Maximum Matching and Linear Programming in Fixed-Point Logic with Counting

Matthew Anderson, Anuj Dawar, and Bjarki Holm

University of Cambridge Computer Laboratory
firstname.lastname@cl.cam.ac.uk

***Abstract*—We establish the expressibility in fixed-point logic with counting (FPC) of a number of natural polynomial-time problems. In particular, we show that the size of a maximum matching in a graph is definable in FPC. This settles an open problem first posed by Blass, Gurevich and Shelah [1], who asked whether the existence of perfect matchings in general graphs could be determined in the more powerful formalism of choiceless polynomial time with counting. Our result is established by noting that the ellipsoid method for solving linear programs of full dimension can be implemented in FPC. This allows us to prove that linear programs of full dimension can be optimised in FPC if the corresponding separation oracle problem can be defined in FPC. On the way to defining a suitable separation oracle for the maximum matching problem, we provide FPC formulas defining maximum flows and canonical minimum cuts in capacitated graphs.**

I. Introduction

The question of whether there is a logical characterisation of the class P of problems solvable in polynomial time, first posed by Chandra and Harel [2], has been a central research question in descriptive complexity for three decades. At one time it was conjectured that FPC, the extension of inflationary fixed-point logic by counting terms, would suffice to express all polynomial-time properties, but this was refuted by Cai, Fürer and Immerman [3]. Since then, a number of logics have been proposed whose expressive power is strictly greater than that of FPC but still contained within P. Among these are FPR, fixed-point logic with rank operators [4], and $\tilde{\text{C}}$PT(Card), choiceless polynomial time with counting [1], [5]. For both of these it remains open whether their expressive power is strictly weaker than P.

Although it is known that FPC does not express all polynomial-time computable properties, the descriptive power of FPC still forms a natural class within P. For instance, it has been shown that FPC can express all polynomial-time properties on many natural graph classes, such as any class of proper minor-closed graphs [6]. Delimiting the expressive power of FPC therefore remains an interesting challenge. In particular, it is of interest to establish what non-trivial polynomial-time algorithmic techniques can be expressed in this logic. The conjecture that FPC captures P was based on the intuition that the logic can define all "obvious" polynomial-time algorithms. The result of Cai et al. and the subsequent work of Atserias et al. [7] showed that one important technique—that of Gaussian elimination for matrices over finite fields—is not captured by FPC. The question remains what other natural problems for which membership in P is established by non-trivial algorithmic methods might be expressible in FPC.

For instance, it was shown by Blass et al. [5] that there is a sentence of FPC that is true in a bipartite graph G if, and only if, G contains a perfect matching. They posed as an open question whether the existence of a perfect matching on general graphs can be defined in $\tilde{\text{C}}$PT(Card) (see also [8], [9] for more on this open question). Indeed, this question first appears in [1] where it is stated that it seems "unlikely" that this problem can be decided in $\tilde{\text{C}}$PT(Card). One of our main contributions in this paper is to settle this question by showing that the size of a maximum matching in a general graph can be defined in FPC (and therefore also in $\tilde{\text{C}}$PT(Card)).

On the way to establishing this result, we show that a number of other interesting problems can be defined in FPC. We begin by showing that FPC can define feasible points in full-dimensional polyhedra. That is, there is an FPC formula which, for a full-dimensional compact convex set in Euclidean space given by finite intersections of linear inequalities called *constraints* (suitably represented as a relational structure, without an ordering on the set of variables or constraints), defines a point inside the set if one exists.

Our FPC-definable point is obtained by encoding a version of Khachiyan's ellipsoid method [10]. It is not difficult to see that the numerical calculations required in that method can all be expressed in FPC, since this logic can express all polynomial-time properties of ordered structures. The key stumbling block is that the ellipsoid method requires us, at each iteration, to *choose* a constraint that is violated by the centre of the current ellipsoid containing the polytope (if there is no such constraint, then the centre provides us with our feasible point). This chosen constraint is then used to reduce the volume of the ellipsoid and a new centre is calculated. In order to obtain a *choiceless* version of this algorithm, we note that the ellipsoid method is robust under which violated constraint is used. Indeed, a linear combination of violated constraints suffices. With this in mind, we take, at each iteration, the *sum* of all violated constraints. This yields an implementation of the ellipsoid method in FPC, which gives us our first result.

In many applications, the set of constraints is not given explicitly (indeed, it may be exponentially large) but is determined instead by a *separation oracle*. This is a procedure which, given a candidate point x, determines whether x is feasible and, if it is not, returns a constraint that is violated by x. We show that as long as a separation oracle for a full-dimensional polyhedron is itself expressible in FPC, then the linear optimisation problem on the polyhedron is expressible in FPC (that is, there is an FPC interpretation defining a point in the polyhedron which maximises a given linear function). This parallels the classical polynomial-time reduction from optimisation to separation (c.f., [11]). These results are presented in more detail in Section III.

As a first application of the FPC-definability of explicitly-given linear programs, we show that a maximum flow in a capacitated graph is definable in FPC. Indeed, this follows rather directly from the first result, since the flow polytope is of size polynomial in G and explicitly given, and hence a separation oracle can be easily defined from G in FPC. These results are presented in Section IV.

Next, we use the definability of maximum flows to show that minimum cuts are also definable in FPC. That is, in the vocabulary of capacitated graphs, there is a formula which defines a set of vertices C corresponding to a minimum value cut separating s from t. The cut C defined in this way is canonical in a strong sense, in that we show that it is the *smallest* (under set-inclusion) minimum cut separating s from t. These results are discussed in Section V.

Finally, we turn to the maximum matching problem. For a graph $G = (V, E)$, the matching polytope is given by a set of constraints of size exponential in the size of G. We show that there is a separation oracle for this set definable from G in FPC, using the definability of minimum cuts. To be precise, we use the fact that a separation oracle for the matching polytope can be obtained from a computation of minimum odd-size cuts in a graph [12]. In Section VI we prove that there is always a pair of vertices s, t such that a canonical minimum (s, t)-cut is a minimum odd-size cut. This, combined with the definability of canonical minimum cuts gives us the separation oracle for matching that we seek. Note that it is not possible in general to actually define a canonical maximum matching. To see this, consider K_n, the complete graph on n vertices. This graph contains an exponential number of maximum matchings and for any two of these matchings, there is an automorphism of the graph taking one to the other. Thus, it is not possible for any formula of FPC (which is necessarily invariant under isomorphisms) to pick out a particular matching. What we can do, however, is to define a formula that gives the size of the maximum matching in a graph. This, in turn, enables us to write a sentence of FPC that is true in a graph G if, and only if, it contains a perfect matching. Our results on matchings are presented in Section VII.

For brevity, we defer most proofs to the complete version of this paper [13]. The complete version also contains a generalisation of our linear-programming results, for polytopes which may not have full dimension.

II. Background

We write $[n]$ to denote the set of positive integers $\{0, \ldots, n-1\}$. Given sets I and A, a column vector u over A indexed by I is a function $u : I \to A$, and we write A^I for the set of all such vectors. Similarly, an I, J-matrix over A is a function $M : I \times J \to A$ and we write M_{ij} for $M(i, j)$ and M_i for the row (vector) of M indexed by i. For an integer z, $|z|$ denotes its absolute value. For a vector $v \in \mathbb{Q}^I$, $\|v\|$ denotes its Euclidean norm and $\|v\|_\infty := \max_{i \in I} |v_i|$ denotes its infinity norm.

A. Logics and Structures

A relational *vocabulary* τ is a finite sequence of relation and constant symbols $(R_1, \ldots, R_k, c_1, \ldots, c_\ell)$, where every relation symbol R_i has a fixed *arity* $a_i \in \mathbb{N}$. A structure $\mathbf{A} = (\mathrm{dom}(\mathbf{A}), R_1^{\mathbf{A}}, \ldots, R_k^{\mathbf{A}}, c_1^{\mathbf{A}}, \ldots, c_\ell^{\mathbf{A}})$ over the vocabulary τ (or a τ-*structure*) consists of a non-empty set $\mathrm{dom}(\mathbf{A})$, called the *universe* of $\mathbf{A}$, together with relations $R_i^{\mathbf{A}} \subseteq \mathrm{dom}(\mathbf{A})^{a_i}$ and constants $c_j^{\mathbf{A}} \in \mathrm{dom}(\mathbf{A})$ for each $1 \leq i \leq k$ and $1 \leq j \leq \ell$. Members of the set $\mathrm{dom}(\mathbf{A})$ are called the *elements* of $\mathbf{A}$ and we define the *size* of $\mathbf{A}$ to be the cardinality of its universe. In what follows, we often consider multi-sorted structures. That is, $\mathrm{dom}(\mathbf{A})$ is given as the disjoint union of a number of different *sorts*. In this paper we consider only structures over a finite universe and for a particular vocabulary τ we use $\mathrm{fin}[\tau]$ to denote the set of all finite τ-structures.

a) Fixed-point logic with counting: Fixed-point logic with counting (FPC) is an extension of inflationary fixed-point logic with the ability to express the cardinality of definable sets. The logic has two types of first-order variable: *element variables*, which range over elements of the structure on which a formula is interpreted in the usual way, and *number variables*, which range over some initial segment of the natural numbers.

The atomic formulas of FPC$[\tau]$ are all formulas of the form: $\mu = \eta$ or $\mu \leq \eta$, where μ, η are number variables; $s = t$ where s, t are element variables or constant symbols from τ; and $R(t_1, \ldots, t_m)$, where each t_i is either an element variable or a constant symbol and R is a relation symbol of arity m. The set FPC$[\tau]$ of FPC *formulas* over τ is built up from the atomic formulas by applying an inflationary fixed-point operator $[\mathbf{ifp}_{R,\vec{x}}\phi](\vec{t})$; forming *counting terms* $\#_x\phi$, where ϕ is a formula and x an element variable; forming formulas of the kind $s = t$ and $s \leq t$ where s, t are number variables or counting terms; as well as the standard first-order operations of negation, conjunction, disjunction, universal and existential quantification. Collectively, we refer to element variables and constant symbols as *element terms*, and to number variables and counting terms as *number terms*.

For the semantics, number terms take values in $[n+1]$ and element terms take values in $\mathrm{dom}(\mathbf{A})$ where $n := |\mathrm{dom}(\mathbf{A})|$. The semantics of atomic formulas, fixed-points and first-order operations are defined as usual (c.f., e.g., [14] for details), with comparison of number terms $\mu \leq \eta$ interpreted by comparing the corresponding integers in $[n + 1]$. Finally, consider a counting term of the form $\#_x\phi$, where ϕ is a formula and x

an element variable. Here the intended semantics is that $\#_x\phi$ denotes the number (i.e., the element of $[n+1]$) of elements that satisfy the formula ϕ.

In general, a formula $\phi(\vec{x}, \vec{\mu})$ of FPC defines a relation over $\mathrm{dom}(\mathbf{A}) \uplus [n+1]$ that is invariant under automorphisms of $\mathbf{A}$. For a more detailed definition of FPC, we refer the reader to [14], [15].

b) Logical interpretations: We frequently consider ways of defining one structure within another in some logic L, such as first-order logic or fixed-point logic with counting. Consider two vocabularies σ and τ and a logic L. An *m-ary* L*-interpretation of* τ *in* σ is a sequence of formulae of L in vocabulary σ consisting of: (i) a formula $\delta(\vec{x})$; (ii) a formula $\varepsilon(\vec{x}, \vec{y})$; (iii) for each relation symbol $R \in \tau$ of arity k, a formula $\phi_R(\vec{x}_1, \dots, \vec{x}_k)$; and (iv) for each constant symbol $c \in \tau$, a formula $\gamma_c(\vec{x})$, where each $\vec{x}$, $\vec{y}$ or $\vec{x}_i$ is an m-tuple of free variables. We call m the *width* of the interpretation. We say that an interpretation Θ associates a τ-structure $\mathbf{B}$ to a σ-structure $\mathbf{A}$ if there is a surjective map h from the m-tuples $\{\vec{a} \in (\mathrm{dom}(\mathbf{A}) \uplus [n+1])^m \mid \mathbf{A} \models \delta[\vec{a}]\}$ to $\mathbf{B}$ such that:

- $h(\vec{a}_1) = h(\vec{a}_2)$ if, and only if, $\mathbf{A} \models \varepsilon[\vec{a}_1, \vec{a}_2]$;
- $R^{\mathbf{B}}(h(\vec{a}_1), \dots, h(\vec{a}_k))$ if, and only if, $\mathbf{A} \models \phi_R[\vec{a}_1, \dots, \vec{a}_k]$;
- $h(\vec{a}) = c^{\mathbf{B}}$ if, and only if, $\mathbf{A} \models \gamma_c[\vec{a}]$.

Note that an interpretation Θ associates a τ-structure with $\mathbf{A}$ only if ε defines an equivalence relation on $(\mathrm{dom}(\mathbf{A}) \uplus [n+1])^m$ which is a congruence with respect to the relations defined by the formulae ϕ_R and γ_c. In such cases, however, $\mathbf{B}$ is uniquely defined up to isomorphism and we write $\Theta(\mathbf{A}) := \mathbf{B}$.

It is not difficult to show that formulas of FPC compose with reductions in the sense that, given an interpretation Θ of σ in τ and a σ-formula ϕ, we can define a τ-formula ϕ' such that $\mathbf{A} \models \phi'$ if, and only if, $\Theta(\mathbf{A}) \models \phi$ (see [16, Sec. 3.2]).

B. Numbers, Vectors and Matrices

Let z be an integer, $b \geq \lceil \log_2(|z|) \rceil$, $B = [b]$ and write $\mathrm{bit}(x, k)$ to denote the k-th least-significant bit in the binary expansion of $x \in \mathbb{N}$. We view the integer $z = s \cdot x$ as a product of a sign $s \in \{-1, 1\}$ and a natural number x. We can represent z as a single-sorted structure $\mathbf{B}$ on a domain of bits B over the vocabulary $\tau_{\mathbb{Z}} := \{X, S, \leq_B\}$. Here $\leq_B$ is interpreted as a linear ordering of B, the unary relation S indicates that the sign s of the integer is 1 if $S^{\mathbf{B}} = \emptyset$ and -1 otherwise, and the unary relation X is interpreted as $X^{\mathbf{B}} = \{k \in B \mid \mathrm{bit}(x, k) = 1\}$. That is $k \in X^{\mathbf{B}}$ when the "the k-th bit in the binary expansion of x is 1." Similarly we consider a *rational number* $q = s \cdot \frac{x}{d}$ as a structure on the domain of bits B over $\tau_{\mathbb{Q}} := \{X, D, S, \leq_B\}$, where X and S are as before and D is interpreted as the binary encoding of the denominator d when $D^{\mathbf{B}} \neq \emptyset$.

We now generalise these notions and consider unordered tensors over the rationals (the case of integers is completely analogous). Let $J_1, \dots, J_r$ be a family of finite non-empty sets. An unordered tensor T over $\mathbb{Q}$ is a function $T : J_1 \times \cdots \times J_r \to \mathbb{Q}$. We write $t_{j_1 \dots j_r} = s_{j_1 \dots j_r} \frac{x_{j_1 \dots j_r}}{d_{j_1 \dots j_r}}$ to denote the element of T indexed by $(j_1, \dots, j_r) \in J_1 \times \cdots \times J_r$. Writing $m \in \mathbb{N}$ for the the maximum absolute value of integers appearing as either numerators or denominators of elements in the range of T, let $b \geq \lceil \log_2(|m|) \rceil$ and $B = [b]$. The tensor T is then an $(r+1)$-sorted structure $\mathbf{T}$ with r index sorts $J_1, \dots, J_r$ and a bit sort B over the vocabulary $\sigma_{\text{ten},r} := \{X, D, S, \leq_B\}$. Here $\leq_B$ is interpreted as before, the $(r+1)$-ary relation S is interpreted as indicating the value of the sign $s_{j_1 \dots j_r} \in \{-1, 1\}$ as before, the $(r+1)$-ary relation X is interpreted as

$$\{(j_1, \dots, j_r, k) \in J_1 \times \cdots \times J_r \times B \mid \mathrm{bit}(x_{j_1 \dots j_r}, k) = 1\},$$

and the $(r+1)$-ary relation D is similarly interpreted as the binary representation of the denominators of T. We are only interested in the case of rational vectors and matrices and so define the vocabularies $\tau_{\text{vec}} := \sigma_{\text{ten},1}$ and $\tau_{\text{mat}} := \sigma_{\text{ten},2}$.

In [17] it is shown that a variety of basic linear-algebraic operations on rational vectors and matrices described in this way can be expressed in fixed-point logic with counting. These include computing equality, norms, dot product, matrix product, determinant and inverse.

C. Linear Programming

We recall some basic definitions from polyhedral combinatorics and linear optimisation. For further background, see, for example, the textbook by Grötschel et al. [11].

1) Geometry: Consider the rational Euclidean space $\mathbb{Q}^V$ indexed by a set V. For a point $x \in \mathbb{Q}^V$ and $\varepsilon \in \mathbb{Q}_{\geq 0}$ let $\mathrm{S}(x, \varepsilon) := \{y \in \mathbb{Q}^V \mid \|x - y\| \leq \varepsilon\}$ denote the *sphere* of radius ε about x. This notion intuitively extends to $\mathrm{S}(K, \varepsilon) := \bigcup_{x \in K} \mathrm{S}(x, \varepsilon)$ for any set $K \subseteq \mathbb{Q}^V$. A *positive definite* matrix $E \in \mathbb{Q}^{V \times V}$, i.e., E is symmetric and has only positive eigenvalues, and point $x \in \mathbb{Q}^V$ specify an ellipsoid

$$\mathrm{Ell}(E, x) := \{y \in \mathbb{Q}^V \mid (y - x)^\top E^{-1} (y - x) \leq 1\}.$$

Because E is positive definite, $\mathrm{Ell}(E, x)$ has *full dimension* in $\mathbb{Q}^V$, i.e., it contains a sphere of non-zero volume.

a) Polytopes: The solutions to a system of linear equalities and inequalities over $\mathbb{Q}^V$ is the intersection of some number of *half-spaces* of the kind $\{x \in \mathbb{Q}^V \mid a^\top x \leq b\}$ specified by the *constraint* $a^\top x \leq b$, where $a \in \mathbb{Q}^V$ and $b \in \mathbb{Q}$. A (rational) *polytope* is a convex set $P \subseteq \mathbb{Q}^V$ which is the intersection of a *finite* number of half-spaces. That is to say, there are a set of constraints C, a *constraint matrix* $A \in \mathbb{Q}^{C \times V}$ and vector $b \in \mathbb{Q}^C$, such that $P = P_{A,b} := \{x \in \mathbb{Q}^V \mid Ax \leq b\}$. A polytope which is also compact is called a *polyhedron*.

The *size*, or bit complexity, of rational vector c (denoted $\langle c \rangle$) is the number of bits required to encode c in some standard encoding. Note that $\langle c \rangle$ is at least the dimension of c. The size of a constraint $a^\top x \leq b$ is then $\langle \binom{a}{b} \rangle$. If $Ax \leq b$ is a system of linear inequalities, its size is the maximum over the sizes of its individual constraints. Note that this measure is explicitly independent of the number of constraints in the system. The size of a polytope P is the minimum over the sizes of the systems $Ax \leq b$ such that $P = P_{A,b}$.

A polyhedron P is said to be *R-circumscribed* when there is an $R \in \mathbb{Q}_{\geq 0}$ such that $P \subseteq \mathrm{S}(0^V, R)$. A polytope is *bounded* if it has full dimension and is circumscribed.

2) Problems on Polytopes: We are interested in two main combinatorial problems on polytopes: *linear optimisation* and *separation*.

Problem 1 (Linear Optimisation)**.** *Let V be a set, $P \subseteq \mathbb{Q}^V$ be a polytope and $c \in \mathbb{Q}^V$. The* linear optimisation problem *on P is the problem of determining either (i) an element $y \in P$ such that $c^\top y = \max\{c^\top x \mid x \in P\}$, (ii) that $P = \emptyset$, or (iii) that P is unbounded in the direction of c.*

An instance of the linear optimisation problem is called a *linear program* and the linear function $x \mapsto c^\top x$ is called the *objective function*.

Problem 2 (Separation)**.** *Let V be a set, $P \subseteq \mathbb{Q}^V$ be a polytope and $y \in \mathbb{Q}^V$. The* separation problem *on P is the problem of determining either (i) that $y \in P$ or (ii) a vector $c \in \mathbb{Q}^V$ with $c^\top y > \max\{c^\top x \mid x \in P\}$ and $\|c\|_\infty = 1$.*

Over families of rational polytopes, the optimisation and separation problems are polynomial-time equivalent (c.f., e.g., [11, Theorem 6.4.9]). Here the time bound is measured in the size of the polytope and all other parameters of the problem.

D. Representation

When we deal with polytopes as objects in a computation, we need to choose a representation which gives a finite description of a polytope. In particular, in dealing with logical definability of problems on polytopes, we need to choose a representation of polytopes by relational structures.

Definition 3. *A* representation *of a class $\mathcal{P}$ of polytopes is a relational vocabulary τ along with an* onto *function $\nu : \mathrm{fin}[\tau] \to \mathcal{P}$ which is isomorphism invariant, that is, $\mathbf{A} \cong \mathbf{B}$ implies $\nu(\mathbf{A}) \cong \nu(\mathbf{B})$.*

For concreteness, consider the vocabulary $\tau := \tau_{\text{mat}} \uplus \tau_{\text{vec}}$ obtained by taking the disjoint union of the vocabularies for rational matrices and vectors. A τ-structure over a universe consisting of a set V of variables and a set C of constraints describes a constraint matrix $A \in \mathbb{Q}^{C \times V}$ and bound vector $b \in \mathbb{Q}^C$. Thus, the function taking such a structure to the polytope $P_{A,b}$ is a representation of the class of rational polytopes. We call this the *explicit representation*.

Note that the explicit representation of polytopes has the property that both the size of the polytope (i.e., the maximum size of any constraint) and the number of constraints of $\nu(\mathbf{A})$ are polynomially bounded in the size of $\mathbf{A}$. We will also be interested in representations ν where the number of constraints in $\nu(\mathbf{A})$ is exponential in $|\mathbf{A}|$, but we always confine ourselves to representations where the size of the constraints is bounded by a polynomial in $\mathbf{A}$. We formalise this by saying that a representation ν is *well described* if there is a polynomial p such that $\langle \nu(\mathbf{A}) \rangle = p(|\mathbf{A}|)$, for all τ-structures $\mathbf{A}$. In particular, in all representations we consider the *dimension* of the polytope $\nu(\mathbf{A})$ is bounded by a polynomial in $|\mathbf{A}|$.

We are now define what it means to express the linear optimisation, separation and circumscription problems in FPC.

Definition 4. *We say that the linear optimisation problem for a class of polytopes $\mathcal{P}$ is expressible in* FPC *with respect to a representation $\nu : \mathrm{fin}[\tau] \to \mathcal{P}$ if there is an* FPC *interpretation of $\tau_{\mathbb{Q}} \uplus \tau_{vec}$ in $\tau \uplus \tau_{vec}$ which takes a structure coding a τ-structure $\mathbf{A}$ and a vector c to a number f and a vector y such that either (i) $f = 0$, $y \in \nu(\mathbf{A})$ and $c^\top y = \max\{c^\top x \mid x \in \nu(\mathbf{A})\}$, (ii) $f = 1$, $y = 0$ and $\nu(\mathbf{A}) = \emptyset$, or (iii) $f = 1$, $y = 1$ and $\nu(\mathbf{A})$ is unbounded in the direction of c.*

Definition 5. *The separation problem for a class of polytopes $\mathcal{P}$ is expressible in* FPC *with respect to a representation $\nu : \mathrm{fin}[\tau] \to \mathcal{P}$ if there is an* FPC *interpretation of τ_{vec} in $\tau \uplus \tau_{vec}$ which takes a structure coding a τ-structure $\mathbf{A}$ and a vector y to a vector c such that either (i) $y \in \nu(\mathbf{A})$ and $c = 0$, or (ii) $c \in \mathbb{Q}^V$ with $c^\top y > \max\{c^\top x \mid x \in \nu(\mathbf{A})\}$ and $\|c\|_\infty = 1$.*

Definition 6. *The circumscription problem for a class of polytopes $\mathcal{P}$ is expressible in* FPC *with respect to a representation $\nu : \mathrm{fin}[\tau] \to \mathcal{P}$ if there is an* FPC *interpretation of $\tau_{\mathbb{Q}}$ in τ which takes a τ-structure to a rational R such that $\nu(\mathbf{A}) \subseteq \mathrm{S}(0, R)$.*

III. Linear Programs and Separation Oracles

Linear programming has a rich history in complexity theory. Early work by Dantzig [18] gave a combinatorial algorithm—the *simplex method*—for solving linear programs. The algorithm traverses the vertices of the polytope favouring vertices that improve the objective value. Although the simplex method has been useful in practice, it tends to be less useful in theory because strong worst-case performance guarantees are not known (stronger guarantees are known for its average-case and smoothed complexity [19]). The 1970s saw a series of works studying linear programming from a geometric perspective (e.g., [20]). This line culminated in the breakthrough of Khachiyan [10] which established a polynomial-time algorithm—the *ellipsoid method*—for solving linear programs. Subsequent work (e.g., [21]) showed that the ellipsoid method is quite robust and can in fact be used to solve more general optimisation problems over convex sets, provided there is a means of solving the separation problem for those sets (briefly, if there exists a *separation oracle*). Indeed, even optimisation problems with an exponential number of constraints may be solved provided a separation oracle.

Section III-A discusses a simplified version of the ellipsoid method. Section III-B shows how to express the simplified ellipsoid method in FPC and, in doing so, gives a separation oracle for polytopes represented explicitly by an input matrix and vector. Section III-C discusses how the classic polynomial-time reduction from optimisation to separation on rational bounded polytopes can be expressed in FPC.

A. Overview of the Ellipsoid Method

The most basic version of the ellipsoid method takes as input an explicit description of a full-dimension rational poly-

Fig. 1. A simplified Ellipsoid Method.

$\mathrm{SEM}(A, b, R)$

Input: $A \in \mathbb{Q}^{C \times V}$, $b \in \mathbb{Q}^C$, $R \in \mathbb{Q}$ such that $P_{A,b}$ is in $\mathrm{S}(0, R)$ and has full dimension.
Output: $y \in P_{A,b}$.

1: Set starting ellipsoid $\mathrm{Ell}(E^0, y^0)$ to sphere of radius R around the origin: $E^0 \leftarrow R^2 \cdot I_V$ and $y^0 \leftarrow 0^V$.
2: **for** $i \in \{0, 1, 2, \ldots\}$ **do**
3: **if** $Ay^i \leq b$ **then return** y^i.
4: Select $k \in C$ such that $A_k y^i > b_k$.
5: $c \leftarrow A_k$.
6: $y^{i+1} \leftarrow y^i - \frac{1}{|V|+1} \frac{E^i c}{\sqrt{c^\top E^i c}}$.
7: $E^{i+1} \leftarrow \frac{2|V|^2+3}{2|V|^2} \left(E^i - \frac{2}{|V|+1} \frac{E^i c c^\top E^i}{c^\top E^i c} \right)$.

tope $P_{A,b}$, which is contained in a sphere of radius R about the origin, and outputs a point $y \in P_{A,b}$. Intuitively, the ellipsoid method is a geometric version of binary search. Start with an ellipsoid (the sphere $\mathrm{S}(0, R)$) containing the entire polytope. If, at any step, the centre of the current ellipsoid is contained within the polytope, then the search is done. Otherwise, there is a violated constraint by the ellipsoid centre and hence there is a hyperplane through the ellipsoid centre which divides the ellipsoid in half without intersecting the polytope. In this case, we select a new ellipsoid which contains the half of the current ellipsoid that contains the polytope, and the procedure is repeated. This algorithm (SEM) is described in Figure 1.

Since $P_{A,b}$ has full dimension, it has nonzero volume. It can be shown that the volume of the ellipsoids generated by SEM decrease to less than the volume of $P_{A,b}$ within a polynomial number of steps in the size of the polytope. Hence the algorithm must locate a point y within the polytope in polynomial time [10]. The ellipsoid method has the remarkable property that its correctness is independent of the separating hyperplane chosen; in addition the hyperplane's bit size contributes only polynomially to the running time [11].

B. *Expressing the Simplified Ellipsoid Method in* FPC

In this section we show that in fixed-point logic with counting, we can define an element of any explicitly represented bounded polytope.

Theorem 7 (Simplified Ellipsoid Method)**.** *There is an* FPC-*interpretation of τ_{vec} in $\tau_{mat} \uplus \tau_{vec} \uplus \tau_{\mathbb{Q}}$ which takes a structure coding $A \in \mathbb{Q}^{C \times V}$, $b \in \mathbb{Q}^C$ and $R \in \mathbb{Q}$, for which $P_{A,b} \subseteq \mathrm{S}(0, R)$ and has full dimension, to a vector $y \in P_{A,b}$.*

To prove this theorem, it suffices to show that the SEM algorithm can be expressed in FPC. To that end, we note that in FPC we can express the relevant manipulations on rational values, vectors and matrices, such as addition and multiplication [17], even when they are indexed by unordered sets. The potential irrationality of square-roots (line 6 of the algorithm) can be dealt with by appropriate rounding to a definable precision. It can also be seen that the iteration of the algorithm can be simulated by a fixed-point computation to a fixed polynomial degree. Overall, it is not hard to see that all of the steps in SEM can be expressed in fixed-point logic with counting, save one: line 4.

To explain where the problem lies, consider $A \in \mathbb{Q}^{C \times V}$ and $b \in \mathbb{Q}^C$. Given a point $y \in \mathbb{Q}^V$, it can be expressed in FPC whether y satisfies $Ay \leq b$ and, if not, it can be seen that there is a formula of FPC which defines a subset of C corresponding to the violated constraints. However, the logic is in general not able to *choose* a particular constraint from the unordered set C. Our key observation here is that linearity implies that the sum of all such violated constraints is itself a violated constraint for non-empty polytopes and hence the choice made by SEM is superfluous. This can be formally stated as follows.

Proposition 8. *Let $A \in \mathbb{Q}^{C \times V}$, $b \in \mathbb{Q}^C$, $x \in \mathbb{Q}^V$ and $C \supseteq S \neq \emptyset$. Suppose $P_{A,b}$ is non-empty and $(Ax)_s \not\leq b_s$ for all $s \in S$. Define $a_S := \sum_{s \in S} A_s$. Then $a_S^\top x > \max\{a_S^\top y \mid y \in P_{A,b}\}$ and $a_S \neq 0^V$.*

Proof. Define $b_S := \sum_{s \in S} b_s$. That $a_S^\top x > b_S$ is immediate from linearity. Since the polytope is non-empty pick any point $y \in P_{A,b}$. By definition, $Ay \leq b$. Linearity implies that $a_S^\top y \leq b_S$. Thus $a_S^\top x > b_S \geq \max\{a_S^\top y \mid y \in P_{A,b}\}$. This also implies that $a_S \neq 0^V$. □

This observation leads to a definition in FPC of the separation problem for $P_{A,b}$ with respect to x. Specifically, let $S \subseteq C$ be the set of constraints which violate the inequality $Ax \leq b$. This set can be defined by a FPC formula using rational arithmetic. If S is empty, expressing $c = 0^V$ correctly indicates that $x \in P_{A,b}$. Otherwise S is non-empty; let a_S be the sum of the constraints which x violates. Since the set S is definable in FPC so is the sum of constraints indexed by S. If $a_S \neq 0^V$, Proposition 8 implies that expressing c as the division of a_S by its (non-zero) infinity norm correctly indicates a separating hyperplane for $P_{A,b}$ through x; moreover, both operations are in FPC. Otherwise, $a_S = 0^V$ and Proposition 8 indicates that $P_{A,b}$ is empty. This means that any non-zero vector defines a separating hyperplane for $P_{A,b}$. Thus it suffices for the interpretation to express the vector $c = 1^V$. Overall, the above discussion gives an FPC interpretation expressing the separation problem for $P_{A,b}$.

Theorem 9. *There is an* FPC *interpretation of τ_{vec} in $\tau_{mat} \uplus \tau_{vec} \uplus \tau_{vec}$ expressing the separation problem for the class of polytopes explicitly given by constraints and represented naturally as $\tau_{mat} \uplus \tau_{vec}$-structures.*

We conclude the sketch proof of Theorem 7 by plugging the FPC interpretation given by Theorem 9 in for lines 3–5 of SEM. The correctness and convergence time of the algorithm are preserved because the standard analysis of the ellipsoid method is independent of the particular separating hyperplane.

C. *Optimisation from Separation in* FPC

We show the following result parallelling the established Turing-reduction from linear optimisation to separation.

Theorem 10 (Optimisation from Separation)**.** *Let $\mathcal{P}$ be a class of bounded polytopes with a well-described representation ν over the vocabulary τ. Let there be* FPC *interpretations of τ_{vec} in $\tau \uplus \tau_{vec}$ and of $\tau_{\mathbb{Q}}$ in τ, respectively, expressing the separation and circumscription problems for $\mathcal{P}$ with respect to ν. There is an* FPC *interpretation of $\tau_{\mathbb{Q}} \uplus \tau_{vec}$ in $\tau \uplus \tau_{vec}$ expressing the optimisation problem for $\mathcal{P}$ with respect to ν.*

Observe that this does not imply that every linear program over bounded polytopes can be solved in FPC. Indeed, the polynomial-time setting shares this same limitation. Rather, we can solve particular classes of linear programming problems where domain knowledge can be used to express the separation and circumscription problems in FPC. We get the following generic consequence when Theorem 10 is combined with Theorem 9 in the case of explicitly given polytopes.

Theorem 11 (Simplified Optimisation)**.** *There is an* FPC*-interpretation of τ_{vec} in $\tau_{mat} \uplus \tau_{vec} \uplus \tau_{vec} \uplus \tau_{\mathbb{Q}}$ which takes a structure coding $A \in \mathbb{Q}^{C\times V}$, $b \in \mathbb{Q}^C$, $c \in \mathbb{Q}^V$ and $R \in \mathbb{Q}$, for which $P_{A,b} \subseteq \mathrm{S}(0, R)$ and has full dimension, to a vector $y \in P_{A,b}$ such that $c^\top y = \max\{c^\top x \mid x \in P_{A,b}\}$.*

The SEM may be used as a subroutine to solve the linear optimisation problem on bounded polytopes by binary searching for the optimal objective value and thus prove Theorem 10. The search tests against a possible value γ by adding the corresponding linear constraint $-c^\top x \leq -\gamma$ to the polytope and using SEM to look for an element in the modified polytope. Simultaneously rounding the solution with largest objective value to have a common denominator with bit length proportional to the size of the polytope ensures an exact rational solution to the linear program (c.f., e.g., [11, Theorem 6.3.2.(a)]). We argue that in FPC we can simulate the behaviour of the combined algorithm and hence much of the analysis follows immediately from the classical argument. This leaves our proof to focus on showing that each step of the algorithm can be faithfully translated into FPC.

IV. Application: Maximum Flow

Let $G = (V, c)$ be a graph with non-negative edge capacities, that is, $c : V \times V \to \mathbb{Q}_{\geq 0}$. For a pair of distinct vertices $s, t \in V$ an *(s,t)-flow*is a function $f : V \times V \to \mathbb{Q}_{\geq 0}$ satisfying capacity constraints $0 \leq f(u, v) \leq c(u, v)$ for each pair of distinct $u, v \in V$ and conservation constraints $\sum_{v\in V}(f(v, u) - f(u, v)) \geq 0$ for all vertices $u \in V\backslash\{s\}$. The *value* $\mathrm{val}(f)$ of the flow f is simply the difference in the in-flow and out-flow at t: $\sum_{v\in V}(f(v,t)-f(t,v))$. A *maximum (s,t)-flow of G* is a flow whose value is maximum over all (s,t)-flows. The standard formulation of the maximum (s,t)-flow problem as a linear program is as follows:

$$\begin{aligned} &\max \sum_{v\in V}(f(v,t) - f(t,v)) \qquad \text{subject to} \\ &\sum_{v\in V}(f(v,u) - f(u,v)) \geq 0, \quad \forall u \in V\backslash\{s\} \\ &0 \leq f(u,v) \leq c(u,v), \quad \forall u \neq v \in V. \end{aligned} \tag{1}$$

Observe that there are $|V|(|V| - 1)$ variables in linear program (1) corresponding to $f(u, v)$ for distinct $u, v \in V$. The program has $2|V|^2 - 2$ constraints. Both the variables and constraints can be indexed by tuples of elements from V. It can easily be established that the maximum (s,t)-flow linear program can be defined in FPC. That is to say, suppose that a capacitated graph (V, c) is given as a τ_{mat}-structure with universe V where the rational matrix $c \in \mathbb{Q}^{V\times V}_{\geq 0}$ codes the capacities. Then, there is an FPC interpretation from τ_{mat} to $\tau_{\mathrm{mat}} \uplus \tau_{\mathrm{vec}}$ that takes a capacitated graph (V, c) and a pair $s, t \in V$ and explicitly expresses a constraint matrix A and vector b encoding the corresponding flow polytope.

In order to apply Theorem 11 to solve this optimisation problem we must establish that the flow polytope is bounded and has FPC interpretations expressing the separation and circumscription problems. The FPC interpretation for separation is immediate from Theorem 9 because the constraints are explicitly given. For circumscription observe that every variable is forced to lie in the range $[0, \|c\|_\infty]$ and hence the polytope is contained in the definable sphere $\mathrm{S}(0, \|c\|_\infty \cdot |V|)$. To see that the polytope has full dimension, consider the flow $f(u, v) := \frac{3}{4}\frac{c_{\min}}{(2|V|)^{d(s,u)}}$ for all vertices u, v with $c(u, v) > 0$, where $c_{\min}$ is the minimum non-zero capacity of G and $d(s, u)$ is the minimum distance from s to u. It can be observed that the flow on each edge with non-zero capacity can be independently changed by a small amount, implying that the polytope has full dimension. Formalising this argument proves the following.

Theorem 12. *There is an* FPC *interpretation $\Phi(s, t)$ of τ_{mat} in τ_{mat} which takes a τ_{mat}-structure coding a capacitated graph G to a τ_{mat}-structure coding a maximum (s,t)-flow of G.*

As the interpretation Φ defines a particular flow, the flow must, in some sense, be canonical because it is produced without making any choices. Informally, it is a convex combination of maximum flows resulting from the consideration of all orderings of the vertices. This is possible because of a convex combination of maximum (s,t)-flows is also a maximum (s,t)-flow. However, the combinatorial objects—cuts and matchings—central to our remaining applications are not closed under convex combinations. In the former case it is still possible to define the notion of a canonical optimum. In the latter case it is easy to observe, as noted in the introduction, that defining a canonical maximum matching is not possible.

V. Application: Minimum Cut

An *(s,t)-cut*of a capacitated graph $G = (V, c)$ is a subset C of the vertices V which contains t but not s. The *value* $\mathrm{val}(C)$ of the cut C is the sum of the capacity of edges going from vertices in $V\backslash C$ to vertices in C. A *minimum (s,t)-cut of G* is a cut whose value is the minimum over all (s,t)-cuts. A *minimum cut of G* is a minimum (s,t)-cut over all choices of distinct vertices s, t. Let f be a maximum (s,t)-flow in G. We say a vertex v is *reachable* from a vertex u in the *residual graph* $G|_f$ if any additional amount of flow can be pushed from u to v in the presence of the flow f.

Define $C_f := \{v \in V \mid t$ reachable from v in $G|_f\}$. By the max-flow/min-cut theorem, C_f is a minimum (s,t)-cut in G. Given the FPC interpretation Φ from Theorem 12 expressing an (s,t)-flow f, it is not difficult to construct a formula of FPC which defines the set of vertices C_f.

Theorem 13. *There is a formula $\xi(x,s,t)$ of* FPC *which given a τ_{mat}-structure coding a capacitated graph $G = (V,c)$, defines the vertices in a minimum (s,t)-cut of G.*

In fact, the cut C_f does not depend on f at all and is the smallest minimum (s,t)-cut in the sense that it is contained in all other minimum (s,t)-cuts of G.

Lemma 14. *Let $G = (V,c)$ be a capacitated graph with distinct vertices $s,t \in V$. Then the cut C_f is independent of the choice of a maximum (s,t)-flow f of G. Moreover, C_f is the intersection of all minimum (s,t)-cuts of G.*

Lemma 14 implies that the FPC formula ξ of Theorem 13 defines a *unique* (s,t)-cut of the graph G and for this reason we call it the *canonical* minimum (s,t)-cut of G: $K_{G,s,t}$.

VI. APPLICATION: MINIMUM ODD CUT

The *minimum odd cut* problem is closely related to the minimum (s,t)-cut problem. Here the goal is to define a minimum odd cut of a graph G. That is, a cut of odd size whose value is minimum among all odd size cuts of G. A capacitated graph $G = (V,c)$ is *symmetric* if for all $u,v \in V$, $c(u,v) = c(v,u)$. In this section we prove that in each symmetric graph G there is at least one pair of vertices s,t such that the canonical minimum (s,t)-cut $K_{G,s,t}$ is a minimum odd cut of G.

Before continuing we must define several special types of cuts. We extend a capacitated graph $G = (V,c)$ to a *marked* capacitated graph $G' = (V,c,M)$ with a marking $M \subseteq V$. We call a vertex $v \in V$ *marked* if $v \in M$. A cut C of a marked graph G is said to be a *marked cut*, if both C and $\overline{C}$ contain a marked vertex. A cut C of a graph $G = (V,c)$ with $|V|$ even is said to be an *odd* cut if $|C|$ is odd. A marked cut C of a marked graph $G = (V,c,M)$ with $|M|$ even is said to be an *odd* marked cut if $|C \cap M|$ is odd (note that this corresponds to the simpler notion when $M = V$). For any set $\mathcal{C}$ of cuts we define the *basic* cuts in $\mathcal{C}$ to be $\{C \in \mathcal{C} \mid \forall C' \in \mathcal{C}, C = C'$ or $C \not\supseteq C'\}$. Note that if $\mathcal{C}$ is non-empty it must contain at least one basic cut. When $\mathcal{C}$ is the set of minimum (s,t)-cuts, the formula ξ of Theorem 13 defines the unique basic cut in $\mathcal{C}$. We frequently describe sets of cuts by a sequence of the above adjectives and determine meaning by first evaluating the adjective which appear closest to the word "cut". The most complex cuts we consider are "basic minimum odd marked cuts".

We require two technical properties involving the intersections of marked cuts. The first says that basic minimum marked cuts do not have complicated intersections with basic minimum (s,t)-cuts.

Lemma 15. *Let $G = (V,c,M)$ be a marked symmetric graph. Let $s,t \in M$ be distinct vertices and let C be a basic minimum (s,t)-cut of G. For every basic minimum marked cut C' of G one of the following holds: (i) $C \supseteq C'$, (ii) $C \cap C' = \emptyset$ or (iii) $\{s,t\} \cap C' \neq \emptyset$.*

The second lemma says that given a basic minimum odd marked cut C, there exists a minimum marked cut C' which does not have a complicated intersection with C.

Lemma 16. *Let $G = (V,c,M)$ be a marked symmetric graph with $|M|$ even. Let C be a basic minimum odd marked cut of G. There exists a minimum marked cut C' of G such that one of the following holds: (i) $C \supseteq C'$ or (ii) $C \cap C' = \emptyset$.*

The main result of this section is the following theorem.

Theorem 17. *Let $G = (V,c,M)$ be a marked symmetric graph with even $|M| > 0$. There exist $s,t \in M$ such that the canonical minimum (s,t)-cut of G is a minimum odd marked cut of G.*

The intuition for the proof is as follows. Let G be a symmetric graph with minimum odd cut C. Mark all vertices in G. We use Lemma 16 to locate a basic minimum marked cut D of G which C does not partition (that is, D is either contained in C or disjoint from C). If D is an odd cut, and because D is basic, a canonical minimum (s,t)-cut separating a vertex $t \in D$ from a vertex $s \notin D$ is also a minimum odd cut of G and we are done. Otherwise $|D|$ is even and we form a new graph G' by collapsing D into a new super-vertex z, and then setting the effected capacities so that value of cuts which do not partition D are unchanged. Since C does not partition D and $|D|$ is even, the collapsed version C' of C is a minimum odd marked cut of G'. We repeat this collapsing procedure maintaining a graph G' and minimum odd marked cut C' until we locate a minimum marked cut D of G' that is also minimum odd marked cut. This provides marked vertices s,t such that the canonical minimum (s,t)-cut of G' is a minimum odd cut of G'. The proof concludes using Lemma 15 to translate this fact back to the original graph G and in doing so argues that the canonical minimum (s,t)-cut of G is a minimum odd cut.

We now formalise the notion of collapsing a graph. We begin by establishing notation for substituting sets into sets. Let $C,D \subseteq V$ and $z \notin V$ such that either $C \supseteq D$ or $C \cap D = \emptyset$. Define $C(z/D)$ to be a subset of $V' := (V \backslash D) \cup \{z\}$

$$C(z/D) := \begin{cases} (C \backslash D) \cup \{z\}, & C \supseteq D, \\ C, & C \cap D = \emptyset, \end{cases}$$

and for $C' \subseteq V'$ define $C'(D/z)$ to be a subset of V

$$C'(D/z) := \begin{cases} (C' \backslash \{z\}) \cup D, & z \in C', \\ C', & z \notin C'. \end{cases}$$

Observe that $(C(z/D))(D/z) = C$.

The subroutine COLLAPSE(G,D,z) in Fig. 2 describes a method of collapsing the vertex set $D \subseteq V$ in the marked graph $G = (V,c,M)$ to a single new super-vertex z. This subroutine is designed to preserve a basic minimum odd marked cut C with respect to a well-chosen marked cut D.

Fig. 2. A subroutine to collapse a set of vertices to a new single vertex.

COLLAPSE(G, D, z)

Input: A marked capacitated graph $G = (V, c, M)$, a set $D \subseteq V$ and $z \notin V$.
Output: The graph obtained from G by collapsing of the vertices in D to z.

$V' \leftarrow V(z/D)$.

$$c'(u,w) \leftarrow \begin{cases} c(u,w), & u, w \in V \backslash D, \\ \sum_{x \in D} c(u,x), & u \in V \backslash D, w = z, \\ \sum_{x \in D} c(x,w), & w \in V \backslash D, u = z. \end{cases}$$

$M' \leftarrow M \backslash D$.

return (V', c', M').

Fig. 3. An algorithm producing a witness for a minimum odd marked cut.

WITMINODDCUT(G, C)

Input: A symmetric graph $G = (V, c, M)$ and a minimum odd marked cut C of G.
Output: $(s,t) \in M^2$ such that $K_{G,s,t}$ is a minimum odd marked cut.

1: $G^0 := (V^0, c^0, M^0) \leftarrow (V, c, M)$.
2: $C^0 \leftarrow C$.
3: **for** $i \in \{0, 1, 2, \ldots\}$ **do**
4: Let D^i be a basic minimum marked cut of G^i such that $C^i \supseteq D^i$ or $C^i \cap D^i = \emptyset$.
5: **if** D^i is an odd marked cut of G^i **then**
6: **return** (s,t) with $t \in D^i \cap M^i$ and $s \in M^i \backslash D^i$.
7: $G^{i+1} \leftarrow$ COLLAPSE(G^i, D^i, z^i).
8: $C^{i+1} \leftarrow C^i(z^i/D^i)$.

Lemma 18. *Let $G = (V, c, M)$ be a marked symmetric graph with $|M|$ even. Let $z \notin V$ and $C, D \subseteq V$ be marked cuts of G such that $C \supseteq D$ or $C \cap D = \emptyset$. Define $C' := C(z/D)$ and $G' := (V', c', M') :=$ COLLAPSE(G, D, z).*

1) *The values of C in G and of C' in G' are identical.*
2) *If C is a basic minimum odd marked cut of G and $|D \cap M|$ is even, then C' is a basic minimum odd marked cut of G', and $|M'|$ is even, non-zero and $M' \subsetneq M$.*

With the key properties of COLLAPSE established we are ready to prove the main result of this section.

Proof of Theorem 17. Fix a basic minimum odd marked cut C of G. Figure 3 describes the algorithm WITMINODDCUT that computes vertices s and t witnessing the claim of the theorem from C by iteratively collapsing G. To prove the theorem it suffices to argue the algorithm halts and produces $(s,t) \in V^2$ such that the canonical minimum (s,t)-cut $K_{G,s,t}$ is a minimum odd marked cut.

As the algorithm runs it maintains the invariant that C^i is a basic minimum odd marked cut of G^i. Observe that this is initially true for $C^0 = C$ because $M^0 = M$ and C is a basic minimum odd marked cut of G. Suppose C^i is a basic minimum odd marked cut of G^i. The basic minimum marked cut D^i of G^i with $C^i \supseteq D^i$ or $C^i \cap D^i = \emptyset$ is guaranteed to exist by Lemma 16 (if the cut given by that lemma is not basic there must be a basic minimum marked cut strictly within it that continues to satisfy the intersection properties with C^i). If D^i is an odd marked cut, the algorithm halts at line 6. Otherwise the graph G^i and cut C^i are collapsed relative to D^i. The second part of Lemma 18 implies that C^{i+1} is a basic minimum odd marked cut of G^{i+1}. Thus the invariant holds.

Lemma 18 also implies that as the algorithm runs, $|M^i|$ is even and $M^{i+1} \subsetneq M^i$. The invariant and $M^0 = V$ imply that the test in line 5 will be successful and cause the algorithm to halt within $\frac{|V|}{2}$ iterations. Because D^i is a marked cut of G^i, when line 6 is reached $D^i \cap M^i$ and $M^i \backslash D^i$ are non-empty disjoint sets. This means that distinct s and t exist and are returned by the algorithm. Let r be the value of i when the algorithm halts. Fix any $t \in D^r \cap M^r$ and $s \in M^r \backslash D^r$. We use the shorthand K^i to denote the canonical minimum (s,t)-cut $K_{G^i,s,t}$ for $0 \leq i \leq r$. It remains to argue that $K^0 = K_{G,s,t}$ is a minimum odd marked cut.

Since the cut D^r is a minimum marked cut of G^r and $s, t \in M^r$, D^r is also a minimum (s,t)-cut of G^r and it has the same value as the canonical minimum (s,t)-cut K^r. This implies that K^r is a minimum marked cut of G^r because $s, t \in M^r$. By Lemma 14, D^r contains K^r, but D^r is also basic, so we conclude that $D^r = K^r$.

The first property of Lemma 18 implies that $\mathrm{val}(C^i(z^i/D^i)) = \mathrm{val}(C^{i+1})$ for all $0 \leq i < r$, and hence that $\mathrm{val}(C^i) = \mathrm{val}(C)$ for all $0 \leq i \leq r$. Since K^r is a minimum marked cut of G^r and C^r is a marked cut of G^r, $\mathrm{val}(K^r) \leq \mathrm{val}(C^r) = \mathrm{val}(C)$. The first part of Lemma 18 also implies that $\mathrm{val}(K^{i+1}(D^i/z^i)) = \mathrm{val}(K^{i+1})$ for all $0 \leq i < r$. Since $K^{i+1}(D^i/z^i)$ is an (s,t)-cut of G^i, $\mathrm{val}(K^i) \leq \mathrm{val}(K^{i+1}(D^i/z^i))$ for all $0 \leq i < r$. Hence we conclude that $\mathrm{val}(K^0) \leq \mathrm{val}(K^r) \leq \mathrm{val}(C)$.

It remains to argue that K^0 is an odd marked cut. Since K^r is an odd marked cut, it suffices to show that K^i is an odd marked cut if K^{i+1} is an odd marked cut, for all $0 \leq i < r$. To this end assume that K^{i+1} is an odd marked cut. Apply Lemma 15 with G^i, K^i and D^i; we note that (i) $K^i \supseteq D^i$, (ii) $K^i \cap D^i = \emptyset$, or (iii) $\{s,t\} \cap D^i \neq \emptyset$. Property (iii) cannot hold because s and t are selected after D^i was collapsed. This means that K^i either contains all of D^i or is disjoint from D^i. Because the K^i and K^{i+1} are canonical (s,t)-cuts,

$$K^i \subseteq K^{i+1}(D^i/z^i) \subseteq (K^i(z^i/D^i))(D^i/z^i) = K^i.$$

As the algorithm did not halt at step i, $|D^i \cap M^i|$ is even and thus K^i is an odd marked (s,t)-cut.

We conclude that K^0 is an odd marked cut with value at most that of a minimum odd marked cut C of G. Therefore $K^0 = K_{G,s,t}$ is a minimum odd marked cut. □

Consider the following procedure for locating a set of minimum odd marked cuts in a marked symmetric graph $G = (V, c, M)$: For all distinct $s, t \in M$ compute the canonical minimum (s,t)-cut $K_{G,s,t}$, eliminate those cuts which are not odd, then eliminate those cuts which are not minimal. Theorem 17 indicates that some cuts remain and that those cuts are minimum odd cuts of G. Note that the algorithm WITMINODDCUT in the proof of Theorem 17 is used only in the analysis and not actually run during the above procedure. This simple algorithm for defining a non-empty set of minimum odd cuts is critical to expressing the separation problem for the matching polytope in FPC.

VII. APPLICATION: MAXIMUM MATCHING

Let $G = (V, E)$ be an undirected graph. A *matching* $M \subseteq E$ is defined by the property that no two edges in M are incident to the same vertex. A matching M is *maximum* if no matchings with size larger than M exist. A maximum matching is *perfect* if every vertex in G is incident to some edge in the matching (i.e., $|M| = \frac{|V|}{2}$).

A. Maximum Matching Program

Maximum matching has an elegant representation as a linear program. In fact, it is an instance of a slightly more general problem: b-matching. Let $c \in \mathbb{Q}^E_{\geq 0}$, $b \in \mathbb{N}^V$ and $A \in \{0,1\}^{V \times E}$ be the incidence matrix of the undirected graph $G = (V, E)$: the columns of A correspond to the edges E and the rows to the vertices V, and $A_{ve} = 1$ if edge e is incident on vertex v. Alternatively we view edges $e \in E$ as two-element subsets of V. The goal of the b-matching problem is to determine an optimum of the following *integer* program

$$\max\ c^\top y \quad \text{subject to} \quad Ay \leq b \ \text{ and } \ y \geq 0^E. \tag{2}$$

We obtain the usual maximum matching problem in the special case where $b = 1^V$ and $c = 1^E$.

Generically, integer programming is NP-complete, so instead of trying to directly solve the above program we consider the following relaxation as a rational linear program

$$\begin{aligned} &\max\ c^\top y \quad \text{subject to} \quad Ay \leq b,\ \ y \geq 0^E,\ \text{and} \\ &\qquad y(W) \leq \frac{1}{2}(b(W) - 1),\ \ \forall W \subseteq V \text{ with } b(W) \text{ odd,} \end{aligned} \tag{3}$$

where $y(W) := \sum_{e \in E, e \subseteq W} y_e$ and $b(W) := \sum_{v \in W} b_v$. Here we have added a new set of constraints over subsets of the vertices. The integral points which satisfy (2), also satisfy the additional constraints that are added in (3). To see this, let y be a feasible integral solution, consider some set W with $b(W)$ odd. If $|W| = 1$, then $y(W) = 0$ because no edges have both endpoints in W, so assume $|W| \geq 2$. It follows that $2y(W) \leq b(W)$, by summing the constraints of $Ay \leq b$ over W with respect to only the edges with *both* endpoints in W. Since $b(W)$ is odd, $\frac{1}{2}b(W)$ is half integral, but $y(W)$ is integral because y is an integral solution; this means the constraint $y(W) \leq \frac{1}{2}(b(W) - 1)$ is a valid constraint for all integral solutions. In fact [22] shows something stronger.

Lemma 19 ([22, Theorem P])**.** *The extremal points of the linear program* (3) *are integral and are the extremal solutions to the b-matching problem.*

Thus to solve b-matching it suffices to solve the relaxed linear program (3). As mentioned before, it will not be possible to show that FPC can generally define a particular maximum matching, there can be simply too many. However, the above lemma means that the existence of a (likely non-integral) feasible point y of (3) with value $c^\top y$ witnesses the existence of a maximum b-matching with value at least $c^\top y$. In addition, the number of constraints in this linear program is exponential in the size of the graph G. Thus, we cannot hope to interpret this linear program directly in G, using FPC. Rather what we can show is that there is an FPC interpretation which, given G, b and c, expresses the separation problem for the b-matching polytope in the linear program (3). Combining this with the results of Section III gives an FPC interpretation expressing the b-matching optimum.

B. Expressing Maximum Matching in FPC

The b-matching polytopes have a natural representation over $\tau_{\text{match}} := \tau_{\text{mat}} \uplus \tau_{\text{vec}}$. Although the number of constraints in the b-matching polytope may be large, the individual constraints have size at most a polynomial in the size of the matching instance. Thus this representation is well-described.

We now describe an FPC interpretation expressing the separation problem for the b-matching polytope given a τ_{match}-structure coding the matrix A and bound vector b. As in the explicit constraint setting, our approach is to come up with a definable set of violated constraints iff the candidate point is infeasible. We then define a canonical violated constraint by summing this definable violated set. Identifying violated vertex and edge constraints can easily be done in FPC as before. However, it is not immediately clear how to do this for the odd set constraints.

To overcome this hurdle we follow the approach of [12]. Let y be point which we wish to separate from the matching polyhedron. Define $s := b - Ay$ to be the slack in the constraints $Ay \leq b$. Analogous to $b(W)$, define $s(W) := \sum_{v \in W} s_v$. Observe that $2y(W) + y(W : V \backslash W) + s(W) = b(W)$ (here $y(W : V \backslash W)$ is sum of edge variables with one endpoint in W and one in $V \backslash W$). This translates the constraints $y(W) \leq \frac{1}{2}(b(W) - 1)$ exactly to $y(W : V \backslash W) + s(W) \geq 1$. This means to find a violated constraint of this type it suffices to find W such that $y(W : V \backslash W) + s(W) < 1$.

Define a marked symmetric graph H over vertex set $U := V \cup \{z\}$ where z is a new vertex. Let H have symmetric capacity d: $d(u, v) := y_e$ when $u, v \in V$ and $u, v \in e$, and $d(u, v) := s_v$ when $u = z$ and $v \in V$. Let $M := \{v \in V \mid b_v \text{ is odd}\}$. If $|M|$ is odd, add z to M. Thus we have a marked symmetric graph $H = (U, d, M)$. Consider any odd marked cut W of H, without loss of generality $z \notin W$ (otherwise, take the complement). Observe that the value of edges crossing the cut is exactly $y(W : V \backslash W) + s(W)$; also note that $s(W)$ is odd. Thus there is a minimum odd marked cut W of H with value less than 1 iff there is a violated odd set constraint in (3).

By Theorem 17, there is a violated odd set constraint iff for some $s, t \in M$ the canonical minimum (s, t)-cut is an minimum odd marked cut with value less than 1. We conclude, using Theorem 13 and Lemma 14, that we can define a family of violated odd set constraints within FPC. Summing these defined violated constraints produces a canonical violated constraint which must be non-trivial by Proposition 8. Thus, as in Theorem 9 there is an FPC interpretation expressing the separation problem for the polytope in the linear program (3).

Lemma 20. *There is an* FPC *interpretation of τ_{vec} in $\tau_{match} \uplus \tau_{vec}$ expressing the separation problem for the b-matching polytopes with respect to their natural representation as τ_{match}-structures.*

Similar to the proof of Theorem 12 one can show that the b-matching polytope has full dimension and can be circumscribed within FPC with respect to its natural well-described representation. Combining this with Lemma 20 and Theorem 10, we conclude that there is an FPC interpretation expressing the value of the maximum b-matching of a graph.

Theorem 21. *There is an* FPC *interpretation of $\tau_{\mathbb{Q}}$ in $\tau_{match} \uplus \tau_{vec}$ which takes a $\tau_{match} \uplus \tau_{vec}$-structure coding a b-matching polytope P and a vector c to a rational number m indicating the value of the maximum b-matching of P with respect to c.*

VIII. Conclusion

Our main result is that the linear programming problem on full-dimensional polyhedrons can be expressed in fixed-point logic with counting—indeed, that the linear optimisation problem can be expressed in FPC for any class of polytopes for which the separation problem can be defined in FPC. As a consequence, we solve an open problem of [1] concluding that there is a formula of fixed-point logic with counting which defines the size of a maximum b-matching in a graph. There are a number of natural research directions to consider.

a) Convex programming: A polytope is an instance of a much more general geometric object: a convex set. The robust nature of the ellipsoid method means it has been extended to help solve more general optimisation problems, e.g., semi-definite programs and quadratic programs. It seems likely that our methods can be extended to these settings.

b) Completeness: Linear programming is complete for polynomial time under logspace reductions [23]. It follows from our results that it cannot be complete for P under logical reductions such as first-order interpretations, since this would imply that P is contained in FPC. Could it still be the case that linear programming is complete for FPC under such weak reductions? Or perhaps FOC reductions? Even if linear programming is not complete, there may be other interesting combinatorial problems that can be expressed in FPC via reduction to linear programming.

c) LP hierarchies and integrality gaps: Another intriguing connection between counting logics and linear programming is established in [24], [25] where it is shown that the hierarchy of Sherali-Adams relaxations [26] of the graph isomorphism integer program interleaves with equivalence in k-variable logic with counting (C^k). It is suggested [24] that inexpressibility results for C^k could be used to derive integrality gaps for such relaxations. It is a consequence of the results in this paper that the Sherali-Adams approximations of not only isomorphism, but of other combinatorial problems can be expressed in FPC. Do our results provide another route to using inexpressibility in FPC to prove integrality gaps?

Acknowledgments

The authors would like to thank Siddharth Barman for his helpful comments on an early draft of this paper and the anonymous reviewers for their constructive suggestions. Research supported by EPSRC grant EP/H026835.

References

[1] A. Blass, Y. Gurevich, and S. Shelah, "Choiceless polynomial time," *Ann. Pure Appl. Logic*, vol. 100, pp. 141–187, 1999.
[2] A. Chandra and D. Harel, "Structure and complexity of relational queries," *J. Comput. Syst. Sci.*, vol. 25, no. 1, pp. 99–128, 1982.
[3] J.-Y. Cai, M. Fürer, and N. Immerman, "An optimal lower bound on the number of variables for graph identification," *Combinatorica*, vol. 12, no. 4, pp. 389–410, 1992.
[4] A. Dawar, M. Grohe, B. Holm, and B. Laubner, "Logics with rank operators," in *LICS*. IEEE, 2009, pp. 113–122.
[5] A. Blass, Y. Gurevich, and S. Shelah, "On polynomial time computation over unordered structures," *J. Symbolic Logic*, pp. 1093–1125, 2002.
[6] M. Grohe, "Fixed-point definability and polynomial time on graph with excluded minors," in *LICS*. IEEE, 2010, pp. 179–188.
[7] A. Atserias, A. Bulatov, and A. Dawar, "Affine systems of equations and counting infinitary logic," *Theor. Comput. Sci.*, vol. 410, no. 18, pp. 1666–1683, 2009.
[8] A. Blass and Y. Gurevich, "A quick update on open problems in Blass-Gurevich-Shelah's article 'On polynomial time computations over unordered structures'," Online at http://research.microsoft.com/ gurevich/annotated.html, 2005, [Accessed July 19, 2010].
[9] B. Rossman, "Choiceless computation and symmetry," in *Fields of Logic and Computation*. Springer, 2010, pp. 565–580.
[10] L. Khachiyan, "Polynomial algorithms in linear programming," *USSR Comp. Math. Math*, vol. 20, no. 1, pp. 53–72, 1980.
[11] M. Grötschel, L. Lovász, and A. Schrijver, *Geometric algorithms and combinatorial optimization*. Springer-Verlag Berlin / New York, 1988.
[12] M. Padberg and M. Rao, "Odd minimum cut-sets and b-matchings," *Math. Oper. Res.*, vol. 7, no. 1, pp. 67–80, 1982.
[13] M. Anderson, A. Dawar, and B. Holm, "Maximum matching and linear programming in fixed-point logic with counting," *arXiv*, 2013. [Online]. Available: http://arxiv.org/abs/1304.6870
[14] H. Ebbinghaus and J. Flum, *Finite Model Theory*. Springer, 1999.
[15] L. Libkin, *Elements of Finite Model Theory*. Springer, 2004.
[16] N. Immerman, *Descriptive Complexity*. Springer-Verlag, 1999.
[17] B. Holm, "Descriptive complexity of linear algebra," Ph.D. dissertation, University of Cambridge, 2010.
[18] G. Dantzig, *Linear programming and extensions*. Princeton University Press, 1963, (most recent edition published in 1998).
[19] D. Spielman and S.-H. Teng, "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time," *JACM*, vol. 51, no. 3, pp. 385–463, 2004.
[20] N. Shor, "Cut-off method with space extension in convex programming problems," *Cybern. Syst. Anal.*, vol. 13, no. 1, pp. 94–96, 1977.
[21] M. Grötschel, L. Lovász, and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization," *Combinatorica*, vol. 1, pp. 169–197, 1981.
[22] J. Edmonds, "Maximum matching and a polyhedron with $0,1$ vertices," *J. Res. Nat. Bur. Stand.*, vol. 69 B, pp. 125–130, 1965.
[23] D. Dobkin, R. Lipton, and S. Reiss, "Linear programming is log-space hard for *P*," *Inform. Process. Lett.*, vol. 8, no. 2, pp. 96–97, 1979.
[24] A. Atserias and E. Maneva, "Sherali-Adams relaxations and indistinguishability in counting logics," in *ITCS*. ACM, 2012, pp. 367–379.
[25] M. Grohe and M. Otto, "Pebble Games and Linear Equations," in *CSL*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2012, pp. 289–304.

[26] H. Sherali and W. Adams, “A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems,” *SIAM Journal on Discrete Mathematics*, vol. 3, no. 3, pp. 411–430, 1990.