Solving Linear Programs without Breaking Abstractions

MATTHEW ANDERSON, ANUJ DAWAR, and BJARKI HOLM, University of Cambridge

We show that the ellipsoid method for solving linear programs can be implemented in a way that respects the symmetry of the program being solved. That is to say, there is an algorithmic implementation of the method that does not distinguish, or make choices, between variables or constraints in the program unless they are distinguished by properties definable from the program. In particular, we demonstrate that the solvability of linear programs can be expressed in fixed-point logic with counting (FPC) as long as the program is given by a separation oracle that is itself definable in FPC. We use this to 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 [Blass et al. 1999]. On the way to defining a suitable separation oracle for the maximum matching program, we provide FPC formulas defining canonical maximum flows and minimum cuts in undirected capacitated graphs.

Categories and Subject Descriptors: F.2.2 [**COMPUTATION BY ABSTRACT DEVICES**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures*; F.4.1 [**MATHEMATICAL LOGIC AND FORMAL LANGUAGES**]: Mathematical Logic—*Computational logic*; G.1.6 [**NUMERICAL ANALYSIS**]: Optimization—*Linear programming*

General Terms: Algorithms; Theory;

Additional Key Words and Phrases: linear programming, maximum matching, ellipsoid method, fixed-point logic with counting, choiceless polynomial time

ACM Reference Format:
Matthew Anderson, Anuj Dawar, and Bjarki Holm, 2014. Solving Linear Programs without Breaking Abstractions *J. ACM* V, N, Article A (January YYYY), 27 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Many fundamental algorithmic problems concern combinatorial structures, like graphs. Such structures provide the right level of abstraction for formulating a large variety of real-world problems, while hiding unnecessary detail and revealing the essential computational challenges. In practice, algorithms solving problems on combinatorial structures often violate the level of abstraction with which the problem is described. For instance, an algorithm for a graph problem may assume that the set of vertices is indexed by natural numbers and may use this indexing to process the vertices in a particular order. This violates the principle of abstraction, but the reason for doing so is clear: it appears to permit more efficient algorithms.

Research supported by EPSRC grant EP/H026835. An extended abstract of this paper appeared in the proceedings of LICS [Anderson et al. 2013].
Authors' addresses: M. Anderson, (Current address) Computer Science Department, Union College; A. Dawar, Computer Laboratory, University of Cambridge; and Bjarki Holm, (Current address) Twigkit Ltd., Cambridge, UK.

DOI: http://dx.doi.org/10.1145/0000000.0000000

Maximum matching is a consequential graph problem that exemplifies this phenomenon. Given a graph $G = (V, E)$ the goal of the problem is to determine the maximum size of a set of edges $M \subseteq E$ so that no two edges in M are incident on the same vertex. This problem is an abstraction of a wide range of practical problems arising in areas such as scheduling, game theory, molecular biology and computer vision, and algorithms for solving it have been extensively studied (c.f., e.g., [Grötschel et al. 1988; Duan and Pettie 2014]).

One of the fastest algorithms solving maximum matching is the $O(\sqrt{|V|}|E|)$ algorithm of [Micali and Vazirani 1980]. In order to prove the algorithm is correct Vazirani develops a number of structural results about graphs, but then defines the notion of an *anomaly* and notes that it is an "algorithmic convenience" that does not appear in the structure theory [Vazirani 1994, p.97]. Indeed, it is instructive to verify that the property of being an anomaly is not a property of the graph G, i.e., whether or not an edge in the graph is an anomaly is determined, not by the graph G alone, but by the particular choices made by the algorithm in constructing a matching up to that point. It does not seem possible to describe the Micali-Vazirani algorithm at the level of abstraction where the only structure on the vertices is given by the graph's edges.

Can an efficient algorithm for maximum matching respect the abstraction?

More generally, is it possible to give such an algorithm for *any* graph problem that admits an efficient algorithmic solution? Such questions motivated Blass, Gurevich and Shelah to introduce the model of *choiceless polynomial time with counting* CPTC, a model of computation aimed at formalising polynomial-time algorithms that cannot make arbitrary choices [Blass et al. 1999]. They showed in [Blass et al. 2002] that the problem of determining whether a bipartite graph has a perfect matching is expressible in CPTC and posed as an open question whether the existence of a perfect matching on general graphs can be defined in this formalism. Indeed, this question first appears in [Blass et al. 1999] (also see [Blass and Gurevich 2005; Rossman 2010]) where it is stated that it seems "unlikely" that this problem can be decided in CPTC. As one main contribution of this paper we settle this question by showing that the size of a maximum matching can be defined even in the weaker formalism of FPC, the extension of inflationary fixed-point logic by counting terms.

Fixed-point logic with counting originated in the field of descriptive complexity theory—whose central question generalises those considered above:

Is there a logic that characterises those problems solvable in polynomial time?

This question first appeared in the work of [Chandra and Harel 1982] in the context of database query languages. At one time it was conjectured that FPC would suffice to express all polynomial-time properties, but this was refuted by Cai, Fürer and Immerman [Cai et al. 1992]. Since then, a number of logics have been proposed whose expressive power is strictly greater than that of FPC but still contained within P, the class of problems decidable by a polynomial-time Turing machine. Among these are FPR, fixed-point logic with rank operators [Dawar et al. 2009], and CPTC, mentioned above. 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, it still forms a natural class within P. For instance, FPC can express all polynomial-time properties on many graph classes, such as any class of proper minor-closed graphs [Grohe 2012]. Reinforcing the view that FPC represents a natural level of expressiveness inside P is the recent result of [Anderson and Dawar 2014] showing that a graph property is definable in FPC if, and only if, it is decidable by a uniform

polynomial-size family of symmetric threshold circuits. Here, a circuit C that takes as input the $\binom{n}{2}$ potential edges of an n vertex graph is said to be *symmetric* if any permutation of the n vertices extends to an automorphism of C. This symmetry condition seems a natural way to express that the computation encoded by C does not depend on arbitrary choices.

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. 1992] and the subsequent work of [Atserias et al. 2009] 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.

Linear programming is a fundamental problem in geometric optimisation with deep connections to combinatorial optimisation, including the maximum matching problem, and, moreover, is only known to be efficiently solvable via involved algorithmic techniques (e.g., [Khachiyan 1980; Karmarkar 1984]). As our main technical result we show that there is a formula of FPC expressing linear programming, i.e., it defines from a polytope a point inside the polytope maximising a given linear objective function, if such a point exists. Here, a *polytope* is a convex set in Euclidean space given by finite intersections of linear inequalities (or *constraints*) over a set of variables, suitably represented as a relational structure without assuming an ordering on the sets of variables or constraints.

In particular, we consider representations where, as in many applications of linear programming, the set of constraints is not given explicitly (indeed, the set 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*, i.e., it is within the polytope, and, if it is not, returns a constraint that is violated by x. It is well known that Khachiyan's polynomial-time algorithm for linear programming—the *ellipsoid method*—can be extended to prove that the linear programming and separation problems are polynomial-time equivalent (c.f., [Khachiyan 1980; Grötschel et al. 1981; Grötschel et al. 1988]). Our main technical result is an analog for FPC.

THEOREM 1.1 (INFORMAL, SEE THEOREM 4.2). *If a separation oracle for a class of polytopes is expressible in* FPC*, then linear programming on that class is expressible in* FPC.

Our approach is inspired by the following observation. Although the set of variables is not inherently ordered, the separation oracle induces a natural equivalence relation on these variables whereby two variables are equivalent if they are not distinguished in any invocation of the oracle. Given a linear ordering on the induced equivalence classes, we can define in FPC a reduction of the optimisation problem to an instance with an ordered set of variables by taking the quotient of the polytope under the equivalence relation. We show that solving the optimisation problem on this quotient polytope—now using the classical polynomial-time reduction made possible by the ordering—allows us to recover a solution to the original problem. In practice, neither the equivalence classes nor the order are given beforehand; rather they are iteratively refined via the invocations of the separation oracle made while optimising over the quotient polytope. The details of this argument are presented in Section 4.

Thus to express the solution to a linear program in FPC it suffices to express a separation oracle in FPC. A key obstacle to expressing separation oracles in FPC is

that a particular violated constraint must be chosen. We show, in Section 3, that when the constraints are given explicitly, a *canonical* violated constraint can be defined by taking the sum of all the violated constraints. This implies that the class of feasible linear programs, when explicitly given, can be expressed in FPC. When the constraints are not given explicitly, it may still be possible to express canonical violated constraints (and hence separation oracles) using domain knowledge.

In total our construction shows that the ellipsoid method can be implemented in a way that respects the symmetries of the problem it solves and yields new insight into this fundamental algorithmic technique. In some cases, this may even lead to more efficient uses of the method by allowing one to reduce the dimension of the polytope by factoring out its symmetries. In a related result, [Grohe et al. 2013] show how vertex colour-refinement techniques can be used to reduce the dimension of linear programs and in doing so heuristically improve running times of optimisation algorithms.

Applications of linear programming in combinatorial optimisation are myriad. We use the FPC-definability of linear programming as a tool to show that some of the problems that reduce to it are also FPC definable.

First, in Section 5.1, we use the definability of explicitly-given linear programs to show that a maximum flow in a capacitated graph $G = (V, c)$ is definable in FPC. Indeed, this follows rather directly from the first result, since the flow polytope is specified by a polynomial number, in $|V|$, of explicitly-given constraints, and hence a separation oracle for the flow polytope can be easily defined from G in FPC.

Next, in Section 5.2, we use the definability of maximum flows to show that minimum (s, t)-cuts are also definable in FPC. That is, in the vocabulary of capacitated graphs, there is a formula which defines a set of vertices $C_{s,t}$ corresponding to a minimum value cut separating the vertex s from the vertex t. The cut $C_{s,t}$ defined in this way is *canonical* in a strong sense: we show that it is the smallest (under set-inclusion) minimum cut separating s from t.

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 [Edmonds 1965]. We show that there is a separation oracle for the matching polytope 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 derived graph [Padberg and Rao 1982]. Indeed, there is always a pair of vertices s, t such that a canonical minimum (s, t)-cut is a minimum odd-size cut [Goemans and Ramakrishnan 1995]. This, combined with the definability of canonical minimum cuts, gives us the separation oracle for the matching polytope that we seek.

Combining this separation oracle with Theorem 1.1 yields that an optimum of the matching polytope can be expressed in FPC. However, such an optimum need only achieve the maximum objective value—it need not indicate a particular maximum matching. This is a consequence of FPC being necessarily invariant under automorphisms of G. To see this, consider $G = K_n$, the complete graph on n vertices; K_n contains an *exponential* number, in n, 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 to pick out a particular matching in this situation. What we can do, however, is define a formula that expresses the size of the maximum matching in a graph using an optimum point in the polytope. 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 matching are presented in Section 5.4.

2. 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, and $\langle z \rangle$ denotes $\lceil \log_2(|z|) + 1 \rceil$, i.e. the number of bits required to represent $|z|$. For a vector $v \in \mathbb{Q}^I$, $\|v\|_\infty := \max_{i \in I} |v_i|$ denotes its infinity norm.

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

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 variables: *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: $\mu \leq \eta$, 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 R is a relation symbol (either from the vocabulary τ or a variable with associated type $\sigma \in \{\mathrm{elem}, \mathrm{num}\}^*$); each t_i is an element variable or a constant symbol (if $\sigma_i = \mathrm{elem}$) or a number variable (if $\sigma_i = \mathrm{num}$). The set FPC$[\tau]$ of FPC *formulas* over τ is built up from the atomic formulas by applying an inflationary fixed-point operator which builds formulas $[\mathrm{ifp}_{R,\vec{x}}\phi](\vec{t})$ when ϕ is a formula, R is a relation variable of type σ, $\vec{x}$ is a σ-tuple of variables and $\vec{t}$ is a σ-tuple of terms; forming *counting terms* $\#_x\phi$, where ϕ is a formula and x an element variable; forming formulas of the form $s \leq t$ 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 $\mathrm{dom}(\mathfrak{A})$ where $n := |\mathrm{dom}(\mathfrak{A})|$. The semantics of atomic formulas, fixed-points and first-order operations are defined as usual (c.f., e.g., [Ebbinghaus and Flum 1999] for details), with comparison of number terms $\mu \leq \eta$ interpreted by comparing the corresponding integers in $[n+1]$. Finally, for a counting term of the form $\#_x\phi$, where ϕ is a formula and x an element variable, the intended semantics is that it denotes the number (i.e., the element of $[n+1]$) of elements satisfying the formula ϕ.

In general, a formula $\phi(\vec{x}, \vec{\mu})$ of FPC defines a relation over $\mathrm{dom}(\mathfrak{A}) \uplus [n+1]$ that is invariant under automorphisms of $\mathfrak{A}$. For a more detailed definition of FPC, we refer the reader to [Ebbinghaus and Flum 1999; Libkin 2004].

It is known, by the results of Immerman and Vardi [Immerman 1986; Vardi 1982], that every polynomial-time decidable property of *ordered* structures is definable in fixed-point logic, and therefore also in FPC. Here, an ordered structure is one which includes a binary relation which is a linear order of its domain. For reference, we state this result as follows.

THEOREM 2.1 (IMMERMAN-VARDI THEOREM). *A property of finite ordered structures is decidable in polynomial time if, and only if, that property can be defined by a formula of* FPC.

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 $\phi_R(\vec{x}_1,\ldots,\vec{x}_k)$; 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 $\mathfrak{B}$ to a σ-structure $\mathfrak{A}$ if there is a surjective map h from the m-tuples $\{\vec{a} \in (\mathrm{dom}(\mathfrak{A}) \uplus [n+1])^m \mid \mathfrak{A} \models \delta[\vec{a}]\}$ to $\mathfrak{B}$ such that:

(1) $h(\vec{a}_1) = h(\vec{a}_2)$ if, and only if, $\mathfrak{A} \models \varepsilon[\vec{a}_1,\vec{a}_2]$;
(2) $R^{\mathfrak{B}}(h(\vec{a}_1),\ldots,h(\vec{a}_k))$ if, and only if, $\mathfrak{A} \models \phi_R[\vec{a}_1,\ldots,\vec{a}_k]$; and
(3) $h(\vec{a}) = c^{\mathfrak{B}}$ if, and only if, $\mathfrak{A} \models \gamma_c[\vec{a}]$.

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

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 $\mathfrak{A} \models \phi'$ if, and only if, $\Theta(\mathfrak{A}) \models \phi$ (see [Immerman 1999, Sec. 3.2]) In particular, if $\Theta(\mathfrak{A})$ is an ordered structure, for all $\mathfrak{A}$, then by the Immerman-Vardi theorem above, for any polynomial-time decidable class C, there is an FPC formula ϕ such that $\mathfrak{A} \models \phi$ if, and only if, $\Theta(\mathfrak{A}) \in C$.

2.2. Numbers, Vectors and Matrices

Let z and b be integers with $b \geq \langle z \rangle$, $B = [b]$ and write $\mathrm{bit}(x,k)$ to denote the k-th (starting with $k = 0$) 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 $\mathfrak{B}$ 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 $S^{\mathfrak{B}} = \emptyset$ and -1 otherwise, and the unary relation X is interpreted as $X^{\mathfrak{B}} = \{k \in B \mid \mathrm{bit}(x,k) = 1\}$. That is $k \in X^{\mathfrak{B}}$ 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 $B = [b]$ (where $b \geq \langle x \rangle, \langle d \rangle$) 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 $D^{\mathfrak{B}} \neq \emptyset$.

We now generalise these notions and consider unordered tensors over the rationals (the case of integers is completely analogous). Let $J_1,\ldots,J_r$ be a family of finite non-empty sets. An unordered tensor T of order r over $\mathbb{Q}$ is a function $T : J_1 \times \cdots \times J_r \to \mathbb{Q}$. We write $t_{j_1\ldots j_r} = s_{j_1\ldots j_r} \frac{x_{j_1\ldots j_r}}{d_{j_1\ldots j_r}}$ to denote the element of T indexed by $(j_1,\ldots,j_r) \in J_1 \times \cdots \times J_r$. 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 $b \geq \langle m \rangle$ and

$B = [b]$. The tensor T is then an $(r+1)$-sorted structure $\mathfrak{T}$ with r index sorts $J_1, \ldots, J_r$ 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 $s_{j_1 \ldots j_r} \in \{-1, 1\}$ as before, the $(r+1)$-ary relation X is interpreted as

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

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 [Holm 2010] it is shown that a variety of basic linear-algebraic operations on unordered vectors and matrices can be expressed in fixed-point logic with counting. We gather some of these results in the following proposition, for later reference.

PROPOSITION 2.2 (OPERATIONS ON VECTORS AND MATRICES). *The following operations on rational vectors and matrices described as finite structures in the signatures τ_{vec} and τ_{mat}, respectively, can be expressed in* FPC.

— EQUALITY. *$A = B$ for matrices $A, B \in \mathbb{Q}^{I \times J}$ and $x = y$ for vectors $x, y \in \mathbb{Q}^I$.*
— POINTWISE ORDERING. *$x \leq y$ for vectors $x, y \in \mathbb{Q}^I$, where $x \leq y$ if, and only if, $x_i \leq y_i$ for all $i \in I$.*
— PRODUCT. *AB, Ax and $x^\top y$, for matrices $A \in \mathbb{Q}^{I \times J}$, $B \in \mathbb{Q}^{J \times K}$ and vectors $x, y \in \mathbb{Q}^J$.*
— BIG SUMS. *$\sum_{i \in I} x_i$ where I is a finite set and each $x_i \in \mathbb{Q}^J$ is a vector.*
— INFINITY NORM. *$\|x\|_\infty$ for a vector $x \in \mathbb{Q}^I$.*

2.3. Linear Programming

We recall some basic definitions from combinatorics and linear optimisation. For further background, see, for example, the textbook [Grötschel et al. 1988].

Polytopes. Consider the rational Euclidean space $\mathbb{Q}^V$ indexed by a set V. 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 $\{x \in \mathbb{Q}^V \mid a^\top x \leq \beta\}$ specified by the *constraint* $a^\top x \leq \beta$, where $a \in \mathbb{Q}^V$ and $\beta \in \mathbb{Q}$. A (rational) *polytope* is a convex set $P \subseteq \mathbb{Q}^V$ which is the intersection of a *finite* number of half-spaces.[1] 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 $P = P_{A,b} := \{x \in \mathbb{Q}^V \mid Ax \leq b\}$. We say two polytopes $P \subseteq \mathbb{Q}^V$ and $P' \subseteq \mathbb{Q}^{V'}$ are *isomorphic*, $P \cong P'$, if there is a bijection $h : V \to V'$ such that for any vector $a \in \mathbb{Q}^V$ with $a = (a_v)_{v \in V}$, $a \in P$ iff $(a_{h(v)})_{v \in V} \in P'$.

Polytopes have an alternative characterisation as a combination of convex hulls and cones. Let S be a finite set of vectors in $\mathbb{Q}^V$ and define the *convex hull* of S

$$\text{conv}(S) := \left\{ \sum_{s \in S} \lambda_s s \;\middle|\; \lambda_s \in \mathbb{Q}_{\geq 0}, \forall s \in S \text{ and } \sum_{s \in S} \lambda_s = 1 \right\},$$

and similarly define the *cone* of S

$$\text{cone}(S) := \left\{ \sum_{s \in S} \lambda_s s \;\middle|\; \lambda_s \in \mathbb{Q}_{\geq 0}, \forall s \in S \right\}.$$

If P is a polytope in $\mathbb{Q}^V$, then there exist finite sets $S_1, S_2 \subseteq \mathbb{Q}^V$ such that $P = P_{S_1,S_2} := \text{conv}(S_1) + \text{cone}(S_2) = \{x_1 + x_2 \mid x_1 \in \text{conv}(S_1), x_2 \in \text{cone}(S_2)\}$.

[1] In the literature of some fields the term "polyhedron" denotes what we call a "polytope", and the term "polytope" is used to denote a polyhedron that is the convex hull of a finite number of points.

The *size*, or bit complexity, of a vector $a \in \mathbb{Q}^V$ (denoted $\langle a \rangle$) is the number of bits required to encode the components of a in some standard encoding of rational numbers. Note that $\langle a \rangle$ is at least $|V|$. The size of a constraint $a^\top x \leq \beta$ is then $\langle a \rangle + \langle \beta \rangle$. If $Ax \leq b$ is a system of linear inequalities then 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 *facet complexity* $\langle P \rangle_{\mathrm{f}}$ of a polytope P is the minimum over the sizes of the systems $Ax \leq b$ such that $P = P_{A,b}$. When $P = \mathbb{Q}^V$, we take $\langle P \rangle_{\mathrm{f}} = |V| + 1$. The *vertex complexity* $\langle P \rangle_{\mathrm{v}}$ of a polytope P is the minimum over the maximum vector size in the union of sets $S_1, S_2 \subseteq \mathbb{Q}^V$ such that $P = \mathrm{conv}(S_1) + \mathrm{cone}(S_2)$. When $P = \emptyset \subseteq \mathbb{Q}^V$, we take $\langle P \rangle_{\mathrm{v}} = |V|$. The facet and vertex complexity of a polytope are polynomially related: $\langle P \rangle_{\mathrm{v}} \leq 4|V|^2 \langle P \rangle_{\mathrm{f}}$ and $\langle P \rangle_{\mathrm{f}} \leq 3|V|^2 \langle P \rangle_{\mathrm{v}}$ [Grötschel et al. 1988, Lemma 6.2.4]. As we are not concerned with polynomial factors, we simply write $\langle P \rangle$ for the complexity of P where either will do.

Problems on polytopes. We are interested in two fundamental combinatorial problems on polytopes: *linear optimisation* and *separation*.

Problem 2.3 (*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*. Over the years, a number of algorithms for solving linear programs have been studied. Early work by [Dantzig 1963] gave a combinatorial algorithm—the *simplex method*—which traverses the vertices (extremal points) of the polytope favouring vertices that improve the objective value. Although the simplex method is useful in practice, it tends not to be theoretically useful because strong worst-case performance guarantees are not known and, in some cases, known not to exist[2]. A series of works studying linear programming from a geometric perspective [Shor 1972; Yudin and Nemirovskii 1976; Shor 1977] culminated in the breakthrough of [Khachiyan 1980] which established a polynomial-time algorithm—the *ellipsoid method*—for solving linear programs. One of the strengths of the ellipsoid method is that it can be applied to linear programs where the constraints are not given explicitly. In such implicitly-defined linear programs, we are instead given a polynomial-time algorithm, known as a *separation oracle*, for solving the following separation problem.

Problem 2.4 (*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 (see Section 4).

2.4. Representing polytopes by relational structures

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.

[2]See [Klee and Minty 1972] for an example of an exponential worst-case time lower bound for the simplex method. Stronger upper bounds are known for the average-case and smoothed complexity of the simplex method [Spielman and Teng 2004].

Definition 2.5. 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, $\mathfrak{A} \cong \mathfrak{B}$ implies $\nu(\mathfrak{A}) \cong \nu(\mathfrak{B})$.

For concreteness, consider the vocabulary $\tau := \tau_{\mathrm{mat}} \uplus \tau_{\mathrm{vec}}$ obtained by taking the disjoint union of the vocabularies for rational matrices and vectors. A τ-structure over a domain 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 $\nu(\mathfrak{A})$ are polynomially bounded in the size of $\mathfrak{A}$. We will also be interested in representations ν where the number of constraints in $\nu(\mathfrak{A})$ is exponential in $|\mathfrak{A}|$, but we always confine ourselves to representations where the size of the constraints is bounded by a polynomial in $\mathfrak{A}$. We formalise this by saying that a representation ν is *well described* if there is a polynomial p such that $\langle \nu(\mathfrak{A}) \rangle \leq p(|\mathfrak{A}|)$, for all τ-structures $\mathfrak{A}$. We say that a class of polytopes is *well described* if it has a representation that is well described. Note that this is not the same as the well-described polyhedrons of Grötschel et al. [Grötschel et al. 1988]. However, it is easily seen that from a well-described representation of a polytope, we can easily obtain a well-described polytope also in their sense. Observe that well-described polytopes $\nu(\mathfrak{A})$ have *dimension* bounded by a polynomial in $|\mathfrak{A}|$. We are now ready to define what it means to express the linear optimisation and separation problems in FPC.

Definition 2.6. 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 $\tau_{\mathbb{Q}} \uplus \tau_{\mathrm{vec}}$ in $\tau \uplus \tau_{\mathrm{vec}}$ which takes a $\tau \uplus \tau_{\mathrm{vec}}$-structure encoding a τ-structure $\mathfrak{A}$ and a vector $c \in \mathbb{Q}^V$ over a domain of bits B such that

— $\nu(\mathfrak{A}) \subseteq \mathbb{Q}^V$; and
— $\langle \nu(\mathfrak{A}) \rangle \leq |B|$

to a rational f and vector y such that either (i) $f = 1$, $y = 0^V$, and $\nu(\mathfrak{A})$ is unbounded in the direction of c, or (ii) $f = 0$, and $\nu(\mathfrak{A}) \neq \emptyset$ iff $y \in \nu(\mathfrak{A})$ and $c^\top y = \max\{c^\top x \mid x \in \nu(\mathfrak{A})\}$.

Definition 2.7. 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 $\tau \uplus \tau_{\mathrm{vec}}$ which takes a $\tau \uplus \tau_{\mathrm{vec}}$-structure encoding a τ-structure $\mathfrak{A}$ and a vector $y \in \mathbb{Q}^V$ over a domain of bits B such that

— $\nu(\mathfrak{A}) \subseteq \mathbb{Q}^V$; and
— $\langle \nu(\mathfrak{A}) \rangle \leq |B|$

to a vector c such that either (i) $y \in \nu(\mathfrak{A})$ and $c = 0^V$, or (ii) $c \in \mathbb{Q}^V \backslash \{0^V\}$ with $c^\top y > \max\{c^\top x \mid x \in \nu(\mathfrak{A})\}$ and $\|c\|_\infty = 1$.

3. EXPRESSING THE SEPARATION PROBLEM IN FPC

Let $A \in \mathbb{Q}^{C \times V}$ be a constraint matrix and $b \in \mathbb{Q}^C$ a constraint vector of the polytope $P_{A,b}$. Algorithm 1 presents a straightforward procedure Δ for solving the separation problem for the explicitly-represented polytope $P_{A,b}$. It is not hard to see that the algorithm Δ can be implemented in time polynomial in the size of the natural explicit representation of inputs A, b and x.

Consider expressing the algorithm Δ in fixed-point logic with counting. First note that, by Proposition 2.2, we can define in FPC all the relevant manipulations on ra-

ALGORITHM 1: Δ—a separation oracle for explicitly-represented rational polytopes.

Input: $A \in \mathbb{Q}^{C \times V}$, $b \in \mathbb{Q}^C$ and $x \in \mathbb{Q}^V$.
Output: $c \in \mathbb{Q}^V$ solving the separation problem for the polytope $P_{A,b}$ and x.

1 **if** $Ax \leq b$ **then return** 0^V;
2 **select** $k \in C$ with $A_k x > b_k$;
3 **if** $A_k = 0^V$ **then return** 1^V;
4 **return** $\frac{A_k}{\|A_k\|_\infty}$;

tional values, vectors and matrices, such as norms, addition and multiplication, even when they are indexed by unordered sets. This implies that both lines 1 and 4 of the separation algorithm can be simulated in FPC. However, line 2 poses a problem—FPC is, in general, not able to *choose* a particular element from an unordered set. 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 Δ is superfluous. This can be formally stated as follows.

PROPOSITION 3.1. *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 the class of explicitly-represented polytopes.

THEOREM 3.2. *There is an* FPC *interpretation of τ_{vec} in $\tau_{mat} \uplus \tau_{vec} \uplus \tau_{vec}$ expressing the separation problem with respect to the explicit representation of polytopes.*

PROOF. Let $\mathfrak{X} = A \uplus b \uplus x$ be a structure of signature $\tau_{\text{mat}} \uplus \tau_{\text{vec}} \uplus \tau_{\text{vec}}$, where $A \in \mathbb{Q}^{C \times V}$ is a constraint matrix, $b \in \mathbb{Q}^C$ is a constraint vector of the polytope $P_{A,b}$, and $x \in \mathbb{Q}^V$ is a vector. It follows from Proposition 2.2 that the product Ax can be expressed in FPC. From this interpreted product we can construct a formula $\phi(s)$ of FPC for which $S := \phi(s)^{\mathfrak{X}} \subseteq C$ is the set of constraints which violate the inequality $Ax \leq b$, this is done by comparing the components of the interpreted vector against the components of b. 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, by Proposition 2.2. If $a_S \neq 0^V$, Proposition 3.1 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 can be formalised in FPC. Otherwise, $a_S = 0^V$ and Proposition 3.1 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 $c = 1^V$. Combining all of the above, we have a proof of the theorem. □

4. REDUCING OPTIMISATION TO SEPARATION IN FPC

In this section we present our main technical result: an FPC reduction from optimisation to separation using the classical polynomial-time reduction of the corresponding problems as a subroutine. The classical result can be stated as follows.

THEOREM 4.1 (C.F., E.G., [GRÖTSCHEL ET AL. 1988, THEOREM 6.4.9]). *The linear optimisation problem can be solved in polynomial time for well-described polytopes given by polynomial-time oracles solving their separation problems.*[3]

Below, we prove the following analogous result for fixed-point logic with counting, which formalises Theorem 1.1 from the introduction.

THEOREM 4.2 (OPTIMISATION TO SEPARATION). *Let $\mathcal{P}$ be a class of well-described rational polytopes represented by τ-structures and the function ν. Let there be an* FPC *interpretation of τ_{vec} in $\tau \uplus \tau_{vec}$ expressing the separation problem for $\mathcal{P}$ with respect to ν. Then there is an* FPC *interpretation of $\tau_{\mathbb{Q}} \uplus \tau_{vec}$ in $\tau \uplus \tau_{vec}$ which expresses the linear optimisation problem for $\mathcal{P}$ with respect to ν.*

Observe that these theorems do not imply that every linear optimisation problem can be solved in FPC (or even in polynomial time). Rather one can solve particular classes of linear optimisation problems where domain knowledge can be used to solve the separation problem. Note that because we are using an FPC interpretation to express a solution to a linear optimisation problem, the solution it describes must be canonical in some sense. Moreover, this canonical solution need not, and in the case of matching maximum *cannot* (see Section 5.4), be a vertex of the polytope as is usually the case in the standard algorithms for linear programming.

We have the following generic consequence in the case of explicitly-represented polytopes when Theorem 4.2 is combined with Theorem 3.2.

THEOREM 4.3 (EXPLICIT OPTIMISATION). *There is an* FPC*-interpretation of $\tau_{\mathbb{Q}} \uplus \tau_{vec}$ in $\tau_{mat} \uplus \tau_{vec} \uplus \tau_{vec}$ expressing the linear optimisation problem with respect to the explicit representation of polytopes.*

4.1. Sketch of the Reduction

The main idea behind the proof of Theorem 4.2 is as follows. Suppose we are given a polytope $P \subseteq \mathbb{Q}^V$ by an FPC-interpretation Σ_P that expresses the separation problem for P. A priori the elements of V are indistinguishable. However, Σ_P may expose an underlying order in V as it expresses answers to the separation problem for P. For example, suppose Σ_P on some input expresses a vector $d \in \mathbb{Q}^V$ where the components d_u and d_v for $u, v \in V$ are different. This information can be used to distinguish the components u and v; moreover, it can be used to order the components because d_u and d_v are distinct elements of a field with a total order. As Σ_P is repeatedly applied it may expose additional information about the asymmetry of P. This partial information can be represented by a sequence of equivalence classes $(V_i)_{i=1}^k$ partitioning V, and is progressively refined through further invocations of the separation oracle. Initially all elements of V reside in a single class.

It is natural to consider the polytope P' derived from P by taking its quotient under the equivalence relation defined in this way. Intuitively, this maps polytopes in $\mathbb{Q}^V$ to polytopes in $\mathbb{Q}^k$ by summing the components in each equivalence class to form a single new component which is ordered by the sequence. We call this process *folding* (defined formally in Definition 4.4 below). We observe that a separation oracle for P' can be constructed using Σ_P, provided the answers of Σ_P never expose more asymmetry than was used to derive P'. However, failing to meet this proviso is informative—it further distinguishes the elements of V—and refines the sequence of equivalences classes.

These observations suggest the following algorithm. Start with a sequence (V_1) of exactly one class that contains all of V. Construct the folded polytope P' with respect

[3]The reverse of this theorem also holds: An oracle for the linear optimisation problem can be used to solve the separation problem.

to this sequence and the associated separation oracle $\Sigma_{P'}$ from Σ_P. Attempt to solve the linear optimisation problem on the folded polytope, which lies in an ordered space, using the Immerman-Vardi Theorem (Theorem 2.1) and the classical polynomial-time reduction from optimisation to separation (Theorem 4.1). Should Σ_P at any point answer with a vector that distinguishes more elements of V than the current sequence of equivalence classes, then we: (i) abort the run; (ii) refine the equivalence classes and the folded polytope with this new information; and, finally, (iii) restart the optimisation procedure on this more representative problem instance. Since the number of equivalence classes increases each time the algorithm aborts, it eventually solves the optimisation problem for some P' without aborting. We argue that this solution for P' can be translated into a solution for P.

A key aspect of this approach is that it treats the polynomial-time reduction from optimisation to separation as a blackbox, i.e., it assumes nothing about how the reduction works internally.[4] Before formally describing the algorithm we establish a number of useful definitions and technical properties.

4.2. Folding

Let V be a set. For $k \leq |V|$, let $\sigma : V \to [k]$ be an onto map. We call σ an *index map*. For $i \in [k]$ define $V_i := \{s \in V \mid \sigma(s) = i\}$. The sequence of sets V_i is a partition of V.

Definition 4.4 (*Folding*).
For a vector $x \in \mathbb{Q}^V$ let the *almost-folded* vector $[x]^{\tilde{\sigma}} \in \mathbb{Q}^k$ be given by

$$([x]^{\tilde{\sigma}})_i := \sum_{v \in V_i} x_v, \text{ for } i \in [k].$$

For a vector $x \in \mathbb{Q}^V$ let the *folded* vector $[x]^{\sigma} \in \mathbb{Q}^k$ be given by

$$([x]^{\sigma})_i := \frac{[x]^{\tilde{\sigma}}}{|V_i|}, \text{ for } i \in [k].$$

For a vector $\mathsf{x} \in \mathbb{Q}^k$ let the *unfolded* vector $[\mathsf{x}]^{-\sigma} \in \mathbb{Q}^V$ be given by

$$([\mathsf{x}]^{-\sigma})_v := \mathsf{x}_i, \text{ with } V_i \ni v, \text{ for } v \in V.$$

We say a vector $x \in \mathbb{Q}^V$ *agrees with* σ when for all $v, v' \in V$, $\sigma(v) = \sigma(v')$ implies $x_v = x_{v'}$. It easily follows that if x agrees with σ then $[[x]^{\sigma}]^{-\sigma} = x$. When vectors agree with σ and σ is clear from context we often use the font, as above with x and x, to indicate whether a vector is unfolded and lies in $\mathbb{Q}^V$, or folded and lies in $\mathbb{Q}^k$, respectively. The notion of folding naturally extends to a set $S \subseteq \mathbb{Q}^V$ (and hence polytopes): let $[S]^{\sigma} := \{[s]^{\sigma} \mid s \in S\}$. See Figures 1 and 2 for examples of folding polytopes. Note that $[P]^{\sigma}$ is a projection of P into the k-dimensional space $\mathbb{Q}^k$. Several useful properties of folding and unfolding follow directly from their definitions.

PROPOSITION 4.5. *Let $\sigma : V \to [k]$ be an index map and $c, x \in \mathbb{Q}^V$ such that c agrees with σ. Then,*

$$c^{\top}[[x]^{\sigma}]^{-\sigma} = c^{\top}x = [c]^{\tilde{\sigma}^{\top}}[x]^{\sigma}.$$

[4]It is, in fact, possible to translate the classical reduction from optimisation to separation directly into FPC in the spirit of Section 3. However, this translation quickly becomes mired in intricate error analysis which is both tedious and opaque.

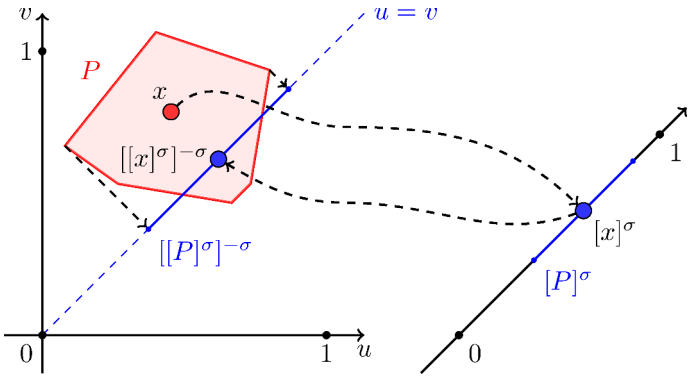

Fig. 1. Folding and unfolding a polytope $P \subseteq \mathbb{Q}^{\{u,v\}}$ with respect to $\sigma = \{u \to 0, v \to 0\}$.

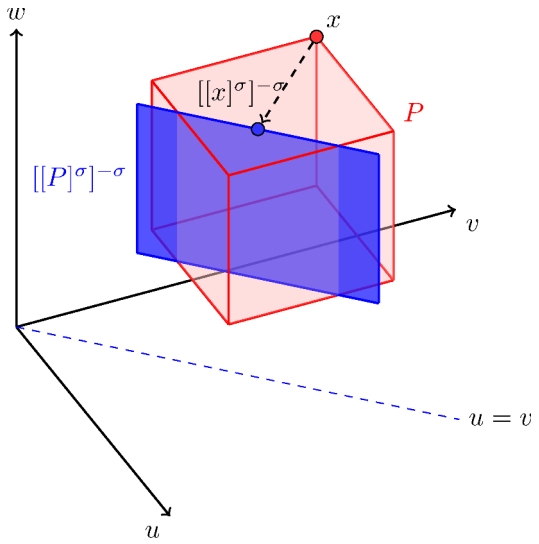

Fig. 2. Folding and unfolding a polytope $P \subseteq \mathbb{Q}^{\{u,v,w\}}$ with respect to $\sigma = \{u \to 0, v \to 0, w \to 1\}$.

PROOF. We begin by proving the first equality. Fix $i \in [k]$. Definition 4.4 implies that

$$\sum_{v \in V_i} ([[x]^\sigma]^{-\sigma})_v - x_v = \sum_{v \in V_i} ([x]^\sigma)_i - \sum_{v \in V_i} x_v = \frac{|V_i|}{|V_i|} \sum_{v' \in V_i} x_{v'} - \sum_{v \in V_i} x_v = 0. \tag{1}$$

Using linearity, the fact that c agrees with σ, and (1) we conclude that

$$c^\top([[x]^\sigma]^{-\sigma} - x) = \sum_{i \in [k]} \sum_{v \in V_i} c_v(([[x]^\sigma]^{-\sigma})_v - x_v) = \sum_{i \in [k]} \frac{([c]^{\tilde{\sigma}})_i}{|V_i|} \sum_{v \in V_i} ([[x]^\sigma]^{-\sigma})_v - x_v = 0.$$

A similar analysis proves the second equality.

$$c^\top x = \sum_{i \in [k]} \sum_{v \in V_i} c_v x_v = \sum_{i \in [k]} \left(\sum_{v' \in V_i} c_{v'} \right) \left(\frac{1}{|V_i|} \sum_{v \in V_i} x_v \right) = [c]^{\tilde{\sigma}\top} [x]^\sigma.$$

□

We conclude by showing that the operations of folding and unfolding can be expressed in FPC. Here, we represent an index map $\sigma : V \to [k]$ over a τ-structure $\mathfrak{A}$ with domain V as a linear pre-order $\lesssim$, whose equivalence classes are the sets $V_1, V_2, \ldots, V_k$ in the order they appear in the pre-order[5]. We also represent vectors in $\mathbb{Q}^k$ as finite structures $\mathfrak{X}$ over vocabulary $\tau_{\text{vec}}^{\text{ord}} := \tau_{\text{vec}} \uplus \{\leqslant\}$, where $\leqslant^{\mathfrak{X}}$ is interpreted as a total ordering of some subset of $\mathrm{dom}(\mathfrak{X})$ of size k.

LEMMA 4.6 (FOLDING AND UNFOLDING IN FPC).

(1) *There is an* FPC *interpretation* Θ_{fold} *of* τ_{vec}^{ord} *in* $\tau_{vec} \uplus \{\lesssim\}$ *such that for any vector-index-map pair* $x \uplus \sigma$*, where* $x \in \mathbb{Q}^V$ *and* $\sigma : V \to [k]$*, it holds that* $\Theta_{fold}(x \uplus \sigma) = [x]^\sigma$.
(2) *There is an* FPC *interpretation* Θ_{unfold} *of* τ_{vec} *in* $\tau_{vec}^{ord} \uplus \{\lesssim\}$ *such that for any vector-index-map pair* $\boldsymbol{x} \uplus \sigma$ *with domain* V*, where* $\boldsymbol{x} \in \mathbb{Q}^k$ *and* $\sigma : V \to [k]$*, it holds that* $\Theta_{unfold}(\boldsymbol{x} \uplus \sigma) = [\boldsymbol{x}]^{-\sigma}$.

PROOF. Firstly, we observe that in FPC we can count the number of equivalence classes of $\lesssim$ satisfying a particular condition [Laubner 2011, Lemma 2.4.3] and thereby, define the classes V_i for $i \in [k]$. Furthermore, for a particular index i, it follows from Proposition 2.2 that we can define $\sum_{v \in V_i} x_v$ in FPC in the usual binary representation. Combining these two observations, it follows that the folding operation can be expressed as an interpretation in FPC. The fact that we can define the equivalence classes of $\lesssim$ also implies directly that unfolding is definable in FPC. □

4.3. Folding Polytopes

The diagrams in Figures 1 and 2 suggest intuitively that the result of folding a polytope is itself a polytope; the following proposition makes this connection concrete.

PROPOSITION 4.7. *Let* P *be a polytope in* $\mathbb{Q}^V$ *and let* $\sigma : V \to [k]$ *be an index map. Then the folded set* $[P]^\sigma$ *is a polytope with* $\langle [P]^\sigma \rangle_{\mathrm{f}} \leq 108k^3|V|^4\langle P \rangle_{\mathrm{f}}$.

PROOF. Let $P = \mathrm{conv}(S_1) + \mathrm{cone}(S_2)$ for two finite sets of points $S_1, S_2 \subseteq \mathbb{Q}^V$. By the linearity of $[\cdot]^\sigma$ we have

$$\begin{aligned}[P]^\sigma &= [\mathrm{conv}(S_1) + \mathrm{cone}(S_2)]^\sigma \\ &= [\mathrm{conv}(S_1)]^\sigma + [\mathrm{cone}(S_2)]^\sigma \\ &= \mathrm{conv}([S_1]^\sigma) + \mathrm{cone}([S_2]^\sigma).\end{aligned}$$

We conclude that $[P]^\sigma$ is a polytope and that the extremal points of $[P]^\sigma$ are the folded extremal points of P.

By [Grötschel et al. 1988, Lemma 6.2.4], we have $\langle [P]^\sigma \rangle_{\mathrm{f}} \leq 3k^2 \langle [P]^\sigma \rangle_{\mathrm{v}}$. To bound the latter consider an extremal point of $u \in P$ such that $[u]^\sigma$ is an extremal point of $[P]^\sigma$. Write $u = (\frac{x_v}{d_v})_{v \in V}$. We bound $\langle [u]^\sigma \rangle$ by expanding it and collecting a common denominator.

$$\langle [u]^\sigma \rangle = \sum_{i \in [k]} \left\langle \frac{1}{|V_i|} \cdot \sum_{v \in V_i} \frac{x_v}{d_v} \right\rangle \leq k \cdot \max_{i \in [k]} \left\langle \frac{1}{|V_i| \cdot \prod_{v \in V_i} d_v} \cdot \sum_{v \in V_i} x_v \cdot \prod_{w \in V_i \setminus \{v\}} d_w \right\rangle.$$

[5]Alternatively, we could represent the index map $\sigma : V \to [k]$ as a number term $\eta(x)$ with one free element variable x, so that for all $v \in V$ it holds that $\sigma(v) = i$ if, and only if, $\eta[v]^{\mathfrak{A}} = i$. The two representations are clearly inter-definable in FPC.

Using the trivial bounds of $|V_i| \leq |V|$, $\langle x_v \rangle, \langle d_v \rangle \leq \langle u \rangle$, and $|V| \geq 1$ we have

$$\begin{aligned}\langle [u]^\sigma \rangle &\leq k \cdot (\lceil \log |V| + 1 \rceil + |V| \cdot \langle u \rangle + \lceil \log |V| + 1 \rceil \cdot |V| \cdot \langle u \rangle) \\ &\leq 3k \cdot \lceil \log |V| + 1 \rceil \cdot |V| \cdot \langle u \rangle \leq 3k \cdot (3 \cdot |V|) \cdot |V| \cdot \langle u \rangle.\end{aligned}$$

From this we conclude that $\langle [P]^\sigma \rangle_{\mathrm{v}} \leq 9k|V|^2 \langle P \rangle_{\mathrm{v}}$ and

$$\langle [P]^\sigma \rangle_{\mathrm{f}} \leq 3k^2 \langle [P]^\sigma \rangle_{\mathrm{v}} \leq 3k^2 \cdot 9k|V|^2 \langle P \rangle_{\mathrm{v}} \leq 27k^3|V|^2 \cdot 4|V|^2 \langle P \rangle_{\mathrm{f}} = 108k^3|V|^4 \langle P \rangle_{\mathrm{f}}.$$

□

For a polytope $P \subseteq \mathbb{Q}^V$ and a point $x \in \mathbb{Q}^V$ (with $x \notin P$) we say that *all separating hyperplanes at x disagree with σ* if there is no $c \in \mathbb{Q}^V$ which both agrees with σ and has $c^\top x > \max\{c^\top y \mid y \in P\}$. This induces an alternative characterisation of the polytope $[P]^\sigma$.

LEMMA 4.8. *Let P be a polytope in $\mathbb{Q}^V$ and $\sigma : V \to [k]$ be an index map. Then*

$$[P]^\sigma = P' := \left\{ \mathsf{x} \in \mathbb{Q}^k \;\middle|\; \begin{array}{c} [\mathsf{x}]^{-\sigma} \in P \text{ or all separating hyperplanes} \\ \text{at } [\mathsf{x}]^{-\sigma} \text{ disagree with } \sigma \end{array} \right\}.$$

PROOF. We show both inclusions.

1. $[P]^\sigma \subseteq P'$*:* Let $\mathsf{x} \in [P]^\sigma$. By definition there is a point $x \in P$ such that $[x]^\sigma = \mathsf{x}$. Suppose $x = [\mathsf{x}]^{-\sigma}$, then $[\mathsf{x}]^{-\sigma} \in P$ and hence $\mathsf{x} \in P'$. Thus assume $[\mathsf{x}]^{-\sigma} \neq x$. Let $c \in \mathbb{Q}^V$ be any vector agreeing with σ. By Proposition 4.5 we have $c^\top [\mathsf{x}]^{-\sigma} = c^\top [[x]^\sigma]^{-\sigma} = c^\top x$. Since $x \in P$, c is not the normal of a separating hyperplane through $[\mathsf{x}]^{-\sigma}$. We conclude that all separating hyperplanes through $[\mathsf{x}]^{-\sigma}$ disagree with σ and hence that $\mathsf{x} \in P'$.

2. $[P]^\sigma \supseteq P'$*:* Let $\mathsf{x} \in P'$. Suppose $[\mathsf{x}]^{-\sigma} \in P$, then $\mathsf{x} = [[\mathsf{x}]^{-\sigma}]^\sigma \in [P]^\sigma$. Thus assume that $[\mathsf{x}]^{-\sigma} \notin P$ and that all separating hyperplanes through $[\mathsf{x}]^{-\sigma}$ disagree with σ. This means that for any vector $c \in \mathbb{Q}^V$ that agrees with σ the hyperplane through $[\mathsf{x}]^{-\sigma}$ with normal c intersects P and thus there is a point $y \in P$ which has $c^\top [\mathsf{x}]^{-\sigma} = c^\top y$. This further implies that $c^\top [\mathsf{x}]^{-\sigma} \leq \max\{c^\top y \mid y \in P\}$. Since c agrees with σ, Proposition 4.5 implies that

$$[c]^{\tilde{\sigma}^\top} \mathsf{x} \leq \max\{[c]^{\tilde{\sigma}^\top} [y]^\sigma \mid y \in P\}.$$

Observe $[\{c \in \mathbb{Q}^V \mid c \text{ agrees with } \sigma\}]^{\tilde{\sigma}} = \mathbb{Q}^k$. This means for any vector $c' \in \mathbb{Q}^k$, $c'^\top \mathsf{x} \leq \max\{c'^\top [y]^\sigma \mid y \in P\}$. In particular, for every constraint defining the polytope $[P]^\sigma$, x also satisfies that constraint. We conclude that $\mathsf{x} \in [P]^\sigma$. □

4.4. Expressing Optimisation in FPC

Suppose we are given a polytope $P \subseteq \mathbb{Q}^V$ via a separation oracle Δ_P, and a vector c indicating a linear objective. The algorithm maintains an index map $\sigma : V \to [k]$ that indicates a sequence of equivalence classes of V which have not been distinguished by the algorithm so far. Initially this index map is given by ordering variables according to their relative values in c. Under the assumption that σ accurately describes the symmetries of P we execute the polynomial-time reduction from optimisation to separation on the polytope $[P]^\sigma$ and objective $[c]^{\tilde{\sigma}}$. Since $[P]^\sigma$ lies in an ordered space, it follows from the Immerman-Vardi theorem that the reduction can be expressed in fixed-point logic with counting.

To this end, a separation oracle $\Delta_{[P]^\sigma}$ must be specified for the polytope $[P]^\sigma$. Given a point $\mathsf{x} \in \mathbb{Q}^k$, we argue that the result of applying Δ_P to the unfolding of x either determines that it is in P, and hence x is in $[P]^\sigma$; or determines a separating hyperplane for P. If a separating hyperplane is determined, it can be folded into a separating

ALGORITHM 2: OPT*—an instrumentation of the reduction from optimisation to separation.

Input:
— A well-described polytope $P \subseteq \mathbb{Q}^V$ with a separation oracle Δ_P, and
— a linear objective $c \in \mathbb{Q}^V$.

Output:
— $f = 1$ and $y = 0^V$, if P is unbounded along c; or otherwise
— $f = 0$ and $y \in \mathbb{Q}^V$, s.t. if $P \neq \emptyset$ then $y \in P$ and $c^\top y = \max\{c^\top x \mid x \in P\}$.

```
σ ← REFINE(0^V, c);
while true do
    (f, x) ← OPT([P]^σ, Δ_{[P]^σ}, [c]^σ̃);
    if aborted with σ' then
        σ ← σ';
    else
        if f = 1 then return (1, 0^V);
        σ' ← REFINE(σ, Δ_P([x]^{-σ}));
        if σ ≠ σ' then σ ← σ' else return (0, [x]^{-σ});
    end
end
```

```
oracle Δ_{[P]^σ}(x) :
    d ← Δ_P([x]^{-σ});
    σ' ← REFINE(σ, d);
    if σ ≠ σ' then abort(σ');
    if d = 0^V then return [0^V]^σ̃;
    return [d]^σ̃ / ‖[d]^σ̃‖_∞;
end
```

hyperplane for $[P]^\sigma$, but only if the hyperplane normal agrees with σ. In the case the separating hyperplane disagrees with σ, our assumption about P is violated, and our separation oracle does not have enough information to proceed. Indeed, folding the resulting normal may produce 0^k which is not a valid answer. In this case, the algorithm aborts the run of the linear optimisation algorithm, and returns the disagreeing hyperplane normal. The algorithm then combines the disagreeing normal with its current index map σ to produce a new index map which is consistent with σ and agrees with the disagreeable hyperplane normal. This strictly increases the number of equivalence classes of variables induced by the index map. The above procedure can abort at most $|V|$ times before σ exactly characterises the order of V relative to P. After this point the linear optimisation algorithm cannot abort and hence must solve the optimisation problem for $[P]^\sigma$ which can be unfolded into a solution for P.

With this intuition in mind the formal proof is as follows.

PROOF OF THEOREM 4.2. For completeness the entire procedure OPT* is described in Algorithm 2. The algorithm uses two subroutines REFINE and OPT. The subroutine REFINE(σ, d) takes as input an index map σ of V represented in $\mathbb{N}^V$ and a vector $d \in \mathbb{Q}^V$ and computes a new index map σ' with the following two properties:

— for all $v, v' \in V$ with $\sigma(v) < \sigma(v')$, $\sigma'(v) < \sigma'(v')$, and
— for all $v, v' \in V$ with $\sigma(v) = \sigma(v')$, $\sigma'(v) < \sigma'(v')$ iff $d_v < d_{v'}$.

It is straightforward to observe that when REFINE(σ, d) produces an index map σ' which is different from σ, then σ' induces strictly more equivalence classes on V than σ does. Clearly, no index map can induce more than $|V|$ equivalence classes. The subroutine OPT solves the linear optimisation problem on an ordered space $\mathbb{Q}^k$ with a given linear objective and a polytope given by a separation oracle. Mirroring Definition 2.6, OPT returns an integer-vector pair (f, y) which is $(1, 0^k)$ when the objective value is unbounded and $(0, y)$, where $y \in \mathbb{Q}^k$ is an optimal point in the polytope if, and only if, the polytope is non-empty.

We first argue that the algorithm is correct, assuming the correctness of OPT and REFINE. For any index map $\sigma : V \to [k]$, $[P]^\sigma$ is a polytope by Proposition 4.7. We show that the procedure $\Delta_{[P]^\sigma}$ described in lines 12 to 18 acts as a separation oracle for $[P]^\sigma$ provided the answer given by the separation oracle Δ_P agrees with σ. If $\Delta_P([\mathsf{x}]^{-\sigma})$ outputs $d = 0^V$, then this indicates that $[\mathsf{x}]^{-\sigma} \in P$, and hence $\mathsf{x} \in [P]^\sigma$ by Proposition 4.8. Trivially 0^V agrees with σ, so $[0^V]^{\tilde{\sigma}} = 0^k$ is returned by $\Delta_{[P]^\sigma}$ correctly indicating that $\mathsf{x} \in [P]^\sigma$. Otherwise, $d \neq 0^V$ and indicates that $[\mathsf{x}]^{-\sigma} \notin P$ but $d^\top [\mathsf{x}]^{-\sigma} > \max\{d^\top y \mid y \in P\}$. If d agrees with σ we have, by Proposition 4.5, $[d]^{\tilde{\sigma}^\top}\mathsf{x} > \max\{[d]^{\tilde{\sigma}^\top}[y]^\sigma \mid y \in P\}$. This is equivalent to $[d]^{\tilde{\sigma}^\top}\mathsf{x} > \max\{[d]^{\tilde{\sigma}^\top}\mathsf{y} \mid \mathsf{y} \in [P]^\sigma\}$. Hence $[d]^{\tilde{\sigma}^\top}$ is the normal of a separating hyperplane of $[P]^\sigma$ through x. Since d agrees with σ, $\sigma' = \sigma$ and $\frac{[d]^{\tilde{\sigma}}}{\|[d]^{\tilde{\sigma}}\|_\infty}$ is correctly returned. If d does not agree with σ, then REFINE produces a $\sigma' \neq \sigma$ and the procedure aborts. We conclude that *(i)* when $\Delta_{[P]^\sigma}$ does not abort it behaves as a separation oracle for $[P]^\sigma$, and *(ii)* when $\Delta_{[P]^\sigma}$ aborts the returned index map σ' is a strict refinement of σ. Thus $\Delta_{[P]^\sigma}$ is a separation oracle for $[P]^\sigma$, provided it does not abort. When OPT runs on $\Delta_{[P]^\sigma}$ without aborting the result must be a solution to the linear optimisation problem on $[P]^\sigma$.

Let $\mathsf{x} \in [P]^\sigma$ be such that $[c]^{\tilde{\sigma}^\top}\mathsf{x} \geq \max\{[c]^{\tilde{\sigma}^\top}\mathsf{y} \mid \mathsf{y} \in [P]^\sigma\}$, i.e., it is a solution to the linear optimisation problem on $[P]^\sigma$ along $[c]^{\tilde{\sigma}}$. By Proposition 4.8 this means that either *(i)* $[\mathsf{x}]^{-\sigma} \in P$ or *(ii)* $\Delta_P([\mathsf{x}]^{-\sigma})$ must disagree with σ. Applying Δ_P to $[\mathsf{x}]^{-\sigma}$ distinguishes these two cases. In case *(i)*, $[c]^{\tilde{\sigma}^\top}\mathsf{x} = c^\top[\mathsf{x}]^{-\sigma} \geq \max\{c^\top y \mid y \in P\}$ by Proposition 4.5, because the initialisation of σ forces c to agree with σ. This means that $[\mathsf{x}]^{-\sigma}$ is a solution to the linear optimisation problem for the polytope P and the objective c. In case *(ii)*, $[\mathsf{x}]^{-\sigma} \notin P$ but $\Delta_P([\mathsf{x}]^{-\sigma})$ is guaranteed to improve the index map. In the case that the linear optimisation algorithm returns that $[P]^\sigma$ is unbounded in the direction of $[c]^{\tilde{\sigma}}$, it implies, via similar analysis, that P is unbounded in the direction c. Finally, when the optimisation algorithm reports that $[P]^\sigma$ is empty we conclude that P must be empty as well, because if P contains at least one point then $[P]^\sigma$ must also contain at least one point. The algorithm correctly translates the solutions for the linear optimisation problem for $[P]^\sigma$ back to solutions for P. This means that OPT* returns the correct result.

We now observe that this algorithm runs in polynomial time. The main loop cannot execute more than $|V|$ times, because, as established above, at each step either the index map σ is improved to induce more equivalence classes—up to $|V|$ classes—or the algorithm returns a correct solution to the linear optimisation problem on P. The size of all of the objects referred to by the algorithm can be polynomially bounded by a function of the input length. In particular, since P is well described by Δ_P, there is a polynomial bound on its bit complexity and this induces a bound on the bit complexity of $[P]^\sigma$ through Proposition 4.7 and implies that $[P]^\sigma$ is well described. This implies that the bit complexity of values in the algorithm can be bounded by some fixed polynomial. This means that folding and unfolding can be computed in polynomial time. Similarly, a naive implementation of the subroutine REFINE can be seen to run in poly-

nomial time in $|V|$ and the bit complexity of its input rational vector. Since $[P]^\sigma$ is a well-described polytope with a polynomial-time separation oracle $\Delta_{[P]^\sigma}$ we can use the polynomial-time algorithm for OPT from Theorem 4.1 to solve the linear optimisation problem on $[P]^\sigma$. Combining all these parts implies that $\textsc{Opt}^*$ is a polynomial-time algorithm.

We conclude by arguing that the behavior of $\textsc{Opt}^*$ can be simulated in FPC. By Lemma 4.6, vector folding and unfolding relative to an index map σ can be expressed in FPC. It is similarly routine to express REFINE in FPC by defining the equivalence classes and then counting sizes to determine the correct position of each equivalence class relative to an FPC-definable σ and vector—see [Laubner 2011, Lemma 2.4.3] for details. In the next claim we establish that the separation problem for $[P]^\sigma$ can be defined in FPC.

To be precise, we want to code in a single oracle the separation problem for $[P]^\sigma$ for arbitrary σ. For this, we consider the vocabulary $\tau_S := \tau \cup \{S\} \cup \tau_{\text{vec}}$ where S is a new binary relation symbol. We think of a structure over this vocabulary as consisting of a τ-structure $\mathfrak{A}$, representing the polytope $\nu(\mathfrak{A}) \subseteq \mathbb{Q}^V$; a vector $y \in \mathbb{Q}^V$ and a linear pre-order S on V coding the index map σ, so that $S(x, y)$ iff $\sigma(x) \leq \sigma(y)$. We define the $[P]^\sigma$-separation problem analogously to Definition 2.7. That is to say, this problem is expressible in FPC if there is an FPC-interpretation that takes a τ_S-structure representing a polytope P, a vector y and index map σ to a τ_{vec} structure representing a vector that is a solution to the separation problem for $[P]^\sigma$.

CLAIM 1. *The $[P]^\sigma$-separation problem is expressible in* FPC.

PROOF OF CLAIM. By assumption, there is an FPC interpretation Σ_P expressing the separation problem for P. Furthermore, as noted above, there is an FPC interpretation for expressing the REFINE routine. Finally, by Lemma 4.6 we can define the folding of a definable vector and hence can define $d := \Delta_P([\mathsf{x}]^{-\sigma})$ in FPC. Composing all of these interpretations, we can conclude that the $[P]^\sigma$-separation problem can be expressed in FPC. □

By the claim, it follows that there is an FPC-interpretation for the combination of $\textsc{Opt}$ and the separation oracle given by $\Sigma_{[P]^\sigma}$, because the polytope $[P]^\sigma$ lies in an ordered space and the Immerman-Vardi Theorem (Theorem 2.1) indicates that any polynomial-time property of ordered structures can be defined in FPC. It is easy to see that the algorithm's main loop and control structure can be simulated in FPC. Combining everything, and using the fact that FPC is closed under composition (see p. 6), gives an FPC-interpretation simulating $\textsc{Opt}^*$. This also shows that there is an interpretation in FPC that can express the linear optimisation problem for P, assuming the separation problem for P can also be defined in the logic. This concludes the proof of Theorem 4.2.

5. APPLICATIONS IN COMBINATORIAL OPTIMISATION

In this section we demonstrate a number of applications of our main technical result, Theorem 4.2, for expressing combinatorial optimisation problems in fixed-point logic with counting. We sequentially build up FPC interpretations for the maximum flow, minimum cut, minimum odd cut and maximum matching problems. In doing so we show that these graph problems can be solved efficiently while respecting the underlying graph abstraction.

5.1. Maximum Flows

Let $G = (V, c)$ be a graph with non-negative edge capacities, that is, $c : V \times V \to \mathbb{Q}_{\geq 0}$. A capacitated graph $G = (V, c)$ is *symmetric* if for all $u, v \in V$, $c(u, v) = c(v, u)$. 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 $0 \leq f(u,v) \leq c(u,v)$ on each pair of distinct $u, v \in V$ and conservation constraints $\sum_{v \in V}(f(v,u) - f(u,v)) = 0$ on all vertices $u \in V \backslash \{s,t\}$. The *value* $\mathrm{val}(f)$ of the flow f is simply the difference between in-flow and out-flow at t, i.e., $\sum_{v \in V}(f(v,t) - f(t,v))$. Observe that any flow f can be *normalised* to f' so that for any pair of distinct $u, v \in V$ at least one of $f'(u,v)$ and $f'(v,u)$ is zero (i.e., if $f(u,v) \geq f(v,u)$, set $f'(u,v) := f(u,v) - f(v,u)$ and $f'(v,u) := 0$; obviously this preserves the capacity constraints, the conservation constraints and the value of the flow). A *maximum* (s,t)*-flow of* G is a flow whose value is maximum over all (s,t)-flows. Observe that if f_1, f_2 are two (s,t)-flows in G, and $\alpha \in \mathbb{Q}$ with $0 \leq \alpha \leq 1$, then $f' := \alpha \cdot f_1 + (1-\alpha) \cdot f_2$ is an (s,t)-flow of G and $\mathrm{val}(f') = \alpha \cdot \mathrm{val}(f_1) + (1-\alpha) \cdot \mathrm{val}(f_2)$. Moreover, if f_1 and f_2 are maximum (s,t)-flows, then so is any convex combination f'. Let $G|_f := (V, c-f)$ denote the *residual graph* of G with respect to the flow f.

The standard formulation of the maximum (s,t)-flow problem as a linear program is as follows:

$$\begin{aligned} &\max \sum_{v \in V}(f(v,t) - f(t,v)) \qquad \text{subject to} \\ &\sum_{v \in V}(f(v,u) - f(u,v)) = 0, \quad \forall u \in V \backslash \{s,t\} \\ &0 \leq f(u,v) \leq c(u,v), \quad \forall u \neq v \in V. \end{aligned} \tag{2}$$

By explicitly interpreting the above flow polytope from a given capacitated graph G and then applying Theorem 4.3 we can express maximum flows in G via an FPC interpretation.

THEOREM 5.1. *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*, along with vertices* $s \neq t$ *to a* τ_{mat}*-structure coding a maximum* (s,t)*-flow of* G*.*

PROOF. Observe that there are $|V|(|V|-1)$ variables in linear program (2) corresponding to $f(u,v)$ for distinct $u, v \in V$. The program has $2|V|^2 - 4$ constraints. Both the variables and constraints can be indexed by tuples of elements from V. The variables are naturally indexed by $I = \{(u,v) \mid u, v \in V \text{ and } u \neq v\}$. For each $u \in V \backslash \{s,t\}$ there are two constraints corresponding to the second line of (2) which we can index by the triples (u,u,s) and (u,u,t); and for each $u, v \in V$ with $u \neq v$ there are two constraints corresponding to the third line of (2), which we index by (u,v,s) and (u,v,t) respectively. Write J for the set of triples in V^3 that index the constraints. It can then easily be established that the maximum (s,t)-flow linear program can be defined by an FPC interpretation. That is to say, suppose that a capacitated graph (V,c) is given as a τ_{mat}-structure with domain V where the rational matrix $c \in \mathbb{Q}_{\geq 0}^{V \times V}$ 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 defines the $I \times J$ constraint matrix A and J-vector b encoding the corresponding flow polytope. Note that the flow polytope is (i) bounded, because each variable is constrained from both above and below, and (ii) non-empty, because the capacities in G are non-negative and hence the zero flow is a member of the polytope. Properties (i) and (ii) imply that solving the linear optimisation problem on the flow polytope must produce an optimum point (flow). Thus, because the interpreted flow polytope is explicitly represented, Theorem 4.3 immediately gives an FPC interpretation expressing the optimisation problem for the interpreted flow linear program, and hence a maximum (s,t)-flow of (V,c). □

Note that 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 consistent with the most refined index map determined by the FPC interpretation of Theorem 4.2. This is possible because convex combinations of flows are also flows. In our remaining applications—minimum (odd) cut and maximum matching—the analogous property does not hold: convex combinations of cuts or matchings are not necessarily cuts or matchings. 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 discussed at the end of the introduction, that defining a canonical maximum matching is not possible.

5.2. Minimum Cuts

An (s,t)-*cut* of a capacitated graph $G = (V, c)$ is a subset C of the vertices V which contains s but not t. The *value* $\mathrm{val}(C)$ of the cut C is the sum of the capacity of edges going from vertices in C to vertices in $V \backslash C$. 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. By the max-flow/min-cut theorem, a maximum (s,t)-flow and a minimum (s,t)-cut have the same value (c.f., e.g., [Cormen et al. 2009]). This duality allows the construction of minimum cuts from maximum flows. In this section we describe an FPC formula defining a canonical minimum (s,t)-cut in a graph using the FPC interpretation for the maximum (s,t)-flow problem given by Theorem 5.1.

Expressing Canonical Minimum Cut in FPC. First, we define a notion of directed reachability in capacitated graphs. A vertex v is *reachable* from a vertex u if there is a path in the graph which follows directed edges with non-zero capacity (this is exactly directed reachability in the graph induced by eliminating zero capacity edges). Let f be a maximum (s,t)-flow in $G = (V, c)$ with normalised flow f'. Define $C_f := \{v \in V \mid v \text{ reachable from } s \text{ in } G|_{f'}\}$. C_f is a minimum (s,t)-cut in G. Since f' is normalised, every edge leaving C_f must be at full capacity in f'. Note that, in the residual flow graph $G|_{f'}$, t is not reachable from s and, by definition, $C_f = C_{f'}$.

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

LEMMA 5.2. *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.*

PROOF. Suppose not. There are two distinct minimum (s,t)-cuts $C := C_f$ and $C' := C_{f'}$ with corresponding normalised (s,t)-flows f and f', respectively. Since C and C' are different there exists, without loss of generality, $v \in C' \backslash C$. Consider the flows through $\overline{C} \cap C'$. We use a, a', b, b', c, c' to denote the net flows into and out of this set. See Figure 3 for definitions. By definition of C and C' there is no flow in f from $\overline{C}$ to C nor is there flow in f' from $\overline{C'}$ to C' as otherwise vertices in the complementary cuts would be reachable from s.

The flow conservation constraints require the flow into $\overline{C} \cap C'$ be matched by the outflow in both f and f'. This implies that $a + b = c$ and $a' = b' + c'$. In addition $a \geq a'$ and $c' \geq c$, because these edges must be at full capacity in f and f' respectively. Combining these equalities and inequalities produces $a + b \leq c'$ and $b' + c' \leq a$. Adding these two constraints together gives $a + b + b' + c' \leq a + c'$. Since all values are non-negative we have $b = b' = 0$. This implies $a = c$ and $a' = c'$. Then, reusing $a \geq a'$ and $c' \geq c$ we conclude $a = c = a' = c'$. This means that in f' the edges going from $C \cap C'$ to $\overline{C} \cap C'$ are at full capacity, and thus no vertex in $\overline{C} \cap C'$ is reachable from s in flow f'. This is a contradiction.

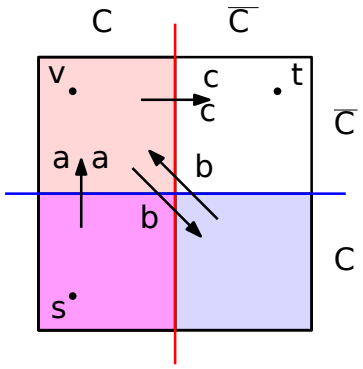

Fig. 3. Diagram for the proof of Lemma 5.2. Here the labels indicate the net flow between two sets under flows f and f'.

Since the flow between $C \cap C'$ and $\overline{C} \cap C'$ is the same in both f and f', flow f' witnesses that $C \cap C'$ is a minimum (s,t)-cut of G. This implies the "moreover" part of the statement and completes the proof.[6] □

Lemma 5.2 implies that for any maximum (s,t)-flow f, C_f is a *unique* (s,t)-cut of the graph G and for this reason we call it the *canonical* minimum (s,t)-cut of G: $K_{G,s,t}$. Given the FPC interpretation Φ from Theorem 5.1 expressing a maximum (s,t)-flow f, it is not difficult to construct a formula of FPC which defines the normalised flow f' and then the set of vertices $C_f = K_{G,s,t}$ using the inflationary fixed-point operator to determine reachability. This argument is formalised in the following theorem.

THEOREM 5.3. *There is a formula $\xi(x,s,t)$ of* FPC *which given a τ_{mat}-structure coding a capacitated graph $G = (V,c)$, along with distinct vertices s and t, defines the vertices in $K_{G,s,t}$.*

5.3. Minimum Odd Cuts

An *odd* cut of a capacitated graph $G = (V,c)$ is a proper subset C of the vertices V with $|C|$ odd. We can extend a capacitated graph $G = (V,c)$ to a *marked* capacitated graph $G' = (V,c,R)$ with a marking $R \subseteq V$, and use τ_{mark} to denote the vocabulary containing a single unary relation symbol for R. A cut C' of a marked graph G' is said to be a *marked cut*, if both C' and $V \backslash C'$ contain a vertex in R. A marked cut C' of a marked graph $G' = (V,c,R)$ with $|R|$ even is said to be an *odd* marked cut if $|C' \cap R|$ is odd (note that this corresponds to the simpler notion of an odd cut when $R = V$). The goal of the *minimum odd (marked) cut* problem is to determine a minimum-value odd (marked) cut of a given input graph. The following result by Goemans and Ramakrishnan shows that the set of all canonical minimum marked (s,t)-cuts of a graph G' captures some minimum odd marked cut of G'.

THEOREM 5.4 (IMPLICIT IN [GOEMANS AND RAMAKRISHNAN 1995, THEOREM 2]). *Let $G' = (V,c,R)$ be a marked symmetric capacitated graph with $|R|$ even and nonzero. There exists distinct vertices $s,t \in R$ such that $K_{G',s,t}$ is a minimum odd marked cut of G'.*[7]

[6] Note that this proof is similar to the "lemma on a quadrangle" from [Dinitz et al. 1976], but that proof does not immediately go through because $\overline{C} \cap C'$ may not be a (s,t)-cut.

[7] Note that the conference version [Anderson et al. 2013] of the present paper includes an independent proof of this theorem.

Combining Theorems 5.3 & 5.4 gives an FPC formula describing exactly those minimum odd marked cuts.

THEOREM 5.5. *There is a formula $\kappa(s,t)$ of* FPC *which given a $(\tau_{mat} \uplus \tau_{mark})$-structure coding a marked symmetric capacitated graph $G' = (V, c, R)$, defines those pairs of distinct vertices $s, t \in R$ where $K_{G',s,t}$ is a minimum odd marked cut of G'. If $|R|$ is even and nonzero, the relation defined by this formula is not empty.*

PROOF. Consider the following procedure for locating a set of minimum odd marked cuts in G': for all distinct $s, t \in R$ 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. If $|R|$ is even and nonzero, then Theorem 5.4 indicates that some cuts remain and that those cuts are minimum odd marked cuts of G'.

This procedure can be implemented in an FPC formula using the formula ξ defining the canonical minimum (s,t)-cut of G' given by Theorem 5.3. We define the auxiliary FPC formula

$$\textsc{Odd}(s,t) := \#_x[\xi(x,s,t) \wedge x \in R] \equiv 1 \pmod 2$$

indicating whether $K_{G',s,t}$ is an odd marked cut. We again use ξ, now naturally viewed as a number (0 or 1), to define in FPC

$$\textsc{Val}(s,t) := \sum_x \xi(x,s,t) \cdot \sum_y (1 - \xi(y,s,t)) \cdot c(x,y)$$

indicating the rational number $\mathrm{val}(K_{G',s,t})$. Finally, define

$$\begin{aligned}\kappa(s,t) := \; & s \neq t \wedge s,t \in R \wedge \textsc{Odd}(s,t) \wedge \#_x[x \in R] \equiv 0 \pmod 2 \\ & \wedge \forall u,v (u \neq v \wedge u,v \in R \wedge \textsc{Odd}(u,v)) \Rightarrow \textsc{Val}(s,t) \leq \textsc{Val}(u,v).\end{aligned}$$

□

The FPC formula κ is critical to expressing the separation problem for the matching polytope in FPC.

5.4. 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., $|M| = \frac{|V|}{2}$.

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* linear program

$$\max \; c^\top y \quad \text{subject to} \quad Ay \leq b, \; y \geq 0^E. \tag{3}$$

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, due to Edmonds, as a

rational linear program.

$$\begin{aligned} \max\ & c^\top y \qquad \text{subject to} \\ & Ay \leq b, \\ & y \geq 0^E, \\ & y(W) \leq \frac{1}{2}(b(W)-1), \ \ \forall W \subseteq V \text{ with } b(W) \text{ odd}, \end{aligned} \tag{4}$$

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 (3), also satisfy the additional constraints that are added in (4). 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 [Edmonds 1965] shows something stronger.

LEMMA 5.6 ([EDMONDS 1965, THEOREM P]). *The extremal points of* (4) *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 (4). 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 (4) 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.[8] 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 (4). Combining this with Theorem 4.2 gives an FPC interpretation expressing the b-matching optimum.

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 [Padberg and Rao 1982]. Let y be the point which we wish to separate from the matching polytope. 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

[8] Recently this was shown to be tight in a strong sense [Rothvoß 2013].

constraint of this type it suffices to find W, with $b(W)$ odd, 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 $R := \{v \in V \mid b_v \text{ is odd}\}$. If $|R|$ is odd, add z to R. Thus we have a marked symmetric graph $H = (U, d, R)$. Note that it is easy to interpret in FPC the graph H from the input τ_{match}-structure coding the b-matching instance and the point y interpreted as a τ_{vec}-structure. To define the universe $V \cup \{z\}$ from V, a standard way is to take the set of all pairs in $V \times V$ and quotient under the equivalence relation that identifies (x, y) with (x', y') whenever $x \neq y$ and $x' \neq y'$. 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 (4).

By Theorem 5.4, there is a violated odd set constraint iff for some $s, t \in R$ the canonical minimum (s, t)-cut is a minimum odd marked cut with value less than 1. For each pair $s, t \in R$ there is at most one minimum odd marked cut $W_{s,t}$ with this property, and it is definable from the parameters s and t. We conclude, using Theorem 5.3 and Lemma 5.2, that we can define a family of violated set constraints within FPC. Summing these defined violated constraints produces a canonical violated constraint which must be non-trivial by Proposition 3.1. Thus, as in Theorem 3.2 there is an FPC interpretation expressing the separation problem for the polytope in the linear program (4).

LEMMA 5.7. *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.*

Like the maximum flow problem in Section 5.1, the b-matching polytope is both compact and non-empty. By combining Lemma 5.7 and Theorem 4.2 with respect to the natural well-described representation of b-matching polytopes, we conclude that there is an FPC interpretation expressing the value of the maximum b-matching of a graph.

THEOREM 5.8. *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.*

6. CONCLUSION

We prove that the linear programming problem 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 [Blass et al. 1999] by concluding that there is a formula of fixed-point logic with counting that defines the size of a maximum b-matching in a graph. More generally, we demonstrate that a number of combinatorial graph problems that can be efficiently solved can also be efficiently solved while respecting the graph abstraction. It is our hope that our results provide some insight towards the eventual resolution of Chandra and Harel's question about the existence of a logical characterisation of P. From here, there are number of natural research directions to consider.

Convex programming. A polytope is an instance of 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 geometric optimisation problems, e.g., those with semi-definite or quadratic constraints. It is possible that our methods for efficiently

preserving abstraction can be extended to these settings. To be more concrete, the standard method for solving the separation problem on semi-definite programs involves locating eigenvectors with non-positive eigenvalues to witness violations of the positive-semi-definitiness constraints. Naïvely, these eigenvectors are determined by constructing an eigenbasis of the constraint matrix. Such a choice of basis cannot be expressed in FPC! The proof of Theorem 4.2 sidesteps a similar basis choice by collapsing spaces with indistinguishable coordinates—can this approach be generalised to semi-definite programs?

Completeness. Linear programming is complete for polynomial time under logspace reductions [Dobkin et al. 1979]. 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 (under the explicit representation) is complete for FPC under such weak reductions, or, perhaps, first-order interpretations with counting? Even if linear programming is not complete, there may be other interesting combinatorial problems that can be expressed in FPC via a reduction to linear programming. There has been some work examining generalisations and improvements to the b-matching approach we followed (c.f., e.g., [Caprara and Fischetti 1996; Letchford et al. 2004]), is it possible to translate these results into FPC?

LP hierarchies and FPC. Another intriguing connection between counting logics and linear programming is established in [Atserias and Maneva 2012; Grohe and Otto 2012] where it is shown that the hierarchy of Sherali-Adams relaxations [Sherali and Adams 1990] of the graph isomorphism integer program interleaves with equivalence in k-variable logic with counting (C^k). It is suggested [Atserias and Maneva 2012] 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, as long as the approximating LP is given by an FPC interpretation over the problem instance. It would be interesting to investigate the consequences of this to see how inexpressibility results in FPC can be translated to lower bound results on integrality gaps or other measures.

ACKNOWLEDGMENT

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.

REFERENCES

M. Anderson and A. Dawar. 2014. On Symmetric Circuits and Fixed-Point Logics. In *STACS*. LIPIcs, 41–52.

M. Anderson, A. Dawar, and B. Holm. 2013. Maximum Matching and Linear Programming in Fixed-Point Logic with Counting. In *LICS*. IEEE, 173–182.

A. Atserias, A. Bulatov, and A. Dawar. 2009. Affine systems of equations and counting infinitary logic. *Theor. Comput. Sci.* 410, 18 (2009), 1666–1683.

A. Atserias and E. Maneva. 2012. Sherali-Adams relaxations and indistinguishability in counting logics. In *ITCS*. ACM, 367–379.

A. Blass and Y. Gurevich. 2005. 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].

A. Blass, Y. Gurevich, and S. Shelah. 1999. Choiceless polynomial time. *Ann. Pure Appl. Logic* 100 (1999), 141–187.

A. Blass, Y. Gurevich, and S. Shelah. 2002. On polynomial time computation over unordered structures. *J. Symbolic Logic* 67, 3 (2002), 1093–1125.

J-Y. Cai, M. Fürer, and N. Immerman. 1992. An Optimal Lower Bound on the Number of Variables for Graph Identification. *Combinatorica* 12, 4 (1992), 389–410.

A. Caprara and M. Fischetti. 1996. $\{0, 1/2\}$-Chvátal-Gomory cuts. *Math. Program.* 74, 3 (1996), 221–235.

A. Chandra and D. Harel. 1982. Structure and complexity of relational queries. *J. Comput. Syst. Sci.* 25, 1 (1982), 99–128.

T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. 2009. *Introduction to Algorithms* (*3. ed.*). MIT Press. I–XIX, 1–1292 pages.

G. Dantzig. 1963. *Linear programming and extensions*. Princeton University Press. (most recent edition published in 1998).

A. Dawar, M. Grohe, B. Holm, and B. Laubner. 2009. Logics with Rank Operators. In *LICS*. IEEE, 113–122.

E.A. Dinitz, A.V. Karazanov, and M.V. Lomonosov. 1976. On the structure of the system of minimum edge cuts in a graph. *Studies in Discrete Optimizations* (1976), 290–306. In Russian.

D. Dobkin, R.J. Lipton, and S. Reiss. 1979. Linear programming is log-space hard for *P*. *Inform. Process. Lett.* 8, 2 (1979), 96–97.

R. Duan and S. Pettie. 2014. Linear-Time Approximation for Maximum Weight Matching. *J. ACM* 61, 1, Article 1 (Jan. 2014), 23 pages. DOI:http://dx.doi.org/10.1145/2529989

H.D. Ebbinghaus and J. Flum. 1999. *Finite Model Theory*. Springer.

J. Edmonds. 1965. Maximum Matching and a Polyhedron with $0, 1$ Vertices. *J. Res. Nat. Bur. Stand.* 69 B (1965), 125–130.

M.X. Goemans and V.S. Ramakrishnan. 1995. Minimizing submodular functions over families of sets. *Combinatorica* 15, 4 (1995), 499–513.

M. Grohe. 2012. Fixed-point definability and polynomial time on graphs with excluded minors. *J. ACM* 59, 5, Article 27 (2012), 64 pages.

M. Grohe, K. Kersting, M. Mladenov, and E. Selman. 2013. Dimension Reduction via Colour Refinement. *CoRR* abs/1307.5697 (2013), 17 pages.

M. Grohe and M. Otto. 2012. Pebble Games and Linear Equations. In *CSL*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 289–304. DOI:http://dx.doi.org/10.4230/LIPIcs.CSL.2012.289

M. Grötschel, L. Lovász, and A. Schrijver. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1 (1981), 169–197. Issue 2.

M. Grötschel, L. Lovász, and A. Schrijver. 1988. *Geometric algorithms and combinatorial optimization*. Springer-Verlag Berlin / New York.

B. Holm. 2010. *Descriptive Complexity of Linear Algebra*. Ph.D. Dissertation. University of Cambridge.

N. Immerman. 1986. Relational queries computable in polynomial time. *Inform. Control* 68, 1-3 (1986), 86–104.

N. Immerman. 1999. *Descriptive Complexity*. Springer-Verlag.

N. Karmarkar. 1984. A new polynomial-time algorithm for linear programming. In *STOC*. ACM, 302–311.

L.G. Khachiyan. 1980. Polynomial algorithms in linear programming. *USSR Comp. Math. Math* 20, 1 (1980), 53–72.

V. Klee and G.J. Minty. 1972. How good is the simplex algorithm?. In *Proceedings of the Third Symposium on Inequalities*, Oved. Shisha (Ed.). Academic Press, New York-London, 159–175.

B. Laubner. 2011. *The Structure of Graphs and New Logics for the Characterization of Polynomial Time*. Ph.D. Dissertation. Humboldt-Universitt zu Berlin.

A.N. Letchford, G. Reinelt, and D. O. Theis. 2004. A faster exact separation algorithm for blossom inequalities. In *Integer programming and combinatorial optimization*. Springer, 196–205.

L. Libkin. 2004. *Elements of Finite Model Theory*. Springer.

S. Micali and V. V. Vazirani. 1980. An $O(\sqrt{|V|}|E|)$ Algorithm for Finding Maximum Matching in General Graphs. In *21st Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 17–27.

M.W. Padberg and M.R. Rao. 1982. Odd Minimum Cut-Sets and b-Matchings. *Math. Oper. Res.* 7, 1 (1982), 67–80.

B. Rossman. 2010. Choiceless Computation and Symmetry. In *Fields of Logic and Computation*. Springer, 565–580.

T. Rothvoß. 2013. The matching polytope has exponential extension complexity. *CoRR* abs/1311.2369 (2013), 19 pages.

H.D. Sherali and W.P. Adams. 1990. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics* 3, 3 (1990), 411–430.

N.Z. Shor. 1972. Utilization of the operation of space dilatation in the minimization of convex functions. *Cybern. Syst. Anal.* 6, 1 (1972), 7–15.

N.Z. Shor. 1977. Cut-off method with space extension in convex programming problems. *Cybern. Syst. Anal.* 13, 1 (1977), 94–96.

D.A. Spielman and S.-H. Teng. 2004. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *JACM* 51, 3 (2004), 385–463.

M. Vardi. 1982. The complexity of relational query languages. In *STOC*. ACM, 137–146.

V. V. Vazirani. 1994. A Theory of Alternating Paths and Blossoms for Proving Correctness of the $O(\sqrt{V}E)$ General Graph Maximum Matching Algorithm. *Combinatorica* 14 (1994), 71–109.

D.B. Yudin and A.S. Nemirovskii. 1976. Informational complexity and efficient methods for the solution of convex extremal problems. *Matekon* 13, 2 (1976), 3–25.

Received ; revised ; accepted