Definability of Linear Programming Problems

Anuj Dawar

University of Cambridge Computer Laboratory

Amsterdam, 22 June 2016

Recapitulation

Descriptive Complexity provides an alternative perspective on Computational Complexity.

For a first-order sentence φ, the class of its finite models can be decided in *polynomial time* and *logarithmic space*.

Existential second-order logic captures exactly the complexity class NP.

The search for a logic for P focusses on logics intermediate between first and second-order logic.

Recapitulation II

FP is a logic that extends first-order logic by means of *inductive definitions.*

On *linearly ordered structures*, FP exactly captures the complexity class P

In the absence of order, FP cannot express *evenness*. This is proved through a pebble game for L^k, first-order logic with k variables.

FPC is the extension of FP with a mechanism for *counting*. Its expressive power can be analyzed through a connection with C^k, first-order logic with k variables and *counting quantifiers*.

We aim to use this to show that solvability of linear systems of equations over $\mathbb{Z}_2$ is not definable in FPC..

Undefinability in FPC

To show that the *satisfiability* of systems of equations is not definable in FPC it suffices to show that for each k, we can construct two systems of equations

$$E_k \text{ and } F_k$$

such that:

- E_k is satisfiable;
- F_k is unsatisfiable; and
- $E_k \equiv^{C^k} F_k$

Constructing systems of equations

Take $\mathcal{G}$ a 3-regular, connected graph.
Define equations $\mathbf{E}_{\mathcal{G}}$ 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 incident on it, we have eight equations:

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

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

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

We can show: $\mathbf{E}_{\mathcal{G}}$ is satisfiable; $\tilde{\mathbf{E}}_{\mathcal{G}}$ 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 $\mathcal{G}$ such that $\mathbf{E}_{\mathcal{G}} \equiv^{C^k} \tilde{\mathbf{E}}_{\mathcal{G}}$.

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 column 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 Bijections

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

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

- A bijection $h : \mathbf{E}_{\mathcal{G}} \to \tilde{\mathbf{E}}_{\mathcal{G}}$ 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.

Computational Problems from Linear Algebra

Linear Algebra is a testing ground for exploring the boundary of the expressive power of FPC.
It may also be a possible source of new operators to extend the logic.

For a set I, and binary relation $A \subseteq I \times I$, take the matrix M over the two element field $\mathbb{Z}_2$:

$$M_{ij} = 1 \quad \Leftrightarrow \quad (i,j) \in A.$$

Most interesting properties of M are invariant under permutations of I.

Matrix Multiplication

We can write a formula prod(x, y, A, B) that defines the *product* of two matrices:

$$(\exists \nu_2 < t)(t = 2 \cdot \nu_2 + 1) \quad \text{for} \quad t = \#z(A(x,z) \wedge B(z,y))$$

A simple application of **ifp** then allows us to define upower(x, y, ν, A) which gives the matrix A^ν:

$$[\mathbf{ifp}_{R,uv\mu} \; (\mu = 0 \wedge u = v \vee \\ (\exists \mu' < \mu)\,(\mu = \mu' + 1 \wedge \mathsf{prod}(u, v, B/R(\mu'), A))](x, y, \nu),$$

where prod$(u, v, B/R(\mu'), A)$ is obtained from prod(u, v, A, B) by replacing the occurrence of $B(z, v)$ by $R(z, v, \mu')$.

Matrix Exponentiation

We can, instead, represent numbers up to $2^{|A|}$ in *binary*.
That is, a unary relation Γ interpreted over the number domain (using numbers up to $|A|$) codes the number $\sum_{\gamma \in \Gamma} 2^{\gamma}$.

Repeated squaring then allows us to define power(x, y, Γ, A) giving A^N where Γ codes a value N which may be exponential.

Non-Singularity

(Blass-Gurevich 04) show that *non-singularity* of a matrix over $\mathbb{Z}_2$ can be expressed in FPC.

GL$(n, \mathbb{Z}_2)$—the *general linear group* of degree n over $\mathbb{Z}_2$—is the group of non-singular $n \times n$ matrices over $\mathbb{Z}_2$.
The order of GL$(n, \mathbb{Z}_2)$ divides

$$N = \prod_{i=0}^{n-1} (2^n - 2^i).$$

Thus, A is *non-singular* if, and only if, $A^N = \mathbf{I}$
Moreover, the inverse A^{-1} is given by A^{N-1}.

Representing Finite Fields

We can represent matrices M over a finite field $\mathbb{F}_q$ by taking, for each $a \in \mathbb{F}_q$ a binary relation $A_a \subseteq I \times I$ with

$$M_{ij} = a \quad \Leftrightarrow \quad (i, j) \in A_a.$$

Alternatively, we could have the elements of $\mathbb{F}_q$ (along with the field operations) as a *separate sort* and include a ternary relation R

$$M_{ij} = a \quad \Leftrightarrow \quad (i, j, a) \in R.$$

These two representations are inter-definable.

FPC over Finite Fields

More generally, over the finite field $\mathbb{F}_q$, *matrix multiplication*; *non-singularity* of matrices; the *inverse* of a matrix; are all definable in FPC.

determinants and more generally, the coefficients of the *characteristic polynomial* can be expressed FPC.

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

solvability of systems of equations is *undefinable*.

the *rank* of a matrix is *undefinable*.

Linear Algebra over the Rational Field

Over the rational field $\mathbb{Q}$, we can also define *matrix multiplication*; *non-singularity* of matrices; the *inverse* of a matrix in FPC

Moreover, we can also define the coefficients of the *characteristic polynomial*

and, we can define the *rank* of a matrix and the *solvability* of systems of equations.

(Holm 2010)

The last result also follows from the stronger result that *optimization of linear programs* is expressible in FPC.

(Anderson, D., Holm 2015)

Representing Rational Numbers

We can take the rational number

$$q = s\frac{n}{d}$$

where $s\{1, -1\}$ and $n, d \in \mathbb{N}$
to be given by a structure

$$(B, <, S, N, D)$$

where $<$ is a linear order on the domain B and S, N and D are unary relations.

$S = \emptyset$ *iff* $s = 1$ and N and D code the binary representation of n and d.

Since the domain is ordered, it is straightforward to see that arithmetic, in the form of addition and multiplication of numbers is definable in FPC

Representing Rational Vectors and Matrices

A *rational vector* indexed by a set I:

$$v : I \to \mathbb{Q}$$

is represented by a structure over domain $I \cup B$ with relations:

- $<$ an order on B;
- $S, N, D \subseteq I \times B$

Similarly, a *rational matrix* $M \in \mathbb{Q}^{I \times J}$ is given by a structure over domain $I \cup J \cup B$ with relations:

- $<$ an order on B;
- $S, N, D \subseteq I \times J \times B$

Weighted Graphs

We use a similar encoding to represent problems over *weighted graphs* where the weights may be integer or rational.

For example, a graph with vertex set V with *non-negative rational* weights might be considered as a relational structure over universe $V \cup B$ where B is bigger than the number of bits required to represent any of the rational weights and we have

- $<$ an order on B;
- *weight relations* $W_n, W_d \subseteq V \times V \times B$

Linear Programming

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

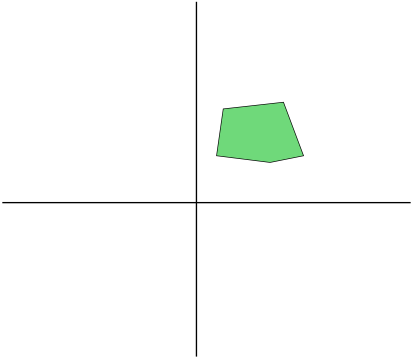
It was shown by **(Khachiyan 1980)** that linear programming problems can be solved in polynomial time.
We have a set C of *constraints* over a set V of $variables$.
Each $c \in C$ consists of $a_c \in \mathbb{Q}^V$ and $b_c \in \mathbb{Q}$.

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

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

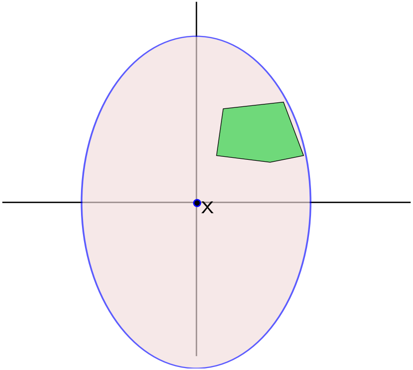
In **Anderson, D., Holm (2013)** we show that this, and the corresponding *optimization problem* are expressible in FPC.

Ellipsoid Method



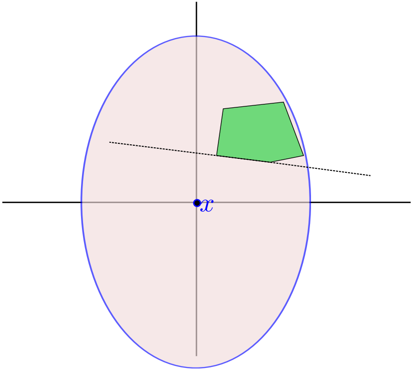
The set of constraints determines a *polytope*

Ellipsoid Method



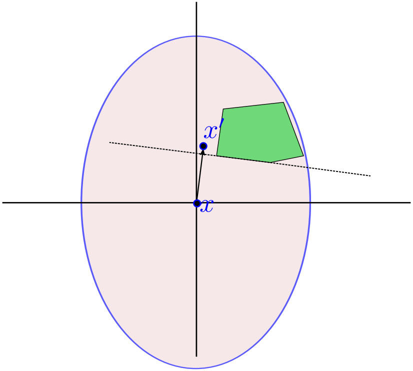
Start at the origin and calculate an *ellipsoid* enclosing it.

Ellipsoid Method



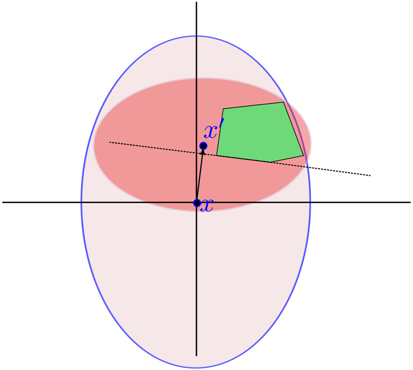
If the centre is not in the polytope, choose a constraint it *violates*.

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

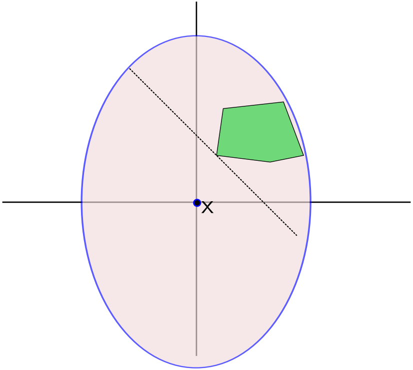
We can encode all the calculations involved in FPC.

This relies on expressing algebraic 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*.

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.

Anderson, D., Holm (2013) shows that as long as the *separation oracle* can be defined in FPC, the corresponding *optimization problem* can be solved in FPC.

Representations of Polytopes

A *representation* of a class $\mathcal{P}$ of *polytopes* is a *relational vocabulary* τ along with a surjective function ν taking τ-structures to polytopes in $\mathcal{P}$, which is isomorphism invariant.

A *separation oracle* for a representation $\nu, \mathcal{P}$ is definable in FPC if there is an FPC formula that given a τ-structure $\mathbb{A}$ and a vector $v \in \mathbb{Q}^V$ either

- determines that $v \in \nu(\mathbb{A})$; or
- defines a hyperplane separating v from $\nu(\mathbb{A})$.

Folding Polytopes

We use the separation oracle to define an *ordered equivalence relation* on the set V of variables.

We also define a *projection* operation on polytopes which either

- preserves feasibility; or
- refines the equivalence relation further.

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.

We saw that the existence of a *perfect matching* is not definable in FP.

(Blass, Gurevich, Shelah 1999) showed that for *bipartite* graphs this is definable in FPC.

They conjectured that this was *not* the case for general graphs.

We consider the more general problem of determining the *maximum weight* of a matching in a *weighted graph*:

$$G = (V, E) \quad w : E \to \mathbb{Q}_{\geq 0}$$

The Matching Polytope

(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 the following linear programming problem

$$\max w^\top y \qquad \text{subject to}$$

$$\begin{aligned} & Ay \leq 1^V, \\ & y_e \geq 0, \;\; \forall e \in E, \\ & \sum_{e \in E \cap W^2} y_e \leq \frac{1}{2}(|W| - 1), \;\; \forall W \subseteq V \text{ with } |W| \text{ odd}, \end{aligned} \tag{1}$$

Matching in FPC

We 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.