# Definability of Semidefinite Programming and Lasserre Lower Bounds for CSPs

Anuj Dawar and Pengming Wang
University of Cambridge Computer Laboratory
Email: {anuj.dawar, pengming.wang}@cl.cam.ac.uk

***Abstract*—We show that the ellipsoid method for solving semidefinite programs (SDPs) can be expressed in fixed-point logic with counting (FPC). This generalizes an earlier result that the optimal value of a linear program can be expressed in this logic.**

**As an application, we establish lower bounds on the number of levels of the Lasserre hierarchy required to solve many optimization problems, namely those that can be expressed as finite-valued constraint satisfaction problems (VCSPs). In particular, we establish a dichotomy on the number of levels of the Lasserre hierarchy that are required to solve the problem exactly. We show that if a finite-valued constraint problem is not solved exactly by its basic linear programming relaxation, it is also not solved exactly by any sub-linear number of levels of the Lasserre hierarchy.**

**The lower bounds are established through logical undefinability results. We show that the SDP corresponding to any fixed level of the Lasserre hierarchy is interpretable in a VCSP instance by means of FPC formulas. Our definability result of the ellipsoid method then implies that the solution of this SDP can be expressed in this logic. Together, these results give a way of translating lower bounds on the number of variables required in counting logic to express a VCSP into lower bounds on the number of levels required in the Lasserre hierarchy to eliminate the integrality gap.**

**As a special case, we obtain the same dichotomy for the class of** MAXCSP **problems, generalizing earlier Lasserre lower bound results by Schoenebeck [17]. Recently, and independently of the work reported here, a similar linear lower bound in the Lasserre hierarchy for general-valued CSPs has also been announced by Thapper and Živný [20], using different techniques.**

## I. Introduction

Semidefinite programming has developed into an essential tool in optimization. Part of its success is owed to its wide applicability to provide approximations to hard combinatorial problems through so-called relaxation hierarchies. These hierarchies describe systematic ways to obtain increasingly good approximations to problems expressed as 0–1 integer linear programs. Indeed, since the problem of determining an optimal solution to a 0–1 integer programming problem is NP-complete, in principle any problem in NP can be so expressed in this framework.

Any integer programming problem admits a linear programming relaxation obtained by dropping the integrality constraints. This relaxed linear program can then be solved by standard polynomial-time algorithms, however it admits solutions that are not solutions to the original integer program. The gap between the optimal solution to the integer program and its linear programming relaxation is known as an *integrality gap*. There are various ways that the linear programming relaxation may be tightened by additional constraints to more closely correspond to the original problem. Several systematic ways have been studied in the literature of constructing hierarchies of ever tighter linear or semidefinite programs, including those of Sherali-Adams [18], Lovasz-Schrijver [14] and Lasserre [13]. Of these, the Lasserre hierarchy is the strongest. It gives, for each $t$, a semidefinite program of size $n^{O(t)}$ (where $n$ is the size of the original integer program) that defines a feasible region whose projection on to the original variables includes the solutions of the integer program. When $t = n$, this projection is exactly the convex hull of the solutions to the original integer program. When this can be achieved for smaller values of $t$, we get substantially faster algorithms for solving (possibly approximately) the original problem. For many combinatorial optimization problems, the Lasserre relaxations provide the best known approximation algorithms (see [7]).

In this paper we establish a definability result for the problem of determining the optimal value of a given SDP. We show that there is an interpretation in fixed-point logic with counting (FPC) that can, for an explicitly given semidefinite program, define its optimal solution, up to a given approximation. This result generalizes earlier work by Anderson et al. [3], [4] who showed an analogous result for the case of linear programs. Extending their techniques, our proof is centered around formalizing the ellipsoid method for solving SDPs in logic.

As a direct application of this result we consider SDPs obtained as Lasserre relaxations of certain constraint problems. Here, we aim to establish integrality lower bounds, i.e. to establish for particular combinatorial optimization problems a lower bound on the value of $t$ such that the $t^{\text{th}}$ level of the Lasserre hierarchy shows no integrality gap. Schoenebeck [17] previously established such lower bounds, showing a linear lower bound on $t$ for a variety of Boolean constraint satisfaction problems, including Max-$k$-XORSAT. We show that those lower bounds are part of a general pattern. Indeed, we demonstrate a dichotomy on the minimum value of $t$ needed in the Lasserre hierarchy to establish exact solutions to optimization problems in the framework of *finite-valued constraint satisfaction problems* (VCSPs). That is, any such VCSP is either already solved by simply relaxing the integral-

This work was partly carried out during the Logical Structures in Computation programme at the Simons Institute for the Theory of Computing.

ity constraints in its 0–1 linear program formulation (resulting in the so-called *basic linear program relaxation* (BLP)), or requires a linear number of Lasserre relaxation steps to be solved exactly. As a direct consequence, we obtain the same dichotomy for the class of (weighted) MAXCSP problems. More recently, and independently of our work, Thapper and Živný [20] have announced a similar lower bound on general-valued CSPs, obtained using different methods.

The study of the complexity of VCSPs has been quite successful in the recent past, culminating in the dichotomy result by Thapper and Živný [19]. There, a complete characterization of the tractable and intractable cases of VCSPs are shown. Namely, any VCSP is either solved exactly by its BLP; or the problem MAXCUT reduces to it, and it is NP-hard. Our result complements this dichotomy by showing a linear lower bound for the levels of Lasserre relaxations required to exactly solve the hard cases. This characterization of VCSPs into those solvable by their BLP, and those to which MAXCUT reduces, also applies in the context of logical definability. It is known from [9] that in the former case, the class is definable in fixed-point logic with counting (FPC), while in the latter case there is no constant $k$ such that it is definable using only $k$ variables, even in an infinitary logic with counting.

In the present paper, we establish a more fine-grained view on the above undefinability result by lifting a previous undefinability result on classical CSPs from [5], and connecting it to the number of levels of the Lasserre hierarchy needed to solve a corresponding VCSP exactly. To be precise, we show that if $k$ variables are required to define the VCSP in logic with counting, then $\Omega(k)$ levels of the Lasserre hierarchy are needed to capture the corresponding feasible region. This is obtained as a consequence of our main definability result for SDPs. It allows us to show that, given an integer program, for each $t$, the optimal value of its $t^{\text{th}}$ Lasserre relaxation is definable in FPC, using a linear number of variables. The dichotomy is then completed by showing that, for every VCSP that is not captured by the basic linear program relaxation, there is a linear lower bound on the number of variables required to define it.

We begin by introducing the necessary definitions from combinatorial optimization, logic and constraint satisfaction problems in Section II. Section III formulates the main results and the steps establishing them are then given in Sections IV, V, and VI.

## II. Background

**Notation.** We write $\mathbb{N}$ for the natural numbers, $\mathbb{Z}$ for the integers, $\mathbb{Q}$ for the rational numbers, and use a superscript plus for the non-negative subset, e.g. $\mathbb{Z}^+ = \mathbb{N} \cup \{0\}$. For a number $d \in \mathbb{N}$, we write $[d]$ to denote the segment of the natural numbers $\{1, 2, \ldots, d\}$. For a set $S$, we denote its *power set* by $\wp(S)$, and we write $\wp_k(S)$ for the set of subsets of $S$ that have size $k$. We write $\mathrm{ar}(t)$ to denote the *arity* of $t$, where $t$ could be a tuple, function, or relation. We use the symbol $\dot\cup$ for the *disjoint union* of sets.

Given sets $A$ and $I$, an $A$-valued *vector* $v$ indexed by $I$ is a function $v : I \to A$. Often we simply use the subscript notation, writing $v_i$ for $v(i)$. When there is no explicit index set given, vectors are indexed by $[d]$ for some $d$. A *matrix* $M$ is a vector which is indexed by a product set $I \times J$. We use $M_{i,j}$ to denote $M(i, j)$.

For a rational valued vector $v \in \mathbb{Q}^I$ over some index set $I$, its *norm* $\|v\|$ is defined as the $L^2$-norm over $\mathbb{Q}^I$. The *inner product* of two vectors $a, b \in \mathbb{Q}^I$ is defined as $\langle a, b\rangle := \sum_{i\in I} a_i b_i$. In the case of matrices $M, N \in \mathbb{Q}^{I\times J}$, this definition results in $\langle M, N\rangle := \sum_{(i,j)\in I\times J} M_{i,j}N_{i,j}$. Note that the norm of a matrix $M$ is defined as the norm of $M$ seen as a vector. For a set $\mathcal{F} \subseteq \mathbb{Q}^I$ and a vector $v \in \mathbb{Q}^I$, we define the *distance* $d(v, \mathcal{F})$ in the usual way, i.e. as $d(v, \mathcal{F}) := \min_{x\in\mathcal{F}} \|x-v\|$. The *ball* $\mathcal{B}(v, r)$ around $v$ with radius $r \in \mathbb{Q}$ is the set $\mathcal{B}(v, r) := \{x \in \mathbb{Q}^I \mid \|x - v\| \le r\}$.

A rational valued symmetric matrix $M \in \mathbb{Q}^{I\times I}$ is *positive semidefinite*, if for any $x \in \mathbb{Q}^I$, it holds $x^T Mx \ge 0$. We use $M \succeq 0$ to denote that $M$ is positive semidefinite.

### A. Semidefinite Optimization

In a typical semidefinite program we are interested in the entries of a symmetric matrix $X \in \mathbb{Q}^{V\times V}$ that maximizes the value of an objective function $\langle C, X\rangle$, subject to a set of constraints of the form $\langle A_i, X\rangle \le b_i$ with the additional constraint that $X$ is a positive semidefinite matrix.

**Definition 1.** Let $V, M$ be sets, and let $V$ be non-empty. A *semidefinite program (SDP)* is given by an objective matrix $C \in \mathbb{Q}^{V\times V}$, a $\mathbb{Q}^{V\times V}$-valued vector $\mathcal{A} \in \mathbb{Q}^{M\times(V\times V)}$, and a vector $b \in \mathbb{Q}^M$.

We call $\mathcal{F}_{\mathcal{A},b} := \{X \in \mathbb{Q}^{V\times V} \mid X \succeq 0, \langle A_i, X\rangle \le b_i, A_i = \mathcal{A}(i), i \in M\}$ the set of *feasible solutions*.

We sometimes call sets that can be defined as feasible regions of an SDP a *positive semidefinite set*. The above definition covers SDPs that are in the so-called *conic standard form*. Sometimes it is more convenient to specify SDPs in their *inequality standard form*. In this form, the SDP is instead given by a matrix $Z \in \mathbb{Q}^{M\times M}$, a matrix-valued vector $\mathcal{Y} \in \mathbb{Q}^{V\times(M\times M)}$, and an objective vector $c \in \mathbb{Q}^V$. The feasible region is then defined as $\mathcal{F}_{\mathcal{Y},Z} := \{x \in \mathbb{Q}^V \mid Z + \sum_{v\in V} x_v\mathcal{Y}_v \succeq 0\}$. Since the two standard forms can be converted into each other by adding, substituting, and rearranging of a linear number of variables, we will use whichever representation is most convenient for any given case.

In the context of optimization, we are interested in the two following problems.

**Definition 2.** Let $V$ be a non-empty set. Given a vector $c \in \mathbb{Q}^V$ and a convex set $\mathcal{F} \subseteq \mathbb{Q}^V$, the *strong optimization problem* is to either find an element $y = \mathrm{argmax}_{x\in\mathcal{F}}\langle c, x\rangle$, or to determine that $\mathcal{F}$ is empty, or that $\max_{x\in\mathcal{F}}\langle c, x\rangle$ is unbounded.

In the *weak optimization problem* we are given an additional error parameter $\delta > 0$, and want to determine an element $y$

that is $\delta$-close to $\mathcal{F}$, i.e. $d(y, \mathcal{F}) \leq \delta$, that is also $\delta$-maximal, i.e. $\langle c, y\rangle + \delta \geq \max_{x\in\mathcal{F}}\langle c, x\rangle$, or, again, to determine that $\max_{x\in\mathcal{F}}\langle c, x\rangle$ is unbounded.

**Definition 3.** Let $V$ be a non-empty set. Given a vector $y \in \mathbb{Q}^V$ and a convex set $\mathcal{F} \subseteq \mathbb{Q}^V$, the *strong separation problem* is the problem of determining either that $y \in \mathcal{F}$, or finding a vector $s \in \mathbb{Q}^V$ with $\langle s, y\rangle > \sup\{\langle s, x\rangle \mid x \in \mathcal{F}\}$ and $\|s\|_\infty = 1$.

In the *weak separation problem*, we are given an additional parameter $\delta > 0$, and are looking to determine that either $y$ is $\delta$-close to $\mathcal{F}$, i.e. $d(y, \mathcal{F}) \leq \delta$, or to find a vector $s \in \mathbb{Q}^V$, such that $\langle s, y\rangle + \delta > \sup\{\langle s, x\rangle \mid x \in \mathcal{F}\}$ and $\|s\|_\infty = 1$.

The relationship between the optimization and separation problem of a given convex set is well-studied and is most prominently expressed by Grötschel, Lovász, and Shrijver [11] as being polynomial time equivalent. More precisely, with the additional assumptions that the set $\mathcal{F}$ is *full-dimensional* ($\mathcal{F}$ has positive volume in $\mathbb{Q}^V$) and *bounded* ($\mathcal{F}$ is contained within a ball of finite radius), the weak optimization problem for $\mathcal{F}$ is solvable in polynomial time if, and only if, the corresponding weak separation problem is solvable in polynomial time. The following is essentially the statement of Theorem 4.2.7 in [11].

**Theorem 1.** Let $V$ be a non-empty set, and $\mathcal{F} \subseteq \mathbb{Q}^V$ a full-dimensional convex set that is located inside the ball $\mathcal{B}(0, R)$ for some known value $R$. The weak optimization problem on $\mathcal{F}$ is solvable in polynomial time if its weak separation problem is solvable in polynomial time.

As one of our main results, we show that the polynomial time reduction described above can in fact be realized in fixed-point logic with counting when considering positive semidefinite sets. Furthermore, we also show that a corresponding weak separation oracle is definable in the same logic as well. Together, this yields the definability result for SDPs in Theorem 6.

### *B. Constraint Satisfaction Programs*

We consider CSPs that are parameterized by a fixed finite *domain* and a *constraint language*, and their optimization variant of *finite-valued* CSPs.

**Definition 4.** A *domain* $D$ is a finite, non-empty set. A *constraint language* $\Gamma$ over $D$ is a set of relations over $D$, where each $R \in \Gamma$ is a relation of some arity $m = \mathrm{ar}(R)$, and $R \subseteq D^m$.

An instance of the *constraint satisfaction problem* over $(D, \Gamma)$, or $\mathrm{CSP}(D, \Gamma)$, is a pair $I = (V, C)$, where $V$ is a finite set of *variables*, and $C$ is a finite set of *constraints*. Each constraint $c \in C$ is a pair $(s, R)$ associating a function $R \in \Gamma$ with a *scope* $s \in V^{\mathrm{ar}(R)}$.

We say an assignment $h : V \to D$ *satisfies* a constraint $c = (s, R)$ if $h(s) \in R$. The goal is to decide whether there exists an assignment that satisfies all constraints $c \in C$.

**Definition 5.** Let $D$ be a domain. A *finite-valued constraint language* $\Gamma$ is a set of functions, where each $f \in \Gamma$ has some arity $m = \mathrm{ar}(f)$, and $f : D^m \to \mathbb{Z}^+$.

An instance of the *valued constraint satisfaction problem* over $(D, \Gamma)$, or $\mathrm{VCSP}(D, \Gamma)$, is a pair $I = (V, C)$, where $V$ is a finite set of *variables*, and $C$ is a finite set of *constraints*. Each constraint $c \in C$ is a triple $c = (s, f, w)$ associating a relation $f \in \Gamma$ with a *scope* $s \in V^{\mathrm{ar}(f)}$ and a *weight* $w \in \mathbb{Z}^+$.

The *value* of an assignment $h : V \to D$ for an instance $I$ is given as $\mathrm{Val}_I(h) := \sum_{(s,f,w)\in C} w \cdot f(h(s))$. The goal is to determine the maximum value $\mathrm{Opt}(I) = \max_h \mathrm{Val}_I(h)$.

**Example 1.** Let MAXCUT be the problem of determining the value of the maximum cut in a graph $G = (V, E)$ with edge weights $w : E \to \mathbb{Z}^+$. Furthermore, we fix $D = \{0, 1\}$, and $\Gamma = \{f\}$ where $f(x, y) = 0$ if $x = y$, and $f(x, y) = 1$ if $x \neq y$. An instance of MAXCUT can be interpreted as an instance $I$ of $\mathrm{VCSP}(D, \Gamma)$: The variables are the vertices $V$, and we have the constraint $((u, v), f, w(u, v))$ for every edge $(u, v) \in E$. The value of the maximum cut is exactly $\mathrm{Opt}(I)$.

**Example 2.** A common class of optimization problems is MAXCSP. Let $\mathrm{MAXCSP}(D, \Gamma)$ be the optimization problem of determining the maximal number of constraints that can be satisfied in a given instance of $\mathrm{CSP}(D, \Gamma)$. It is not difficult to see that every $\mathrm{MAXCSP}(D, \Gamma)$ is a finite-valued CSP. For any $m$-ary relation $R$, define a function $f_R : D^m \to \{0, 1\}$ as $f(t) = 1$ if $t \in R$ and $f(t) = 0$ if $t \notin R$. Let $\Gamma'$ be the finite-valued constraint language that consists of $f_R$ for all $R \in \Gamma$. Then, $\mathrm{MAXCSP}(D, \Gamma) = \mathrm{VCSP}(D, \Gamma')$.

When talking complexity classes and reductions, it is often more convenient to also phrase VCSPs as decision problems. Abusing notation, we will use $\mathrm{VCSP}(D, \Gamma)$ to also denote the set of pairs $(I, t)$ such that $I$ is an instance with $\mathrm{Opt}(I) \geq t$. The decision problem is then, given any pair $(I, t)$, to decide whether $(I, t) \in \mathrm{VCSP}(D, \Gamma)$.

In the study of valued constraint satisfaction problems, linear programming in particular has proven to be a useful tool. In fact, every instance $I$ of $\mathrm{VCSP}(D, \Gamma)$ is equivalent to the following integer linear program.

For an instance $I = (V, C)$, the LP contains variables $\lambda_{c,x}$ for every $c \in C$ with $c = (s, f, w)$ and $x \in D^{\mathrm{ar}(s)}$, and $\mu_{v,a}$ for every $v \in V$ and $a \in D$. A solution that sets a variable $\lambda_{c,x}$ to 1 then corresponds to an assignment that assigns the scope of the constraint $c$ to the tuple $x$. In order to maintain consistency of the assignment between constraints, the variable $\mu_{v,a}$ encodes whether the variable $v$ is assigned the value $a$. The objective is then to maximize the value of the assignment.

The 0–1 program is then given below.

$$\begin{aligned}
\max \sum_{c\in C}\sum_{x\in D^{\mathrm{ar}(s)}} \lambda_{c,x}\cdot w\cdot f(x), \quad & \text{where } c=(s,f,w), \text{ s.t.}\\
\sum_{x\in D^{\mathrm{ar}(s)};x_i=a} \lambda_{c,x} = \mu_{s_i,a} \quad & \forall c\in C, a\in D, i\in[\mathrm{ar}(s)]\\
\sum_{a\in D}\mu_{v,a}=1 \quad & \forall v\in V\\
\lambda_{c,x}\in\{0,1\} \quad & \forall c\in C, x\in D^{\mathrm{ar}(s)}\\
\mu_{v,a}\in\{0,1\} \quad & \forall v\in V, a\in D
\end{aligned}$$

If we relax the integrality constraints of the associated integer LP (i.e. we replace the conditions $\lambda_{c,x},\mu_{v,a}\in\{0,1\}$ by $0\le\lambda_{c,x},\mu_{v,a}\le 1$) we obtain the *basic linear program relaxation* $\mathrm{BLP}(I)$. Since this allows rational assignments, this LP can be solved exactly in polynomial time. In general the optimal value of $\mathrm{BLP}(I)$ only gives an overestimate of the optimal value $\mathrm{Opt}(I)$ to the VCSP. However, there are $(D,\Gamma)$ for which any instance $I$ of $\mathrm{VCSP}(D,\Gamma)$ is solved by $\mathrm{BLP}(I)$ exactly, and solving the LP gives an exact algorithm for $\mathrm{VCSP}(D,\Gamma)$. Thapper and Živný [19] give a complete characterization of those cases – and show that in all other cases the problem is NP-hard.

**Theorem 2** ( [19])**.** For any domain $D$, and any finite-valued constraint language $\Gamma$, either every instance $I$ of $\mathrm{VCSP}(D,\Gamma)$ is solved by $\mathrm{BLP}(I)$; or the problem MAXCUT polynomial-time reduces to $\mathrm{VCSP}(D,\Gamma)$.

In this paper we expand on this dichotomy result, and show a linear lower bound of the required levels of the Lasserre hierarchy for all the cases not solved by the BLP relaxation. This is stated in Theorem 7.

Note that that the framework of finite-valued CSPs is not an overall generalization of classical CSPs, and hence any dichotomy result for the former framework is not immediately valid for the latter one. The framework that generalises both is often called *general-valued* CSPs, and allows functions in $\Gamma$ to take values from $\mathbb{Z}^+\cup\{-\infty\}$. A main conceptual difference is that finite-valued CSPs are always feasible, i.e. their value is always finite, while general-valued CSPs may be infeasible, i.e. when their value is $-\infty$. Our proof technique is specifically suited for the finite-valued variant as this guarantees that the resulting SDPs in the Lasserre hierarchy are non-empty.

### C. Lasserre Hierarchy

The *Lasserre hierarchy* of relaxations defines sequences of SDPs that give increasingly good approximations to 0–1 programs.

**Definition 6.** Let $V$, $U$ be sets and $\mathcal{K}:=\{x\in\mathbb{Q}^V \mid Ax\ge b\}$ a polytope given by $A\in\mathbb{Q}^{U\times V}, b\in\mathbb{Q}^U$.

For a vector $y\in\mathbb{Q}^{\wp(V)}$, and an integer $t$ with $1\le t\le|V|$, we define the *$t^{th}$ moment matrix* of $y$, $M_t(y)$ as the $\wp_t(V)\times\wp_t(V)$-matrix with entries

$$M_t(y)_{I,J}:=y_{I\cup J},\ \text{for } |I|,|J|\le t.$$

Similarly, the *$t^{th}$ moment matrix of slacks* of $y,A,b$, and some $u\in U$ is given by

$$S_t^u(y)_{I,J}:=\sum_{v\in V}A_{u,v}y_{I\cup J\cup\{v\}}-b_u y_{I\cup J},\ \text{for } |I|,|J|\le t.$$

Finally, the *$t^{th}$ level of the Lasserre hierarchy* of $\mathcal{K}$, $\mathrm{Las}_t(\mathcal{K})$ is the set defined by

$$\begin{aligned}
\mathrm{Las}_t(\mathcal{K}) \quad := \quad & \{y\in\mathbb{Q}^{\wp_{2t+1}(V)} \mid y_\emptyset=1, M_t(y)\succeq 0,\\
& S_t^u(y)\succeq 0 \text{ for all } u\in U\}.
\end{aligned}$$

We write $\mathrm{Las}_t^\pi(\mathcal{K}):=\{y_{\{v\}}, v\in V \mid y\in\mathrm{Las}_t(\mathcal{K})\}$ for the projection of $\mathrm{Las}_t(\mathcal{K})$ onto the original variables.

The following basic properties of the Lasserre hierarchy establish that $\mathrm{Las}_t(\mathcal{K})$ is indeed a relaxation of $\mathcal{K}\cap\{0,1\}^V$. We write $\mathcal{K}^*$ for the polytope that is defined by the convex hull of the integer points in $\mathcal{K}$, i.e. $\mathcal{K}^*:=\mathrm{conv}(\mathcal{K}\cap\{0,1\}^V)$.

**Lemma 3.** Let $\mathcal{K}=\{x\in\mathbb{Q}^V \mid Ax\ge b\}$. Then,

1) $\mathcal{K}^*\subseteq\mathrm{Las}_t^\pi(\mathcal{K})$ for all $t\in\{1,\dots,|V|\}$.
2) $\mathrm{Las}_0^\pi(\mathcal{K})\supseteq\mathrm{Las}_1^\pi(\mathcal{K})\supseteq\ldots\supseteq\mathrm{Las}_{|V|}^\pi(\mathcal{K})$.
3) $\mathrm{Las}_0^\pi(\mathcal{K})\subseteq\mathcal{K}$, and $\mathcal{K}^*=\mathrm{Las}_{|V|}^\pi(\mathcal{K})$.

*Proof.* See, for instance, in [16] □

**Definition 7.** Let $I=(c,\mathcal{K})$ be a 0–1 linear program with an objective vector $c\in\mathbb{Q}^V$ optimizing over a feasible region $\mathcal{K}\cap\{0,1\}^V$. We say that $I$ is *captured* at the $t^{\text{th}}$ level of the Lasserre hierarchy if the projection of the $t^{\text{th}}$ level Lasserre relaxation coincides with the convex hull of integer solutions in the direction of $c$, i.e. $\max_{x\in\mathcal{K}^*}\langle c,x\rangle=\max_{x\in\mathrm{Las}_t^\pi(\mathcal{K})}\langle c,x\rangle$. We write $l(I)$ for the minimum $t$, such that $I$ is captured at the $t^{\text{th}}$ level.

For a class of 0–1 linear programs $C$, we say that $C$ is captured at the $t^{\text{th}}$ level, if every program in $C$ is captured at the $t^{\text{th}}$ level of the Lasserre hierarchy.

We denote by $L_C(n)$ the function that maps an integer $n$ to the first level $t$ at which every 0–1 program in $C$ with size $n$ is captured. That is, $L_C(n):=\max_{I\in C;|V|\le n}l(I)$.

We see that at a sufficiently high level, namely at most at level $t=|V|$, any 0–1 program is captured by the $t^{\text{th}}$ level Lasserre relaxation. In those cases, the optimum of the Lasserre set yields not only an approximate optimum, but is the exact optimal value of the original 0–1 problem.

In this paper, we establish a dichotomy for VCSPs with respect to $L_C(n)$: For every $(D,\Gamma)$, $C=\mathrm{VCSP}(D,\Gamma)$, either $L_C(n)=0$ ($\mathrm{VCSP}(D,\Gamma)$ is solved by the basic linear program relaxation); or $L_C(n)\in\Omega(n)$.

### D. Logic

We use the standard notions of relational *vocabularies* and *structures*.

A relational *vocabulary*, or *signature*, $\tau$ is a finite sequence of relation and constant symbols $(R_1,\dots,R_k,c_1,\dots,c_l)$, where every relation symbol $R_i$ has a fixed arity $a_i\in\mathbb{N}$. A structure $\mathbf{A}=(\mathrm{dom}(\mathbf{A}),R_1^{\mathbf{A}},\dots,R_k^{\mathbf{A}},c_1^{\mathbf{A}},\dots,c_l^{\mathbf{A}})$ over the signature $\tau$ (or a *$\tau$-structure*) consists of a non-empty set

$\mathrm{dom}(\mathbf{A})$, called the *universe* of $\mathbf{A}$, together with relations $R_i^{\mathbf{A}} \subseteq \mathrm{dom}(\mathbf{A})^{a_i}$ and constants $c_j^{\mathbf{A}} \in \mathrm{dom}(\mathbf{A})$ for each $1 \leq i \leq k$ and $1 \leq j \leq l$. Members of the set $\mathrm{dom}(\mathbf{A})$ are called the *elements* of $\mathbf{A}$ and we define the *size* of $\mathbf{A}$ to be the cardinality of its universe, often written as $|\mathbf{A}|$.

*1) Fixed-point Logic with Counting:* Fixed-point logic with counting (FPC) is an extension of inflationary fixed-point logic with the ability to express the cardinality of definable sets. We refer to [15] for its formal definition and semantics.

In descriptive complexity theory, fixed-point logics frequently play an important role. Throughout the paper, we make frequent use of the Immerman-Vardi theorem [10], which establishes that fixed-point logic can express all polynomial-time properties of finite ordered structures. It follows that in FPC we can express all polynomial-time relations on the number domain.

In addition to the logic FPC, we also consider $\mathrm{C}^k$, the fragment of first-order logic with counting quantifiers consisting of those formulas that can be written using at most $k$ distinct variables. It is easy to see that any structure with $n$ elements can be described up to isomorphism by a formula using no more than $n$ variables. It follows that any isomorphism-closed collection of structures, each of which has no more than $n$ elements, can be defined by a formula with no more than $n$ variables.

The minimum number of variables needed to define a class of structures in first-order logic with counting turns out to be a useful measure of complexity. This motivates the definition of the *counting width* of a class.

**Definition 8.** For any class of structures $\mathcal{C}$, the *counting width* of $\mathcal{C}$ is the function $\nu_{\mathcal{C}} : \mathbb{N} \to \mathbb{N}$ where $\nu_{\mathcal{C}}(n)$ is the minimum value $k$ such that there is a formula $\phi$ in $\mathrm{C}^k$, such that for any structure $\mathbf{A}$ with $|\mathrm{dom}(\mathbf{A})| \leq n$, $\mathbf{A} \models \phi \Leftrightarrow \mathbf{A} \in \mathcal{C}$.

It is clear that $\nu_{\mathcal{C}} \leq n$ for any class $\mathcal{C}$. It is known that if $\mathcal{C}$ is definable in FPC, then $\nu_{\mathcal{C}}$ is bounded by a constant (see [15]). The converse is not true in general as there are even undecidable classes $\mathcal{C}$ for which $\nu_{\mathcal{C}}$ is bounded by a constant. However, the converse holds in special cases, such as for constraint satisfaction problems. Here we have a dichotomy: every $\mathcal{C} = \mathrm{CSP}(D, \Gamma)$ is either definable in FPC or has unbounded $\nu_{\mathcal{C}}$. For an explanation see [9] where this result is extended to finite-valued CSPs.

In Section VI, we show that the counting width of finite-valued CSPs is either bounded by a constant, or is $\Omega(n)$. We use this for our main result to establish a similar dichotomy on the number of levels of the Lasserre hierarchy needed to capture the 0–1 linear programs coding instances of VCSPs.

*2) Interpretations:* We frequently need to consider ways of defining one structure within another in some logic L, such as first-order logic or FPC. Consider two signatures $\sigma$ and $\tau$ and a logic L. An *$m$-ary* L*-interpretation of $\tau$ in $\sigma$* is a sequence of formulae of L in vocabulary $\sigma$ consisting of: (i) a formula $\delta(x)$; (ii) a formula $\varepsilon(x, y)$; (iii) for each relation symbol $R \in \tau$ of arity $k$, a formula $\phi_R(x_1, \ldots, x_k)$; and (iv) for each constant symbol $c \in \tau$, a formula $\gamma_c(x)$, where each $x$, $y$ or $x_i$ is an $m$-tuple of free variables. We call $m$ the *width* of the interpretation. We say that an interpretation $\Theta$ associates a $\tau$-structure $\mathbf{B}$ to a $\sigma$-structure $\mathbf{A}$ if there is a surjective map $h$ from the $m$-tuples $\{a \in \mathrm{dom}(\mathbf{A})^m \mid \mathbf{A} \models \delta[a]\}$ to $\mathbf{B}$ such that:

- $h(a_1) = h(a_2)$ if, and only if, $\mathbf{A} \models \varepsilon[a_1, a_2]$;
- $R^{\mathbf{B}}(h(a_1), \ldots, h(a_k))$ if, and only if, $\mathbf{A} \models \phi_R[a_1, \ldots, a_k]$;
- $h(a) = c^{\mathbf{B}}$ if, and only if, $\mathbf{A} \models \gamma_c[a]$.

Note that an interpretation $\Theta$ associates a $\tau$-structure with $\mathbf{A}$ only if $\varepsilon$ defines an equivalence relation on $\mathrm{dom}(\mathbf{A})^m$ that is a congruence with respect to the relations defined by the formulae $\phi_R$ and $\gamma_c$. In such cases, however, $\mathbf{B}$ is uniquely defined up to isomorphism and we write $\Theta(\mathbf{A}) := \mathbf{B}$. Throughout this paper, we often use interpretations where $\varepsilon$ is simply defined as the usual equality on $a_1$ and $a_2$. In these instances, we omit the explicit definition of $\varepsilon$.

The notion of interpretations is used to define logical reductions. Let $C_1$ and $C_2$ be two classes of $\sigma$- and $\tau$-structures respectively. We say that $C_1$ L*-reduces* to $C_2$ if there is an L-interpretation $\Theta$ of $\tau$ in $\sigma$, such that $\Theta(\mathbf{A}) \in C_2$ if and only if $\mathbf{A} \in C_1$, and we write $C_1 \leq_{\mathrm{L}} C_2$.

It is not difficult to show that formulas of FPC compose with FPC-reductions in the sense that, given an interpretation $\Theta$ of $\tau$ in $\sigma$ and a $\tau$-formula $\phi$, we can define a $\sigma$-formula $\phi'$ such that $\mathbf{A} \models \phi'$ if, and only if, $\Theta(\mathbf{A}) \models \phi$. Note that if $\phi$ uses $k$ variables, the composition $\phi'$ may contain up to $m \cdot k$ many variables, where $m$ is the width of $\Theta$. Likewise, interpretations themselves compose. That is, given interpretations $\Theta$ of $\tau$ in $\sigma$, and $\Sigma$ of $\sigma$ in $\rho$, we can obtain an interpretation $\Theta'$ of $\tau$ in $\rho$ by composition: $\Theta'$ consists of the functions of $\Theta$ where the relation symbols of $\sigma$ are instead replaced by the corresponding $\rho$-formulas in $\Sigma$.

**Proposition 4.** Let $C_1$ and $C_2$ be two classes of structures, such that $C_1 \leq_{\mathrm{FPC}} C_2$ by some FPC-reduction $\Theta$. Furthermore, let $\theta : \mathbb{N} \to \mathbb{N}$ be defined as $\theta(n) = \max_{\mathbf{A} \in C_1; |\mathbf{A}| \leq n} |\Theta(\mathbf{A})|$. Then $\nu_{C_1}(n) \in O(\nu_{C_2}(\theta(n)))$.

*Proof.* Given any structure $\mathbf{A}$ (in the vocabulary of $C_1$) of size $n$, the corresponding structure $\Theta(\mathbf{A})$ has size at most $\theta(n)$. Let $k := \nu_{C_2}(\theta(n))$, then there is a formula $\phi$ in $C^k$ for which $\Theta(\mathbf{A}) \models \phi \Leftrightarrow \Theta(\mathbf{A}) \in C_2$. By composing $\phi$ with $\Theta$, we obtain a formula $\phi'$ in $C^{mk}$ that satisfies $\mathbf{A} \models \phi' \Leftrightarrow \mathbf{A} \in C_1$, where $m$ is the width of $\Theta$. This constant factor is accounted for in the $O$-notation. □

*3) Representation:* In order to discuss definability of constraint satisfaction and optimization problems, we need to fix a representation of instances of these problems as relational structures. We broadly adapt the representations used in [4].

**Numbers and Vectors.** We represent an integer $z$ as a relational structure in the following way. Let $z = s \cdot x$, with $s \in \{-1, 1\}$ being the sign of $z$, and $x \in \mathbb{N}$, and let $b \geq \lceil \log_2(x) \rceil$. We represent $z$ as the structure $\mathbf{z}$ with universe $[b]$ over the vocabulary $\tau_{\mathbb{Z}} = \{X, S, <\}$, where $<$ is interpreted

the usual linear order on $[b]$; $S^{\mathbf{z}}$ is a unary relation where $S^{\mathbf{z}} = \emptyset$ indicates that $s = 1$, and $s = -1$ otherwise; and $X^{\mathbf{z}}$ is a unary relation that encodes the bit representation of $x$, i.e. $X^{\mathbf{z}} = \{k \in [b] \mid \mathrm{BIT}(x, k) = 1\}$. In a similar vein, we represent a rational number $q = s \cdot \frac{x}{d}$ by a structure $\mathbf{q}$ over the domain $\tau_{\mathbb{Q}} = \{X, D, S, <\}$, where the additional relation $D^{\mathbf{q}}$ encodes the binary representation of the denominator $d$ in the same way as before.

In order to represent vectors and matrices over integers or rationals, we have multi-sorted universes. Let $T$ be a non-empty set, and let $v$ be a vector of integers indexed by $T$. We represent $v$ as a structure $\mathbf{v}$ with a two-sorted universe with an index sort $T$, and bit sort $[b]$, where $b \geq \lceil \log_2(|m|) \rceil$, $m = \max_{t \in T} v_t$, over the vocabulary $(X, D, S, <)$. Now, the relation $S$ is of arity 2, and $S^{\mathbf{v}}(t, \cdot)$ encodes the sign of the integer $v_t$ for $t \in T$. Similarly, $X$ is a binary relation interpreted as $X^{\mathbf{v}} = \{(t, k) \in T \times [b] \mid \mathrm{BIT}(v_t, k) = 1\}$. In order to represent matrices $M \in \mathbb{Z}^{T_1 \times T_2}$, we have three-sorted universes with two sorts of index sets $T_1$ and $T_2$, or simply a single index set that consists of pairs. The matrix $M$ is then represented as a structure $\mathbf{M}$ in the same way as a vector, only $X^{\mathbf{M}}$ is now a ternary relation, instead of binary. Tensors of even higher dimensions, such as matrix-valued vectors, are represented in a similar fashion, simply by increasing the number of index sets, as well as the arity of the relation $X$. The generalization to rationals carries over from the numbers case. We write $\tau_{\text{vec}}$ to denote the vocabulary for vectors over $\mathbb{Q}$, $\tau_{\text{mat}}$ for the vocabulary for matrices over $\mathbb{Q}$, and $\tau_{\text{tens}}$ for the vocabulary for matrix-valued vectors.

**Linear and Semidefinite Programs.** We represent linear or semidefinite programs in their respective standard forms in the following way. An instance of a linear program in standard form is given by a constraint matrix $A \in \mathbb{Q}^{M \times V}$, and vectors $b \in \mathbb{Q}^M, c \in \mathbb{Q}^V$. Hence, we represent it as a structure over the vocabulary $\tau_{\text{LP}} = \tau_{\text{mat}} \dot{\cup} \tau_{\text{vec}} \dot{\cup} \tau_{\text{vec}}$. That is, a linear program is represented as a structure with a three-sorted universe with the two index sets $V$ and $M$, and a bit sort, and we have triples of relations $(X_A, D_A, S_A)$, $(X_b, D_b, S_b)$, and $(X_c, D_c, S_c)$ that encode the entries of $A$, $b$, and $c$ respectively, along with the usual $<$ relation on the bit sort. Note that $X_A$ here is a ternary relation, and $X_b$ and $X_c$ are binary relations.

Likewise, a semidefinite program in conic standard form is specified by a matrix-valued vector $\mathcal{A} \in \mathbb{Q}^{M \times (V \times V)}$, an objective matrix $C \in \mathbb{Q}^{V \times V}$, and a vector $b \in \mathbb{Q}^M$. This is represented as a structure over $\tau_{\text{SDP}} = \tau_{\text{tens}} \dot{\cup} \tau_{\text{mat}} \dot{\cup} \tau_{\text{vec}}$. Here, the universe is again three-sorted, with the two index sets $V \times V$ and $M$ plus a bit sort. Note that $X_{\mathcal{A}}$ is a 4-ary relation, and $X_C$ and $X_b$ are ternary and binary relations.

Sometimes it is more convenient to consider an SDP in inequality standard form, which is specified by a matrix $Z \in \mathbb{Q}^{M \times M}$, a matrix-valued vector $\mathcal{Y} \in \mathbb{Q}^{V \times (M \times M)}$ and an objective vector $c \in \mathbb{Q}^V$. Note that the vocabulary for both representations are the same, and that the conversion between the two standard forms can be expressed as an FPC interpretation, as it only involves simple substitution and rearranging of variables.

**CSPs.** For a fixed domain $D$, and a constraint language $\Gamma$, we can represent an instance of $\mathrm{CSP}(D, \Gamma)$ in a natural way. Namely, the vocabulary $\tau_{\mathrm{CSP}(\Gamma)}$ consists of all relations in $\Gamma$. An instance $I = (V, C)$ is then represented as the $\tau_\Gamma$-structure $\mathbf{I} = (V, (R^{\mathbf{I}})_{R \in \Gamma})$, where the universe is set to the set of variables $V$, and $s \in R^{\mathbf{I}}$ if there is a constraint $c = (s, R)$ in the constraint set $C$.

For the finite-valued variant, we define the vocabulary $\tau_{\mathrm{VCSP}(\Gamma)}$ as $\tau_{\mathrm{VCSP}(\Gamma)} = \{(R_f)_{f \in \Gamma}, W, <\}$. An instance $I = (V, C)$ is then represented as a structure $\mathbf{I}$ with a three-sorted universe: A sort for variables $V$; a sort of constraints $C$; and a bit sort $[b]$ for some sufficiently large $b$. The relation $R_f^{\mathbf{I}} \subseteq V^{\mathrm{ar}(f)} \times C$ then contains a tuple $(s, c)$ if $C$ contains a constraint of the form $(s, f, w)$. Similarly, the relation $W^{\mathbf{I}} \subseteq C \times [b]$ encodes the weight of each constraint $c = (s, f, w)$ in the relational representation of integers, i.e. $W^I(c, \cdot) = \{k \in [b] \mid \mathrm{BIT}(w, k) = 1\}$. Finally, $<$ is again just interpreted as the usual natural order on $[b]$.

Throughout the paper we use the symbols $\tau_{\text{vec}}$, $\tau_{\text{mat}}$, $\tau_{\text{LP}}$, $\tau_{\text{SDP}}$, and $\tau_\Gamma$ to refer to the vocabulary of vectors, matrices, linear programs, semidefinite programs, and (valued) constraint satisfaction problems over $\Gamma$, respectively.

We can now state the definability result from [3], to the effect that there is an FPC interpretation that can define solutions to linear programs. We show a generalization of the result to semidefinite programs in Theorem 6.

**Theorem 5** (Theorem 11, [3])**.** Let instances of a linear program be given by $(A, b, c)$ with $A \in \mathbb{Q}^{M \times V}$, $b \in \mathbb{Q}^M$, and $c \in \mathbb{Q}^V$. Its feasible region is denoted by $\mathcal{F}_{A,b}$. Let $\mathbf{I}$ be the relational representation of this LP. Then, there is an FPC-interpretation $\Phi$ of $\tau_{\mathbb{Q}} \dot{\cup} \tau_{\text{vec}}$ in $\tau_{LP}$ such that $\Phi(\mathbf{I})$ defines a relational representation of $(f, v)$, with $f \in \mathbb{Q}$, $v \in \mathbb{Q}^V$, such that

- $f = 1$ if, and only if, $\max_{x \in \mathcal{F}_{A,b}} c^T x$ is unbounded;
- If $\mathcal{F}_{A,b} \neq \emptyset$ then $v \in \mathcal{F}_{A,b}$;
- and $f = 0, v = \mathrm{argmax}_{x \in \mathcal{F}_{A,b}} c^T x$ otherwise.

## III. Main result

Our first main contribution is a generalization of Theorem 5 to well-bounded semi-definite programs.

**Theorem 6.** Let instances of an SDP be given by $(A, b, C)$ and an error parameter $\delta$, with $A \in \mathbb{Q}^{M \times (V \times V)}$, $b \in \mathbb{Q}^M$, $C \in \mathbb{Q}^{V \times V}$, and $\delta > 0$. Its feasible region is denoted by $\mathcal{F}_{A,b}$. Let $\mathbf{I}$ be the relational representation of this SDP. Then, there is an FPC-interpretation $\Phi$ of $\tau_{\text{mat}}$ in $\tau_{\text{SDP}} \dot{\cup} \tau_{\mathbb{Q}}$ such that $\Phi(\mathbf{I})$ defines a relational representation of $X \in \mathbb{Q}^{V \times V}$, such that

- if $\mathcal{F}_{A,b}$ is non-empty and bounded, then $X$ is a $\delta$-close and $\delta$-maximal solution;
- otherwise $X$ is unspecified.

Furthermore, we apply the above result to SDPs obtained as the Lasserre relaxation of VCSP problems. We establish

a dichotomy: Either $\mathrm{VCSP}(D,\Gamma)$ is tractable, and every instance is captured by its basic linear programming relaxation; or there are instances that are only captured after $\Omega(n)$ levels of the Lasserre hierarchy, where $n$ is the size of the instance. As a special case, we obtain the same dichotomy for the class of MAXCSP problems.

Recall that we use $L_C(n)$ to denote the minimum number $t$, such that the Lasserre relaxation at level $t$ suffices to capture all instances of $C$ of size at most $n$, and we use $\nu_C$ to denote the counting width of a class $C$. For the sake of legibility, we use $L_\Gamma$ as a shorthand for $L_{\mathrm{VCSP}(D,\Gamma)}$ and $\nu_\Gamma$ as shorthand for $\nu_{\mathrm{VCSP}(D,\Gamma)}$.

**Theorem 7.** For any $\mathrm{VCSP}(D,\Gamma)$ either every instance $I$ is solved by $\mathrm{BLP}(I)$; or $L_\Gamma(n) \in \Omega(n)$.

**Corollary 8.** For any $\mathrm{MAXCSP}(D,\Gamma)$ either every instance $I$ is solved by $\mathrm{BLP}(I)$; or $L_{\mathrm{MAXCSP}(D,\Gamma)}(n) \in \Omega(n)$.

The key technical lemma for proving this dichotomy is obtained by using Theorem 6 to bound the level of the Lasserre hierarchy required to capture all instances of $\mathrm{VCSP}(D,\Gamma)$ by the counting width of its class of decision problems.

**Lemma 9.** For any $\mathrm{VCSP}(D,\Gamma)$, $L_\Gamma \in \Omega(\nu_\Gamma)$.

In addition, we prove a counting width dichotomy for $\mathrm{VCSP}(D,\Gamma)$. This is achieved by connecting the results of [9] and [5] to show a linear lower bound of $\nu_\Gamma$ for the hard cases of $\mathrm{VCSP}(D,\Gamma)$.

**Lemma 10.** If there are instances $I$ of $\mathrm{VCSP}(D,\Gamma)$ that are not solved by $\mathrm{BLP}(I)$, then $\nu_\Gamma(n) \in \Omega(n)$.

Given the above two lemmas, we obtain as a direct consequence Theorem 7. We devote the remaining sections to proving Theorem 6 and Lemmas 9 and 10.

## IV. Expressing semidefinite programs

We now turn to prove Theorem 6, which states that the weak optimization problem for explicitly given SDPs is expressible in FPC. Our result generalizes the previous work by Anderson et al. [3], [4] that established the FPC-definability of linear programming, as stated in Theorem 5. In the same fashion as in their work, the central piece of the proof is a formulation of the ellipsoid method in FPC. In the present paper we extend their methods to cope with feasible regions of SDPs, instead of being limited to polyhedra. In particular, we show here that the ellipsoid method for semi-definite regions, as well as a suitable weak separation oracle for SDPs can both be formalized in FPC.

### A. Separation Oracle

Similar to the work in [3], our proof strategy is to use an FPC-formulation of the ellipsoid method to reduce the optimization problem to the separation problem. Therefore in order to prove Theorem 6 we show first that we can express a separation oracle for SDPs in FPC.

**Lemma 11.** There is an FPC-interpretation $\Phi$ of $\tau_{\mathrm{mat}}$ in $\tau_{\mathrm{SDP}} \dot\cup \tau_{\mathbb{Q}}$ that does the following:

An instance of the separation problem is given by $(A,b,Y)$, and an error parameter $\delta$, with $A \in \mathbb{Q}^{M\times(V\times V)}$, $b \in \mathbb{Q}^M$, $Y \in \mathbb{Q}^{V\times V}$, and $\delta > 0$. Let $\mathbf{I}$ be the relational representation of this instance.

Then, $\Phi(\mathbf{I})$ defines a relational representation of $S \in \mathbb{Q}^{V\times V}$, such that

- if $\mathcal{F}_{A,b}$ is empty or bounded, then there is no specification on $S$;
- otherwise if $S = 0$, then $Y$ is $\delta$-close to $\mathcal{F}_{A,b}$;
- otherwise $\langle S, Y\rangle + \delta > \max\{\langle S, X\rangle \mid X \in \mathcal{F}_{\mathcal{A},b}\}$.

Algorithm 1 describes a method to solve the weak separation problem for SDPs. It follows the classical separation algorithm that attempts to find an eigenvector of a negative eigenvalue of the input matrix. We introduce a couple of modifications which allow this procedure to be formalized in the logic FPC.

**Algorithm 1** Weak separation oracle for semidefinite programs

**Input:** $\mathcal{A} = \{A_1,\dots,A_m \in \mathbb{Q}^{V\times V}\}$, $b \in \mathbb{Q}^m$, $Y \in \mathbb{Q}^{V\times V}$, $\delta \in \mathbb{Q}$ such that $\delta > 0$.
**Output:** Solves weak separation problem on $\mathcal{F}_{\mathcal{A},b}$, $Y$, and $\delta$.

1: **function** SEPARATION($\mathcal{A}$,$b$,$Y$,$\delta$):
2: $\quad\mathcal{V} \leftarrow \{A_i \in \mathcal{A} \mid \langle A_i, Y\rangle > b_i\}$
3: $\quad$**if** $\mathcal{V}$ is non-empty **then**
4: $\quad\quad v \leftarrow \sum_{A_i\in\mathcal{V}} A_i$
5: $\quad\quad$**return** $\frac{v}{\|v\|}$
6: $\quad$Approximate eigenvalues $\{\tilde\lambda_1,\dots,\tilde\lambda_{|V|}\}$ of $Y$ up to precision $\frac{\delta}{4}$
7: $\quad$**if** there is $\tilde\lambda$ with $\tilde\lambda_i < \frac{\delta}{2}$ **then**
8: $\quad\quad v \leftarrow$ Vector satisfying $\|(Y - \tilde\lambda_i I)v\| < \frac{\delta}{2}$ and $\|v\|_\infty = 1$
9: $\quad\quad$**return** $(-1)/\|vv^T\| \cdot vv^T$
10: $\quad$**return** ACCEPT

The translation of Algorithm 1 into an FPC-interpretation uses some known tools from descriptive complexity. First, we note that the basic vector and matrix operations, such as addition, multiplication, norm and even computing the characteristic polynomial can all be defined in FPC [12]. A key modification is in Line 4: In the original algorithm, we have to choose a violated constraint from an unordered set of constraints. In FPC, we have no mechanism to express this choice. However, we can employ the same technique as in [3]: The explicit choice of a constraint can be avoided by summing all violated constraints, since by linearity, the sum of violated constraints is again a violated constraint, which in turn is expressible in FPC.

In Line 6, we compute the eigenvalues of the input matrix $Y$ up to a given precision $\delta/4$. This is possible in FPC since it is powerful enough to define the coefficients of the characteristic polynomial of definable matrices (see [12]).

**Proposition 12.** There is an FPC interpretation of $\tau_{\mathbb{Q}}$ in $\tau_{\text{mat}} \dot{\cup} \tau_{\mathbb{Q}}$ that for a given a matrix $A \in \mathbb{Q}^{V \times V}$ and a value $\delta \in \mathbb{Q}$ (in their relational representation) defines the value of the smallest eigenvalue of $A$ up to a precision of $\delta$.

*Proof.* Holm [12] establishes that there is an interpretation in FPC by which we can obtain from $A$ the coefficients $\alpha_1, \ldots, \alpha_n$ of the characteristic polynomial $p(x) = \det(xI - A) = x^n - \alpha_1 x^{n-1} + \ldots + (-1)^n \alpha_n$. Note that the coefficients are linearly ordered (by the power of their corresponding monomial), and hence by the Immerman-Vardi theorem, any polynomial time computable property can be defined in FPC, such as computing the smallest eigenvalue up to a precision $\delta$. □

Furthermore, the exact calculation of the eigenvector corresponding to a negative eigenvalue has been replaced by a linear optimization step in Line 8. In general, the eigenvectors corresponding to some eigenvalue $\lambda$ are not uniquely defined. Not only can we scale eigenvectors by an arbitrary amount, in the case of an eigenvalue of higher multiplicity we have to choose a representative from a whole multidimensional eigenspace. To avoid this choice, we reformulate the problem as a linear program and rely on Theorem 5 to express this step in FPC. In particular, finding a vector $v$ minimizing the $L^1$-norm $\min \|(Y - \tilde{\lambda}_i I)v\|_1$, subject to $\|v\|_\infty = 1$, can be expressed as a LP. For a fixed dimension $n$, the $L^1$- and $L^2$-norms only differ by a constant, and can be adjusted by the choice of $\delta$. Hence, by Theorem 5, a solution $v$ can be expressed in FPC.

The correctness of the algorithm then follows from some basic calculations: Consider an input query $(\mathcal{A}, b, Y, \delta)$. If the algorithm accepts this input, then $Y$ violated none of the inequalities $\langle A_i, Y \rangle \leq b_i$, and all eigenvalues of $Y$ are non-negative, and we can conclude that $Y$ is in fact inside the feasible region $\mathcal{F}_{\mathcal{A},b}$.

If the algorithm does not accept, then either some inequality $\langle A_i, Y \rangle \leq b_i$ is violated, in which case the algorithm produces a correct separating hyperplane; or some approximated eigenvalue $\tilde{\lambda}$ is smaller than $\delta/2$. In the latter case, the linear optimization step looks for a vector $v$ such that $\|v\| = 1$ and $\|(Y - \tilde{\lambda})v\| < \delta/2$. Note that such a vector always exists. Let $\lambda$ be the actual eigenvalue, such that $\tilde{\lambda} = \lambda + \epsilon$ for some error $\epsilon$ with $|\epsilon| \leq \delta/4$. We have

$$(Y - \tilde{\lambda} I)v = (Y - \lambda I - \epsilon I)v.$$

Setting $v$ to some eigenvector corresponding to $\lambda$ with $\|v\| = 1$, we get

$$\|(Y - \tilde{\lambda})v\| = \|(Y - \lambda I - \epsilon I)v\| = |\epsilon| < \delta/2.$$

Finally, given such a vector $v$, the normal of a weakly separating plane is given by

$$S := -1/\|vv^T\| \cdot vv^T.$$

To verify this, let $\epsilon := Yv - \tilde{\lambda}v$. Then it follows

$$\begin{aligned} \langle S, Y \rangle &= -1/\|vv^T\| \cdot (v^T Y v) \\ &= -1/\|vv^T\| \cdot (v^T(\tilde{\lambda}v + \epsilon)) \\ &= -1/\|vv^T\|(\tilde{\lambda} + v^T\epsilon) < \delta. \end{aligned}$$

This shows that we can define a weak separation oracle for SDPs in FPC.

### B. Reducing Optimization to Separation

In this section we construct a FPC-reduction from the weak optimization problem to the weak separation problem for SDPs, by formalizing the ellipsoid method in logic.

**Lemma 13.** If there is a FPC-interpretation expressing the weak separation problem for the feasible region of a given SDP, then there is a FPC-interpretation which expresses the weak semidefinite optimization problem.

The above result is known from [3], [4] for the case of LPs instead of SDPs. The overall algorithm here follows a similar line as the one for linear programs, however with a couple of key extensions necessary to cope with the feasible regions of SDPs.

The main idea behind the construction is to maintain and increasingly refine an order relation on the set of variables. The order is maintained through a set of equivalance classes containing currently incomparable elements, and a linear order on these equivalence classes. Since the equivalence classes are linearly ordered, we can use the Immerman-Vardi theorem to define the ellipsoid method. Technically, this is achieved by defining a series of increasingly fine equivalence relations on the variable set $V$, specified by so-called *foldings* that we formalize below. Intuitively, these equivalence relations are obtained in the following way: In the beginning, all elements of $V$ reside in the same equivalence class. However, there may be some inputs on which the separation oracle returns a vector $d$ with different values $d_u$ and $d_v$ for $u, v \in V$, which distinguishes the two elements $u$ and $v$, say $d_u < d_v$. In subsequent iterations, $u$ and $v$ are put in different equivalence classes, say $\mathcal{E}_u$ and $\mathcal{E}_v$, and we define an order on the classes to reflect the revealed order of $d_u$ and $d_v$, i.e. $\mathcal{E}_u < \mathcal{E}_v$. This process is repeated until we obtain a sufficiently refined ordered partition of $V$. Once such a partition of $V$ is obtained, we are able to apply the ellipsoid method on the set of equivalence classes. As the equivalence classes themselves are ordered, this is definable in FPC by Immerman-Vardi. Finally, it can be shown that any (weakly) optimal solution to a sufficiently folded SDP (where the variables are equivalence classes) is also (weakly) optimal for the original input.

There are a couple of key modifications to be made to the algorithm from [3]. Namely, we show (1) that the folding operation preserves the positive semidefiniteness of sets, and (2) how to cope with the additional parameters introduced by the weak versions of the separation and optimization problems.

We start by defining the notion of *folding*.

**Definition 9.** Let $V$ be a non-empty set. For $k \leq |V|$, we call a surjective mapping $\sigma : V \to [k]$ an *index map*. Furthermore, for each $i \in [k]$ we define $V_i := \{v \in V \mid \sigma(v) = i\}$.

For a vector $x \in \mathbb{Q}^V$, the *almost-folded* vector $[x]^{\tilde{\sigma}} \in \mathbb{Q}^k$ is given by

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

Its *folded* vector $[x]^{\sigma} \in \mathbb{Q}^k$ is given by

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

For a vector $\hat{x} \in \mathbb{Q}^k$, its *unfolded* vector $[\hat{x}]^{-\sigma} \in \mathbb{Q}^V$ is given by

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

For a given index map $\sigma$ and a vector $x \in \mathbb{Q}^V$, we say $x$ *agrees* with $\sigma$ when for all $u, v \in V$ $\sigma(u) = \sigma(v)$ implies $x_u = x_v$. The notion also extends in a natural way to sets $S \subseteq \mathbb{Q}^V$, simply by defining the folded set $[S]^{\sigma} := \{[s]^{\sigma} \mid s \in S\}$. This can be seen as a projection of $S$ into the (ordered) $k$-dimensional space $\mathbb{Q}^k$. When talking about matrices, that is, when the variable set $V$ consists of pairs from some product set $V' \times V'$, we implicitly also require that $\sigma$ is *consistent*. Namely we require that an index map $\sigma : V' \times V' \to [k] \times [k]$ is defined by an underlying index map $\tau : V' \to [k]$, with $\sigma(u, v) = (\tau(u), \tau(v))$.

There are some useful properties of the folding operation that allow us to infer some information about the geometry of a folded set from its original.

**Proposition 14.** Let $\sigma : V \to [k]$ be an index map, $x, c$ vectors in $\mathbb{Q}^V$, where $c$ agrees with $\sigma$. Then,

$$\langle c, [[x]^{\sigma}]^{-\sigma} \rangle = \langle c, x \rangle = \langle [c]^{\tilde{\sigma}}, [x]^{\sigma} \rangle.$$

*Proof.* See Proposition 12 from [3]. □

**Proposition 15.** Let $\mathcal{P} \subseteq \mathbb{Q}^V$ be a polytope in $\mathbb{Q}^V$ and let $\sigma : V \to [k]$ be an index map. Then the folded set $[\mathcal{P}]^{\sigma}$ is a polytope in $\mathbb{Q}^k$.

*Proof.* See Proposition 13 from [3]. □

**Proposition 16.** Let $X \in \mathbb{Q}^{V \times V}$ be a positive semidefinite matrix, and let $\sigma : V \times V \to [k] \times [k]$ be consistent. Then $[X]^{\sigma}$ is also a positive semidefinite matrix in $\mathbb{Q}^{k \times k}$.

*Proof.* Since $\sigma$ is consistent, we have a map $\tau : V \to [k]$ with $\sigma(u, v) = (\tau(u), \tau(v))$. Furthermore, since $X$ is positive semidefinite, there exists vectors $g_v \in \mathbb{Q}^l$ for all $v \in V$ and some $l \geq 1$ such that $X_{u,v} = \langle g_u, g_v \rangle$ for all $u, v \in V$ (i.e. the Gram representation of $X$). Let us now define vectors $g_1^{\tau}, \ldots, g_k^{\tau}$ by

$$g_i^{\tau} = \frac{1}{|V_i|} \sum_{v \in V_i} g_v,$$

where $V_i := \{v \in V \mid \tau(v) = i\}$. The vectors obtained in this way now form a Gram representation of the folded matrix $[X]^{\sigma}$, since

$$\begin{aligned} [X]^{\sigma}_{i,j} &= \frac{1}{|V_{i,j}|} \sum_{(u,v) \in V_{i,j}} X_{u,v} \\ &= \frac{1}{|V_i||V_j|} \sum_{u \in V_i} \sum_{v \in V_j} \langle g_u, g_v \rangle \\ &= \langle g_i^{\tau}, g_j^{\tau} \rangle. \end{aligned}$$

As the existence of a Gram representation implies positive semidefiniteness, this proves our claim. □

Since the feasible region of an SDP is the intersection of a polytope with the positive semidefinite cone, Propositions 15 and 16 show that the result of folding the feasible region of an SDP is again the feasible region of an SDP. Next we show that a weak separation oracle of the original set either serves as an oracle for the folded set, or produces some vector that does not agree with the index map of the folding.

**Proposition 17.** Let $\mathcal{F} \subseteq \mathbb{Q}^V$ be a convex set, and let $\sigma : V \to [k]$ be an index map. Given a vector $x \in \mathbb{Q}^V$ that is $\delta$-close to $\mathcal{F}$ for some $\delta \geq 0$, the folded vector $[x]^{\sigma} \in \mathbb{Q}^k$ is also $\delta$-close to the folded set $[\mathcal{F}]^{\sigma}$.

*Proof.* Let $x = f + d$, where $f \in \mathcal{F}$ is some point in the set $\mathcal{F}$, and $d$ the difference vector with $\|d\| \leq \delta$. By the definition of folding, we then have $[x]^{\sigma} = [f]^{\sigma} + [d]^{\sigma}$, where $[f]^{\sigma}$ is now a point in the folded set $[\mathcal{F}]^{\sigma}$. We can bound the norm of $[d]^{\sigma}$ by

$$\begin{aligned} \|[d]^{\sigma}\| &= \sqrt{\sum_{i \in [k]} \left( \frac{1}{|V_i|} \sum_{v \in V_i} d_v \right)^2} \\ &\leq \sqrt{\sum_{i \in [k]} (\max_{v \in V_i} d_v)^2} \\ &\leq \sqrt{\sum_{v \in V} d_v^2} = \|d\|. \end{aligned}$$

Since $\|d\| \leq \delta$, we have $\|[d]^{\sigma}\| \leq \delta$. □

**Proposition 18.** Let $\mathcal{F} \subseteq \mathbb{Q}^V$ be a convex set, and let $\sigma : V \to [k]$ be an index map. Given vectors $s, y \in \mathbb{Q}^V$ where $s$ agrees with $\sigma$, and $\langle s, y \rangle + \delta > \max\{\langle s, x \rangle \mid x \in \mathcal{F}\}$, it holds that $\langle [s]^{\sigma}, [y]^{\sigma} \rangle + \delta > \max\{\langle [s]^{\sigma}, x \rangle \mid x \in [\mathcal{F}]^{\sigma}\}$.

*Proof.* Let $x \in \mathcal{F}$ be a point in $\mathcal{F}$ such that $\langle s, x \rangle$ is maximal. It follows from Proposition 14 that $[x]^{\sigma}$ is also a maximal point in $[\mathcal{F}]^{\sigma}$ with respect to $\langle [s]^{\sigma}, [x]^{\sigma} \rangle$. We then have

$$\begin{aligned} \langle [s]^{\sigma}, [x]^{\sigma} \rangle - \langle [s]^{\sigma}, [y]^{\sigma} \rangle &= \langle [s]^{\sigma}, [x - y]^{\sigma} \rangle \\ &\leq \langle s, x - y \rangle \\ &\leq \langle s, x \rangle - \langle s, y \rangle < \delta. \end{aligned}$$

For the first equality we use the fact that $[x - y]^{\sigma} = [x]^{\sigma} - [y]^{\sigma}$, and for the first inequality we use Proposition 14 to get $\langle [s]^{\sigma}, [x]^{\sigma} \rangle \leq \langle [s]^{\tilde{\sigma}}, [x] \rangle = \langle s, x \rangle$. □

Assume we are given a convex set $\mathcal{F} \subseteq \mathbb{Q}^V$ by means of a corresponding weak separation oracle, and some index map $\sigma : V \to [k]$. Proposition 17 ensures that whenever the oracle accepts some input $(y, \delta)$, then the folded vector $[y]^\sigma$ is also $\delta$-close to the folded set $[\mathcal{F}]^\sigma$. Likewise, by Proposition 18 we know that whenever the oracle for $\mathcal{F}$ outputs a separation normal $s$ that agrees with $\sigma$, then the folded vector $[s]^\sigma$ is also a $\delta$-weak separation normal that separates $[y]^\sigma$ from $[\mathcal{F}]^\sigma$.

This leads us to a simple algorithm for the weak separation problem for $[\mathcal{F}]^\sigma$: On some input $\hat{y} \in \mathbb{Q}^k$ and $\delta \geq 0$, we simply give the input $[\hat{y}]^{-\sigma}$ and $\delta$ to the oracle for $\mathcal{F}$. If the oracle accepts, or returns a separation normal that agrees with $\sigma$, we are done. In the other cases, the oracle returns a separation normal that does not agree with our current index map $\sigma$, in which case we can use that output to further refine the underlying equivalence relation. After at most $|V|$ such refinements we obtain a correct weak separation oracle for $[\mathcal{F}]^\sigma$.

The overall algorithm works as described in [3], and we substitute their blackbox for a separation oracle by the one we obtain from SectionIV-A. Together with the result from Section IV-A that the weak separation oracle for the feasible region of an explicitly given SDP can be defined in FPC, this now almost establishes Theorem 6. A small technicality still remains: We assume as a condition in Theorem 6 that the feasible region of the given SDP instance is bounded and non-empty, while the original reduction given in Theorem 1 assumes the region to be bounded and full-dimensional. However, any non-empty region can be turned into a full-dimensional one in a simple pre-processing step, by adding some slack to the constraints.

This concludes the proof of Theorem 6. Note that the conditions on the feasible region of the definable SDP instances are readily satisfied for instance by those arising from finite-valued CSPs: The variables only range in $[0, 1]$, and there always exists a feasible solution. In fact, any (even non-optimal) assignment in the VCSP gives rise to a feasible solution of the 0–1 LP instance.

## V. Lasserre lower bounds

We now apply the definability result on SDPs obtained in Theorem 6 to prove Lemma 9.

The following proposition allows us to translate approximate solutions to exact ones. It quantifies the quality of approximation needed so that we can obtain the exact optimum of the original 0–1 problem by rounding an approximate optimum of its Lasserre SDP.

**Proposition 19.** Let $I = (A, b, c)$ be a 0–1 linear program whose optimal solution is integral. Its feasible region is given by $\mathcal{K} = \{x \in \mathbb{Q}^V \mid Ax \geq b\} \cap \{0, 1\}^V$ with an objective vector $c \in \mathbb{Q}^V$. Furthermore let $\mathrm{Las}_t^\pi(\mathcal{K}) = \mathcal{K}^*$ for some $t$, and let $s \in \mathbb{Q}$ be the value of a $1/(4 \max\{1, \|c\|\})$-close and $1/(4 \max\{1, \|c\|\})$-maximal solution to $\mathrm{Las}_t(\mathcal{K})$ under the objective $c$. Then, by rounding $s$, we obtain the exact optimal value for $I$.

*Proof.* Let $s^*$ be the exact optimal value of $\mathrm{Las}_t(\mathcal{K})$, and by assumption, also the optimal value for $I$. We argue that $|s - s^*| \leq 1/4$, and hence rounding $s$ yields $s^*$, since $s^* \in \mathbb{Z}$.

First, note that the condition that $s$ is $1/(4 \max\{1, \|c\|\})$-maximal means that $s + 1/(4 \max\{1, \|c\|\}) \geq \max_{x \in \mathcal{K}} \langle c, x \rangle = s^*$. Hence, we have the lower bound $s \geq s^* - 1/4$.

The other direction follows from the fact that $s$ is the value of a close solution, say, $s = \langle c, y \rangle$ for some $y \in \mathbb{Q}^V$. Since $y$ is $1/(4 \max\{1, \|c\|\})$-close to $\mathcal{K}^*$, it can be decomposed into $y = x + e$ where $x \in \mathcal{K}^*$ and $\|e\| \leq 1/(4 \max\{1, \|c\|\})$. The value of $y$ is then bounded by $s = \langle c, y \rangle \leq \max_{x \in \mathcal{K}} \langle c, x \rangle + \langle c, e \rangle \leq s^* + 1/4$. □

Next, we show that it is possible in FPC to define the $t$-th level of the Lasserre hierarchy for any explicitly given 0–1 program using only $O(t)$ variables.

**Lemma 20.** There is an FPC-interpretation from $\tau_{\mathrm{LP}}$ to $\tau_{\mathrm{SDP}}$ that for a given 0–1 linear program expresses the $t$-th level Lasserre hierarchy, using $O(t)$ variables.

*Proof.* Let an instance $I$ of a 0–1 program be given by a matrix $A \in \mathbb{Q}^{U \times V}$, and vectors $b \in \mathbb{Q}^U, c \in \mathbb{Q}^V$. In order to show that we can define the $t$-th level Lasserre relaxation from $I$, it suffices to show that we can define the matrices $M_t(y)$ and $S_t^u(y)$ from $I$ for any $y \in \mathbb{Q}^{\wp_{2t+1}(V)}, u \in U$. In particular, we represent $M_t(y)$ as a sequence of matrices $(\hat{M}_{t,q})_{q \in Q}$ where $Q := \wp_{2t+1}(V) \cup \{0\}$, such that $M_t(y) = \hat{M}_{t,0} + \sum_{q \in Q \setminus \{0\}} y_q \hat{M}_{t,q}$, and show that these matrices are definable. We represent $S_t^u(y)$ in an analogous way as $S_t^u(y) = \hat{S}_{t,0}^u + \sum_{q \in Q \setminus \{0\}} y_q \hat{S}_{t,q}^u$. Hence, here it suffices to argue that the matrices $(\hat{M}_{t,q})_{q \in Q}$ and $(\hat{S}_{t,q}^u)_{q \in Q}$ are definable from $I$ within FPC.

Observe that for the $t$-th level Lasserre relaxation the matrices $M_t(y)$ and $S_t^u(y)$ are indexed by powersets $\wp_t(V)$, and the feasible region itself lies in a vector space indexed by $\wp_{2t+1}(V)$. As we need these index sets in our interpretation, we first describe how to define the powersets $\wp_k(V)$ for some fixed $k$. Namely, we encode a set $S \in \wp_k(V)$ by a $k$-ary tuple $T \in (V \cup \{0\})^k$ that contains each of the elements in $S$ once, and where the symbol 0 fills the rest of the positions. As there are up to $k!$ many tuples encoding the same set, we additionally define an equivalence relation $\doteq_k$ on $k$-tuples that identifies two tuples if they are just permutations of each other. This can be defined by the following first order formulas.

$$\delta_{\wp_k}(x_1, \ldots, x_k) := \bigwedge_{i \in [k]} (x_i = 0 \vee x_i \in V) \bigwedge_{i,j \in [k]} (x_i = 0 \vee x_i \neq x_j),$$

$$\doteq_k(x_1, \ldots, x_k, y_1, \ldots, y_k) := \bigvee_{\pi \in Sym(k)} \bigwedge_{i \in [k]} x_i = y_{\pi(i)},$$

where $\delta_{\wp_k}$ defines the set of tuples encoding some element in $\wp_k(V)$, and $\doteq_k$ defines a binary equivalence relation between those tuples. We use $Sym(k)$ to denote the set of permutations

on $[k]$. From these definitions it is not hard to define basic set operations on the elements of $\wp_k(V)$. For instance, we can define a $4k$-ary relation $union_k$ that encodes the union of two sets $S, T \in \wp_k(V)$.

$$union_k(x,s,t) = \bigwedge_{i\in[2k]} \left( x_i = 0 \bigvee_{j\in[k]} x_i = s_j \vee x_i = t_j \right) \bigwedge_{i\in[k]} \bigvee_{j\in[2k]} s_i = x_j \bigwedge_{i\in[k]} \bigvee_{j\in[2k]} t_i = x_j,$$

where $x \in \delta_{\wp_{2k}}$, and $s, t \in \delta_{\wp_k}$. Since $union_k(x,s,t)$ simply encodes $(x = s \cup t)$, we continue using the latter more familiar notation for set operations. One point to note here is that all formulas so far are all defined using $O(k)$ many variables.

Now we can turn to the definition of the matrices $(\hat{M}_{t,q})_{q\in Q}$ and $(\hat{S}^u_{t,q})_{q\in Q}$. Each matrix $\hat{M}_{t,q}$ and $\hat{S}^u_{t,q}$ is indexed over the set $\delta_{\wp_t} \times \delta_{\wp_t}$. For $x, y \in \delta_{\wp_t}$ and $q \in \delta_{\wp_{2t+1}}$, their entries are given by

$$\hat{M}_t(q,x,y) = \begin{cases} 1 & \text{if } x \cup y = q \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\hat{S}^u_t(q,x,y) = \begin{cases} A_{u,v} & \text{if } \exists v \in V : x \cup y \cup \{v\} = q \\ -b_u & \text{if } x \cup y = q \\ 0 & \text{otherwise.} \end{cases}$$

By the above expressions $(\hat{M}_{t,q})_{q\in Q}$ and $(\hat{S}^u_{t,q})_{q\in Q}$ are definable in FPC using only $O(t)$ variables. From this, we obtain the full SDP in inequality standard form by merging the constraints $M_t(y) \succeq 0$ and $S^u_t(y) \succeq 0$ into one single constraint of the form $Z \succeq 0$.

□

**Lemma 21.** Let $D$ be a domain, and $\Gamma$ a finite-valued constraint language. There is an FPC-interpretation of constant width from $\tau_\Gamma$ to $\tau_{LP}$ that defines for a given instance $I$ of $\mathrm{VCSP}(D,\Gamma)$ its corresponding 0–1 linear program.

*Proof.* The 0–1 program that encodes a VCSP instance is given in Section II-B. It has been shown in [9] that this LP is definable in FPC. The construction there also only uses a constant number of variables. □

Finally, we are now ready to prove Lemma 9.

*Proof of Lemma 9.* For the proof we fix a domain $D$ and a finite-valued constraint language $\Gamma$. For better legibility, we write $L_\Gamma$ for $L_{\mathrm{VCSP}(D,\Gamma)}$, $\nu_\Gamma$ for $\nu_{\mathrm{VCSP}(D,\Gamma)}$, and $\tau_\Gamma$ for the vocabulary $\tau_{\mathrm{VCSP}(\Gamma)}$.

The proof idea is as follows. The argument is by contradiction. Suppose that $L_\Gamma(n) \in o(\nu_\Gamma(n))$. However, by composing the interpretations from Lemmas 21 and 20 and Theorem 6 we can define a formula $\phi$ that decides membership for the decision version of $\mathrm{VCSP}(D,\Gamma)$ for instances of size $n$ using only $o(\nu_\Gamma(n))$ many variables, which violates the assumed counting width bound of $\nu(n)$.

To be more precise, let $\Theta_t$ be the composition of the interpretations from Lemmas 21 and 20. That is, $\Theta_t$ is an interpretation of $\tau_{\mathrm{SDP}}$ in $\tau_\Gamma$ that defines for a given VCSP instance $I = (V, C, w)$ the SDP of the $t$-th level of the Lasserre relaxation of the corresponding 0–1 linear program. Note that $\Theta_t$ is of width $O(t)$.

Note that the 0–1 linear programs corresponding to VCSP instances are always feasible, bounded in the 0–1 hypercube, and their optimum is always integral.

Suppose now $L_\Gamma(n) \in o(\nu_\Gamma(n))$, i.e. every instance $I = (V, C, w)$ of $\mathrm{VCSP}(D,\Gamma)$ could be captured by some Lasserre relaxation of level $t \in o(\nu_\Gamma(|V|))$. Hence, $\Theta_t(I)$ defines a Lasserre relaxation whose optimal value is exactly the optimal value to $I$.

Then, by Theorem 6 there is an interpretation $\Sigma$ of $\tau_{\text{vec}}$ in $\tau_{\mathrm{SDP}} \dot\cup \tau_{\mathbb{Q}}$ that defines $\delta$-close and $\delta$-maximal solutions to $\Theta_t(I)$. Using Proposition 19, setting $\delta$ as $\delta = 1/4|C|$ allows us to obtain the exact optimal value for $\Theta_t(I)$ (and equivalently, for $I$) by means of rounding. Both defining the value for $\delta$ as well as the rounding can be done in FPC. Hence composing $\Theta_t$ and $\Sigma$, we obtain an interpretation $\Phi$ of width $O(t)$ that defines for a given instance of $\mathrm{VCSP}(D,\Gamma)$ its optimal value.

Finally, using $\Phi$ it is not difficult to construct an FPC-formula $\phi$ using at most $O(t)$ many variables that decides membership for the decision version of $\mathrm{VCSP}(D,\Gamma)$: For an instance $(I,t)$ we simply compare $\Phi(I)$ to $t$. Since we assumed $t \in o(\nu_\Gamma(|V|))$, $\phi$ also uses only $o(\nu_\Gamma(|V|))$ many variables. This is a contradiction to the definition of $\nu_\Gamma$. □

## VI. Counting width of finite-valued CSPs

In this section we aim to provide a proof for Lemma 10. The main pieces of the argument are known results from the literature, and we simply lay out how they together imply the claim.

We aim to show a linear lower bound for the counting width of those VCSPs that are not solved by the BLP relaxation. This aligns with the dichotomy result of Thapper and Živný (Theorem 2). That is, if $\mathrm{VCSP}(D,\Gamma)$ is not solved by the BLP relaxation, we know that MAXCUT reduces to it. Our strategy is to show that (i) MAXCUT has linear counting width; and (ii) there is a linear size FPC-reduction from MAXCUT to $\mathrm{VCSP}(D,\Gamma)$, if it is not solved by its BLP relaxation. By Proposition 4 this suffices to prove our claim.

For (i), we consider the problem 3LIN: An instance of 3LIN consists of a set of variables $V$, and two sets of equations, $E_0$ and $E_1$. Each equation in $E_0$ has the form $a \oplus b \oplus c = 0$, where $\oplus$ denotes addition modulo 2, and $a, b, c \in V$. Similarly, each equation in $E_1$ has the form $a \oplus b \oplus c = 1$. The problem is then to determine whether there is an assignment $h : V \to \{0,1\}$ such that all equations are satisfied.

**Lemma 22.** $\nu_{\mathrm{3LIN}}(n) \in \Omega(n)$.

*Proof.* In [5] Atserias et al. show a lower bound of for the counting width of the problem 3LIN that is proportional to the tree-width of the instance. More precisely, they show a construction that transforms any given graph $G = (V, E)$ with

tree-width $t$ into a pair of 3LIN instances $(I, I')$, each having $O(|V|)$ variables, such that $I$ is satisfiable, but $I'$ is not, and no $\mathrm{C}^k$ formula of at most $t$ variables distinguishes between them.

The claim then follows by picking a class of graphs that have linear tree-width. Such graphs exist, for instance in the class of 3-regular expander graphs [1]. (A similar argument using graphs with linear-size balanced separators was already present in [6]). □

As a direct consequence, we obtain that 3SAT also has linear counting width, since 3LIN can be interpreted as a special case of 3SAT.

**Lemma 23.** $\nu_{\mathrm{3SAT}}(n) \in \Omega(n)$.

We continue the reduction to MAXCUT.

**Lemma 24.** $\nu_{\mathrm{MAXCUT}}(n) \in \Omega(n)$.

*Proof.* In [9], we find an explicit construction of an FPC-reduction from 3SAT to MAXCUT. This reduction is also linear size. □

Finally, the reduction from MAXCUT to those $\mathrm{VCSP}(D, \Gamma)$ that are not solved by their BLP relaxation has already been explicitly constructed in [9]. It is not difficult to confirm that these reductions are in fact linear in size. This chain of reductions then concludes the proof of Lemma 10.

## VII. Conclusion

We have established that the problem of determining the optimal value of an explicitly given SDP can be solved in fixed-point logic with counting. As an application of this result, we examined the power of the Lasserre relaxation hierarchy in the context of finite-valued constraint satisfaction problems. Here, we established a dichotomy result, showing that *every* finite-valued CSP that is not solvable by its basic linear programming relaxation (and this includes all constraint maximization problems that are known to be NP-hard) requires a linear number of levels of the Lasserre hierarchy to solve exactly. Such linear lower bounds on the number of levels of the Lasserre hierarchy were known previously for specific CSPs. Our result shows that these are part of a sweepingly general pattern. This is established by considering the definability of semidefinite programs in logic, and using a measure of logical complexity, that we call *counting width*, to classify CSPs. This suggests some directions for further investigation.

A central motivating interest in semidefinite programming in general and Lasserre hierarchies in particular comes from their use in approximation algorithms. It would be interesting to extend our methods to show lower bounds on the levels required to approximate a solution, as well as to obtain exact solutions. A potential direction is to define a measure of counting width, not just for a class of structures $\mathcal{C}$ but based on the number of variables to separate two classes $\mathcal{C}_1$ and $\mathcal{C}_2$. We could then seek to establish lower bounds on these numbers where $\mathcal{C}_1$ is a collection of instances of a VCSP with high optimum values and $\mathcal{C}_2$ contains only instances with low optima. This would show that instances with high optima cannot be separated from those with low optima by means of a small number of levels the Lasserre hierarchy.

Definability in FPC is closely linked to *symmetric* computation (see [2], [8]). In other words, algorithms that can be translated to this logic are symmetric in a precise sense. This suggests that many of our best approximation algorithms for constraint satisfaction, such as the Lasserre semidefinite programs are encountering a "symmetry barrier". Breaking through this barrier, and coming up with algorithms that break symmetries, may be crucial to more effective approximation algorithms.

## References


[1] M. Ajtai. Recursive construction for 3-regular expanders. *Combinatorica*, 14(4):379–416, 1994.

[2] M. Anderson and A. Dawar. On symmetric circuits and fixed-point logics. *Theory of Computing Systems*, 2016.

[3] M. Anderson, A. Dawar, and B. Holm. Maximum matching and linear programming in fixed-point logic with counting. In *Proceedings of the 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 173–182, 2013.

[4] M. Anderson, A. Dawar, and B. Holm. Solving linear programs without breaking abstractions. *J. ACM*, 62(6):48:1–48:26, 2015.

[5] A. Atserias, A. Bulatov, and A. Dawar. Affine systems of equations and counting infinitary logic. *Theoretical Computer Science*, 410(18):1666 – 1683, 2009.

[6] J-Y. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[7] E. Chlamtac and M. Tulsiani. Convex relaxations and integrality gaps. In F. Miguel Anjos and B. Jean Lasserre, editors, *Handbook on Semidefinite, Conic and Polynomial Optimization*, pages 139–169. Springer US, Boston, MA, 2012.

[8] A. Dawar. The nature and power of fixed-point logic with counting. *SIGLOG News*, 2(1):8–21, 2015.

[9] A. Dawar and P. Wang. A definability dichotomy for finite valued CSPs. In *24th EACSL Annual Conference on Computer Science Logic, CSL 2015*, pages 60–77, 2015.

[10] H-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 2nd edition, 1999.

[11] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

[12] B. Holm. *Descriptive Complexity Of Linear Algebra*. PhD thesis, University of Cambridge, 2010.

[13] J. B. Lasserre. An explicit exact SDP relaxation for nonlinear 0–1 programs. In Karen Aardal and Bert Gerards, editors, *Integer Programming and Combinatorial Optimization*, volume 2081 of *Lecture Notes in Computer Science*, pages 293–303. Springer Berlin Heidelberg, 2001.

[14] L. Lovász and A. Schrijver. Cones of matrices and set-functions and 0—1 optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.

[15] M. Otto. *Bounded Variable Logics and Counting — A Study in Finite Models*, volume 9 of *Lecture Notes in Logic*. Springer-Verlag, 1997.

[16] T. Rothvoß. The Lasserre hierarchy in approximation algorithms. In *Lecture Notes for the MAPSP 2013 Tutorial*, 2013.

[17] G. Schoenebeck. Linear level lasserre lower bounds for certain k-CSPs. In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 593–602, 2008.

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

[19] J. Thapper and S. Živný. The complexity of finite-valued CSPs. In *Proceedings of the 45th ACM Symposium on the Theory of Computing*, STOC '13, pages 695–704, 2013.

[20] J. Thapper and S. Živný. The limits of SDP relaxations for general-valued CSPs. *CoRR*, abs/1612.01147, 2016.