Descriptive Complexity: Part 2

Anuj Dawar

University of Cambridge Computer Laboratory

Caleidoscope, Paris, 19 June 2019

Outline

Lecture 1: Introduction to Descriptive Complexity.

- Proving inexpressibility in Logics.
- Characterizing complexity classes.
- FPC and the Cai-Fürer-Immerman construction.

Lecture 2: FPC and its connections with:

- circuit complexity
- extension polytopes
- hardness of approximation

Review

Fagin: ESO = NP

Immerman-Vardi: FP = P on *ordered* structures.

We are building up tools for proving inexpressibility in ever more powerful logics.

We used *Ehrenfeucht games* to show that first-order logic cannot define *Evenness*, *Connectivity*, *2-Colourability*.

We used *pebble games* to show that FP cannot define *Evenness*, *Perfect Matchings*, *Hamiltonicity*.

We want to use games to show that FPC cannot define $\mathrm{Solv}(\mathbb{Z}_2)$.

Constructing systems of equations

Take G a 4-regular, connected graph.
Define equations $\mathbf{E}_G$ with two variables x_0^e, x_1^e for each edge e.
For each vertex v with edges e_1, e_2, e_3, e_4 incident on it, we have 16 equations:

$$E_v : \qquad x_a^{e_1} + x_b^{e_2} + x_c^{e_3} + x_d^{e_4} \equiv a + b + c + d \pmod 2$$

$\tilde{\mathbf{E}}_G$ is obtained from $\mathbf{E}_G$ by replacing, for exactly one vertex v, E_v by:

$$E'_v : \qquad x_a^{e_1} + x_b^{e_2} + x_c^{e_3} + x_d^{e_4} \equiv a + b + c + d + 1 \pmod 2$$

We can show: $\mathbf{E}_G$ is satisfiable; $\tilde{\mathbf{E}}_G$ is unsatisfiable.

Satisfiability

Lemma $\mathbf{E}_G$ is satisfiable.

by setting the variables x_i^e *to* i*.*

Lemma $\tilde{\mathbf{E}}_G$ is unsatisfiable.

Consider the subsystem consisting of equations involving only the variables x_0^e*.*
The sum of all left-hand sides is

$$2\sum_e x_0^e \equiv 0 \pmod 2$$

However, the sum of right-hand sides is 1*.*

Now we show that, for each k, we can find a graph G such that $\mathbf{E}_G \equiv^{C^k} \tilde{\mathbf{E}}_G$.

Counting Game

Immerman and Lander (1990) defined a *pebble game* for C^k.
This is again played by *Spoiler* and *Duplicator* using k pairs of pebbles $\{(a_1, b_1), \ldots, (a_k, b_k)\}$ on a pair of structures $\mathbb{A}$ and $\mathbb{B}$

At each move, Spoiler picks i and a set of elements of one structure (say $X \subseteq B$)

Duplicator responds with a set of vertices of the other structure (say $Y \subseteq A$) of the same size.

Spoiler then places a_i on an element of Y and Duplicator must place b_i on an element of X.

Spoiler wins at any stage if the partial map from $\mathbb{A}$ to $\mathbb{B}$ defined by the pebble pairs is not a partial isomorphism

If Duplicator has a winning strategy for p moves, then $\mathbb{A}$ and $\mathbb{B}$ agree on all sentences of C^k of quantifier rank at most p.

Bijection Games

$\equiv^{C^k}$ is also characterised by a k-pebble *bijection game*. **(Hella 96)**.
The game is played on graphs $\mathbb{A}$ and $\mathbb{B}$ with pebbles $a_1, \ldots, a_k$ on $\mathbb{A}$ and $b_1, \ldots, b_k$ on $\mathbb{B}$.

- *Spoiler* chooses a pair of pebbles a_i and b_i;
- *Duplicator* chooses a bijection $h : A \to B$ such that for pebbles a_j and $b_j (j \neq i)$, $h(a_j) = b_j$;
- *Spoiler* chooses $a \in A$ and places a_i on a and b_i on $h(a)$.

Duplicator loses if the partial map $a_i \mapsto b_i$ is not a partial isomorphism.
Duplicator has a strategy to play forever if, and only if, $\mathbb{A} \equiv^{C^k} \mathbb{B}$.

Equivalence of Games

It is easy to see that a winning strategy for *Duplicator* in the bijection game yields a winning strategy in the counting game:

> *Respond to a set* $X \subseteq A$ *(or* $Y \subseteq B$*) with* $h(X)$ *(*$h^{-1}(Y)$*, respectively).*

For the other direction, consider the partition induced by the equivalence relation

$$\{(a, a') \mid (\mathbb{A}, \mathbf{a}[a/a_i]) \equiv^{C^k} (\mathbb{A}, \mathbf{a}[a'/a_i])\}$$

and for each of the parts X, take the response Y of *Duplicator* to a move where *Spoiler* would choose X.
Stitch these together to give the bijection h.

Cops and Robbers

A game played on an undirected graph $G = (V, E)$ between a player controlling k cops and another player in charge of a robber.

At any point, the cops are sitting on a set $X \subseteq V$ of the nodes and the robber on a node $r \in V$.
A move consists in the cop player removing some cops from $X' \subseteq X$ nodes and announcing a new position Y for them. The robber responds by moving along a path from r to some node s such that the path does not go through $X \setminus X'$.
The new position is $(X \setminus X') \cup Y$ and s. If a cop and the robber are on the same node, the robber is caught and the game ends.

Cops and Robbers on the Grid

If G is the $k \times k$ toroidal grid, than the *robber* has a winning strategy in the k-*cops and robbers* game played on G.

To show this, we note that for any set X of at most k vertices, the graph $G \setminus X$ contains a connected component with at least half the vertices of G.

If all vertices in X are in distinct rows then $G \setminus X$ is connected. Otherwise, $G \setminus X$ contains an entire row and in its connected component there are at least $k - 1$ vertices from at least $k/2$ columns.

Robber's strategy is to stay in the large component.

Cops, Robbers and Treewidth

Actually, the cops and robbers game *characterizes tree-width*.

> *A connected graph G has tree-width $\geq k$ if, and only if, robber has a winning strategy against a team of k cops on G.*

Cops, Robbers and Bijections

Suppose G is such that the *robber* has a winning strategy in the $2k$-*cops and robbers* game played on G.

We use this to construct a winning strategy for Duplicator in the k-pebble bijection game on $\mathbf{E}_G$ and $\tilde{\mathbf{E}}_G$.

- A bijection $h : \mathbf{E}_G \to \tilde{\mathbf{E}}_G$ is *good bar* v if it is an isomorphism everywhere except at the variables x_a^e for edges e incident on v.
- If h is good bar v and there is a path from v to u, then there is a bijection h' that is good bar u such that h and h' differ only at vertices corresponding to the path from v to u.
- Duplicator plays bijections that are good bar v, where v is the robber position in G when the cop position is given by the currently pebbled elements.

Restricted Graph Classes

If we restrict the class of structures we consider, FPC may be powerful enough to express all polynomial-time decidable properties.

1. FPC captures P on *trees*. **(Immerman and Lander 1990)**.
2. FPC captures P on any class of graphs of *bounded treewidth*. **(Grohe and Mariño 1999)**.
3. FPC captures P on the class of *planar graphs*. **(Grohe 1998)**.
4. FPC captures P on any *proper minor-closed class of graphs*. **(Grohe 2010)**.

In each case, the proof proceeds by showing that for any G in the class, a *canonical, ordered* representaton of G can be interpreted in G using FPC.

Beyond FPC

How do we define logics extending FPC while remaining inside P?

FPrk is fixed-point logic with an operator for *matrix rank* over finite fields.

(D., Grohe, Holm, Laubner, 2009)

Choiceless Polynomial Time with counting ($\tilde{\mathrm{C}}$PT(Card)) is a class of computational problems defined by **(Blass, Gurevich and Shelah 1999)**.
It is based on a *machine model (Gurevich Abstract State Machines)* that works directly on a graph or relational structure (rather than on a string representation).
$\tilde{\mathrm{C}}$PT(Card) is the polynomial time and space restriction of the machines.

Both of these have expressive power *strictly greater* than FPC.
Their relationship to each other and to P remains unknown.

We need new tools to analyze the *expressive power* of these logics.

Circuit Complexity

A *language* $L \subseteq \{0,1\}^*$ can be described by a family of *Boolean functions*:

$$(f_n)_{n \in \omega} : \{0,1\}^n \to \{0,1\}.$$

Each f_n may be computed by a *circuit* C_n made up of

- Gates labeled by Boolean operators: $\wedge, \vee, \neg$,
- Boolean inputs: $x_1, \ldots, x_n$, and
- A distinguished gate determining the output.

If there is a polynomial $p(n)$ bounding the *size* of C_n, i.e. the number of gates in C_n, the language L is in the class P/poly.

If, in addition, the function $n \mapsto C_n$ is computable in *polynomial time*, L is in P.

Note: For these classes it makes no difference whether the circuits only use $\{\wedge, \vee, \neg\}$ or a richer basis with *threshold* or *majority* gates.

Circuits for Graph Properties

We want to study families of circuits that decide properties of *graphs* (or other relational structures—for simplicity of presentation we restrict ourselves to graphs).

We have a family of Boolean circuits $(C_n)_{n\in\omega}$ where there are n^2 inputs labelled $(i, j) : i, j \in [n]$, corresponding to the *potential edges*.
Each input takes value 0 or 1;

Graph properties in P are given by such families where:

- the size of C_n is bounded by a polynomial $p(n)$; and
- the family is uniform, so the function $n \mapsto C_n$ is in P (or DLogTime).

Invariant Circuits

C_n is *invariant* if, for every input graph, the output is unchanged under a permutation of the inputs induced by a permutation of $[n]$.

That is, given any input $G : [n]^2 \to \{0, 1\}$, and a permutation $\pi \in S_n$,

C_n accepts G if, and only if, C_n accepts the input πG given

$$(\pi G)(i, j) = G(\pi(i), \pi(j)).$$

Note: this is not the same as requiring that the result is invariant under *all* permutations of the input. That would only allow us to define functions of the *number* of 1s in the input.
This requirement is simpy that the circuit recognises an *encoding-invariant* graph property.

Symmetric Circuits

Say C_n is *symmetric* if any permutation of $[n]$ applied to its inputs can be extended to an automorphism of C_n.

> *i.e., for each $\pi \in S_n$, there is an automorphism of C_n that takes input (i, j) to $(\pi i, \pi j)$.*

Any symmetric circuit is invariant, but *not* conversely.

> *Consider the natural circuit for deciding whether the number of edges in an n-vertex graph is even.*

Any invariant circuit can be converted to a symmetric circuit, but with potentially *exponential blow-up*.

Logic and Circuits

Any formula of φ *first-order logic* translates into a uniform family of circuits C_n

> *For each subformula $\psi(\overline{x})$ and each assignment $\overline{a}$ of values to the free variables, we have a gate.*
> *Existential quantifiers translate to big disjunctions, etc.*

The circuit C_n is:

- of *constant* depth (given by the depth of φ);
- of size at mose $c \cdot n^k$ where c is the number of subformulas of φ and k is the *maximum number of free variables* in any subformula of φ.
- *symmetric* by the action of $\pi \in S_n$ that takes $\psi[\overline{a}]$ to $\psi[\pi(\overline{a})]$.

FP and Circuits

For every sentence φ of FP there is a k such that for every n, there is a formula φ_n of L^k that is equivalent to φ on all graphs with at most n vertices.

The formula φ_n has

- *depth* n^c for some constant c;
- at most k free variables in each sub-formula for some constant k.

It follows that every graph property definable in FP is given by a family of *polynomial-size, symmetric* circuits.

FPC and Counting

For every sentence φ of FP there is a k such that for every n, there is a formula φ_n of C^k that is equivalent to φ on all graphs with at most n vertices.

The formula φ_n has

- *depth* n^c for some constant c;
- at most k free variables in each sub-formula for some constant k.

It follows that every graph property definable in FP is given by a family of *polynomial-size, symmetric* circuits in a basis with *threshold gates*.

Note: we could also alternatively take a basis with *majority* gates.

Main Result

The following is established in **(Anderson, D. 2017)**:

Theorem

A class of graphs is accepted by a P-uniform, polynomial-size, symmetric family of threshold circuits if, and only if, it is definable in FPC.

We could jut use *majority* instead of threshold gates. Or we could through in all *fully symmetric* Boolean functions.

We get a natural and purely circuit-based characterisation of FPC definability.

Inexpressibility results for FPC can be seen as lower bound results against a natural circuit class.

Counting Width

For any class of structures $\mathcal{C}$, we define its *counting width* $\nu_{\mathcal{C}} : \mathbb{N} \to \mathbb{N}$ so that

> $\nu_{\mathcal{C}}(n)$ *is the least* k *such that* $\mathcal{C}$ *restricted to structures with at most* n *elements is closed under* $\equiv^{C^k}$.

Every class in FPC has counting width bounded by a *constant*.

3-Sat, *Hamiltonicity*, *3-Colourability* all have counting width $\Omega(n)$.

FPC-Reductions

We can define a notion of one class $\mathcal{C}$ being FPC-*reducible* to another $\mathcal{D}$

If $\mathcal{C} \leq_{\mathsf{FPC}} \mathcal{D}$ then

$$\nu_{\mathcal{D}} = \Omega(\nu_{\mathcal{C}}^{1/d}).$$

If the reduction takes $\mathcal{C}$-instances to $\mathcal{D}$-instances of *linear size*, then

$$\nu_{\mathcal{D}} = \Omega(\nu_{\mathcal{C}}).$$

Known linear lower bounds follow from $\nu_{\mathsf{Solv}(\mathbb{Z}_2)} = \Omega(n)$.

Linear Programming

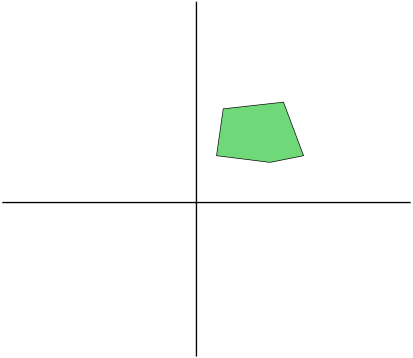
Linear Programming is an important algorithmic tool for solving a large variety of optimization problems.

It was shown by **(Khachiyan 1980)** that linear programming problems can be solved in polynomial time.
We have a set C of *constraints* over a set V of *variables*.
Each $c \in C$ consists of $a_c \in \mathbb{Q}^V$ and $b_c \in \mathbb{Q}$.

Feasibility Problem: Given a linear programming instance, determine if there is an $x \in \mathbb{Q}^V$ such that:

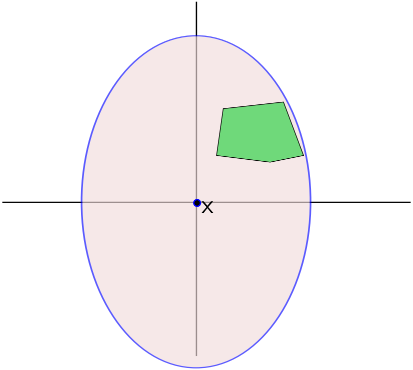
$$a_c^T x \leq b_c \quad \text{for all } c \in C$$

We can show that this, and the corresponding *optimization problem* are expressible in FPC.



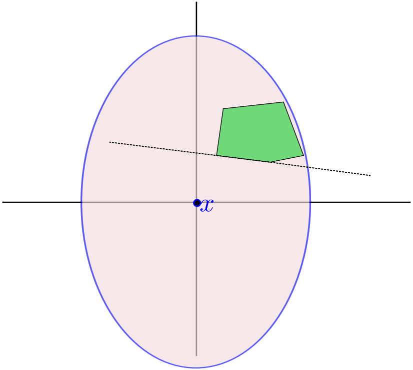
The set of constraints determines a *polytope*

Ellipsoid Method



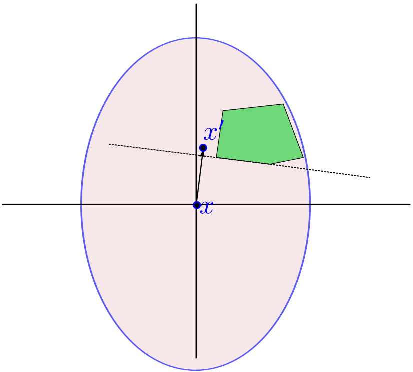
Start at the origin and calculate an *ellipsoid* enclosing it.

Ellipsoid Method



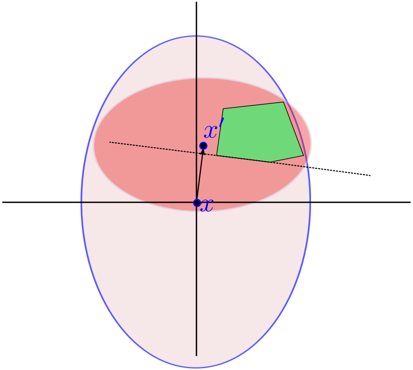
If the centre is not in the polytope, choose a constraint it *violates*.

Ellipsoid Method



Calculate a new *centre*.

Ellipsoid Method



And a new ellipsoid around the centre of at most *half* the volume.

Ellipsoid Method in FPC

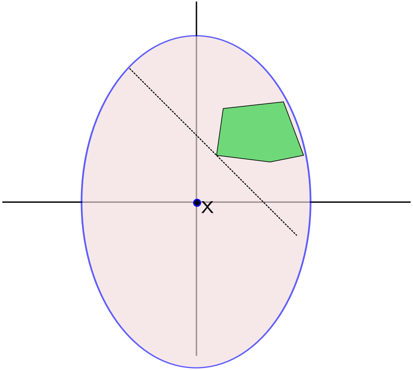
We can encode all the calculations involved in FPC.

This relies on expressing algebraic manipulations of *unordered* matrices.

What is not obvious is how to *choose* the violated constraint on which to project.

However, the ellipsoid method works as long as we can find, at each step, some *separating hyperplane*.

Ellipsoid Method in FPC



Ellipsoid Method in FPC

We can encode all the calculations involved in FPC.

This relies on expressing algebraic manilpulations of *unordered* matrices.

What is not obvious is how to *choose* the violated constraint on which to project.

However, the ellipsoid method works as long as we can find, at each step, some *separating hyperplane*.

So, we can take:

$$(\sum_{c \in S} a_c)^T x \leq \sum_{c \in S} b_c$$

where S is the *set* of all violated constraints.

Separation Oracle

More generally, the ellipsoid method can be used, even when the *constraint matrix* is not given explicitly, as long as we can always determine a *separating hyperplane*.

In particular, the polytope represented may have *exponentially many* facets.

We can show that as long as the *separation oracle* can be defined in FPC, the corresponding *optimization problem* can be solved in FPC.

Graph Matching

Recall, in a *graph* $G = (V, E)$ a matching $M \subset E$ is a set of edges such that each vertex is incident on *at most* one edge in M.

(Edmonds 1965) showed that the problem of finding a *maximum weight matching* in $G = (V, E), \quad w : \mathbb{Q}^E_{\geq 0}$ can be expressed as an *exponential size* linear program.

We can show that a *separation oracle* for this polytope is definable by an FPC formula interpreted in the weighted graph G.

As a consequence, there is an FPC formula defining the *size* of the maximum matching in G.

Note that this does not allow us to define an *actual* matching.

Lift and Project Hierarchies

Given a *polytope* $\mathcal{K}$ for *integer* optimization problem, we can get a better approximation of the *convex hull* of the integer points by means of *lift-and-project* programs.

The general idea is to add new variables $y_{x_1,\ldots,x_t}$ to denote the product $x_1 \cdots x_t$ and add linear (or semi-definite) constraints to try and force this meaning.

We get hierarchies as t increases:

- *Sherali-Adams*: $\mathsf{SA}_t(\mathcal{K})$
- *Lovasz-Schrijver*: $\mathsf{LS}_t(\mathcal{K})$
- *Lasserre*: $\mathsf{Las}_t(\mathcal{K})$

Of these, the last is the strongest.

For many cases, we can show that the number of levels$t$ required to get an exact soultion can be bounded by $\Omega(\nu_{\mathcal{C}})$.

Hardness of Approximation

MAX 3SAT:
We are given a *Boolean formula* φ in 3CNF, i.e. a conjunction of clauses with three literals per clause.

Say φ has n Boolean variables and m clauses.

Let m^* denote the *maximum* number such that some assignment of values to the Boolean variables makes m^* clauses of φ true.

Algorithmic Problems:

- *Find* an assignment of values to the variables that makes m^* clauses of φ true;
- *Determine* the value of m^*;
- c-*approximate* m^* for some constant $0 < c < 1$, i.e. give a value m' with a guarantee that $cm^* < m' \leq m^*$.

Lower Bounds

NP-completeness (Cook; Levin 1973):
Unless P = NP, there is no *polynomial-time algorithm* that can determine m^*.

PCP Theorem (Arora et al. 1998):
There is a constant $c < 1$ such that, unless P = NP, there is no *polynomial-time algorithm* that can c-approximate m^*.

(Håstad 2001):
Unless P = NP, for every $\epsilon > 0$ there is no *polynomial-time algorithm* that can $(\frac{7}{8} + \epsilon)$-approximate m^*.

> *Note: This is optimal since there is a trivial algorithm that can $\frac{7}{8}$-approximate m^*.*

MAX SNP

(Papadimitriou-Yannakakis 1991) define (in *syntactic terms*) a class MAX SNP of *NP optimization problems*.

For every problem in MAX SNP, there is a constant d such that there is a polynomial-time d-approximation algorithm.

They also define a notion of *approximation preserving reduction* under which MAX 3SAT is MAX SNP-*complete*.

It is a consequence of the PCP theorem that for every MAX SNP-*complete* problem, there is a constant $c < 1$ such that, unless P = NP, there is no polynomial-time c-approximation algorithm.

This poses a challenge for each problem in MAX SNP, to determine the best possible value of d.

MAX 3XOR

We are given a Boolean formula φ in 3XOR, i.e. a conjunction of clauses each of which is the *exclusive or* ($\oplus$) of three literals.

Say φ has n Boolean variables and m clauses.

Let m^* denote the *maximum* number such that some assignment of values to the Boolean variables makes m^* clauses of φ true.

- determining whether $m^* = m$ can be done in polynomial-time, by Gaussian elimination;
- determining the exact value of m^* is MAX SNP-complete.

(Håstad 2001):
Unless P = NP, for every $\epsilon > 0$ there is no *polynomial-time algorithm* that can $(\frac{1}{2} + \epsilon)$-approximate m^*.

> *This is optimal since there is a trivial algorithm that can $\frac{1}{2}$-approximate m^*.*

Vertex Cover

In a graph $G = (V, E)$, $S \subseteq V$ is a *vertex cover* if each edge in E has at least one endpoint in S.

$\mathsf{vc}(G)$ is the *size* of the smallest vertex cover in G.

(Dinur-Safra 2005):
Unless $\mathsf{P} = \mathsf{NP}$, there is no polynomial-time algorithm that can approximate $\mathsf{vc}(G)$ up to a factor of 1.36.

> *Note 1: Since this is a minimization problem, the approximation ratio is a constant $c > 1$.*
> *Note 2: This has recently been improved to $\sqrt{2}$* **(Khot, Minzer, Safra 2018+)**.

There are polynomial-time algorithms that can approximate $\mathsf{vc}(G)$ up to a factor of 2.

Conjecture:
Unless $\mathsf{P} = \mathsf{NP}$, for every $\epsilon > 0$ there is no polynomial-time algorithm that can approximate $\mathsf{vc}(G)$ up to a factor of $2 - \epsilon$.

Methods

Say that a 3CNF formula is *c-satisfiable* if $m^* > cm$.

The proof of the PCP theorem gives (for some constant c) a reduction from 3SAT to itself which:

- maps a satisfiable formula to a satisfiable formula; and
- maps an unsatisfiable formula to one that is not c-satisfiable.

As a consequence, any class $\mathcal{C}$ of formulas that includes the satisifiable ones and excludes the ones that are *not* c-satisfiable, is NP-hard to decide.

The gap is amplified by further reductions, such as Håstad's long-code reductions.

In the case of 3XOR:

> *For any $\epsilon > 0$, any class $\mathcal{C}$ of formulas that includes the $(1-\epsilon)$-satisifiable ones and excludes the ones that are not $(\frac{1}{2}+\epsilon)$-satisfiable, is NP-hard to decide.*

Results

For any $\epsilon > 0$ there is no term of FPC which, interpreted in a 3CNF formula φ, defines a number guaranteed to be within $\frac{7}{8} + \epsilon$ of $m^*(\varphi)$.

For any $\epsilon > 0$ there is no term of FPC which, interpreted in a 3XOR formula φ, defines a number guaranteed to be within $\frac{1}{2} + \epsilon$ of $m^*(\varphi)$.

There is no term of FPC which, interpreted in a graph G, defines a value guaranteed to be within a factor 1.36 of $\mathsf{vc}(G)$.

New Challenges for Duplicator

The results are estabilshed by showing *definability gaps*:

If $\mathcal{C}$ is any class of 3CNF formulas that includes the satisfiable ones and excludes those that are not $(\frac{7}{8} + \epsilon)$-satisfiable, then $\mathcal{C}$ has counting width $\Omega(n^{\delta})$ for some $\delta > 0$.

If $\mathcal{C}$ is any class of 3XOR formulas that includes the satisfiable ones and excludes those that are not $(\frac{1}{2} + \epsilon)$-satisfiable, then $\mathcal{C}$ has counting width $\Omega(n^{\delta})$ for some $\delta > 0$.

Initial Gap

Unlike the PCP theorem, we establish an initial gap for 3XOR:

> *If $\mathcal{C}$ is any class of 3XOR formulas that includes the satisfiable ones and excludes those that are not $(\frac{1}{2} + \epsilon)$-satisfiable, then $\mathcal{C}$ has counting width $\Omega(n)$.*

We then *amplify the gap*, and extend it to 3SAT and *vertex cover* by means of *reductions* definable in *first-order logic*.

This involves showing that known polynomial-time reductions in the literature can be done in first-order logic.

Gap Construction

The initial gap is established by a variant of the *Cai-Fürer-Immerman* construction.

For a set V of n *variables*, choose uniformly at random, a collection of $m > n$ subsets $\{x_1, x_2, x_3\}$ of V of three elements.

> *With high probability, the resulting bipartite graph has certain expansion properties.*

Construct a system of equations $x_1 + x_2 + x_3 = b$ where the left-hand sides are the chosen sets and b is 0 or 1 based on the toss of a coin.

> *With high probability, the system is not $(\frac{1}{2} + \epsilon)$-satisfiable.*
>
> *The expansion properties guarantee that it is k-locally consistent for $k = \Omega(n)$.*

A *CFI* construction on this then gives a system that is not $(\frac{1}{2} + \epsilon)$-satisfiable but $\equiv^{C^k}$-equivalent to a satisfiable one.

Perspectives

FPC is a *subclass* of P that captures a natural notion of *symmetric algorithm*.

We are able to show both

- that powerful algorithmic techniques are expressible in FPC; and
- *unconditional* inexpressibility results for many problems.

The lower bound results reveal

- fundamental *structural* properties of the probems; and
- lower bounds on important algorithmic techniques.

Pointers

For the classical material in Lecture 1, you may consult a textbook such as: L. Libkin, *Elements of Finite Model Theory*.

The recent work in Lecture 2 may be found in the following papers:

M. Anderson, A. Dawar:
On Symmetric Circuits and Fixed-Point Logics. Theory Comput. Syst. (2017)

M. Anderson, A. Dawar, B. Holm:
Solving Linear Programs without Breaking Abstractions. J. ACM (2015)

A. Dawar, P. Wang:
Definability of Semidefinite Programming and Lasserre Lower Bounds for CSPs. LICS 2017

A. Atserias, A. Dawar:
Definable Inapproximability: New Challenges for Duplicator. CSL 2018

A. Atserias, A. Dawar, J. Ochremiak:
On the Power of Symmetric Linear Programs. LICS 2019