ and consider the problem 2 of Thm. 31 and its family of affine parametrizations of bitsize $O(n^c) = O(N^{4c})$ and complexity $p(n) = 2^{\Omega(n)} = 2^{\Omega(N^{4c})}$. Let $P$ be the polynomial of Thm. 40. $P(2^{\Omega(n)})(2^{\Omega(N^4)}(O(n^c))) = 2^{\Omega(N^c)}$, so, for large enough values of $N, \rho(n)$ is larger than the polynomial. By Thm. 40, it implies that $G$ does not decide maxflow.

9 Conclusion

This reformulation of Mulmuley’s proof shows that the algebraic geometric tools are only used in order to get some bounds on the number of intersections of the surfaces in Subsec. 7.2 and 7.3. The essence of the proof lie elsewhere, in the study of the geometry of a dynamical system. The use of algebraic geometric methods is dictated by the presence of polynomials, which is a consequence of the fact that the PRAMs without bit operations essentially manipulate numbers and not strings of bits.

Our approach is robust in that the decomposition a machine induces on its space of state can be studied, and bounds on its cardinality computed, even if the cells of the decomposition can not naturally be represented as algebraic surfaces.

References