# Complexity Column

NEIL IMMERMAN, University of Massachusetts Amherst
immerman@cs.umass.edu

In the 1980's Eric Lander, Jin-Yi Cai, Martin Fürer and I began to explore the power of fixed point logic with counting (FPC). This natural class is deeply entwined with the challenge of capturing the class of polynomial-time graph properties and determining the complexity of graph isomorphism — problems that remain open. Despite that, so much more is now known about the remarkable power of FPC. Anuj Dawar explains in the following lovely survey.

# The Nature and Power of Fixed-Point Logic with Counting

Anuj Dawar, University of Cambridge

In 1982, Neil Immerman proposed an extension of fixed-point logic by means of counting quantifiers (which we denote FPC) as a logic that might express all polynomial-time properties of unordered graphs. It was eventually proved (by Cai, Fürer and Immerman) that there are polynomial-time graph properties that are not expressible in FPC. Nonetheless, FPC is a powerful and natural fragment of the complexity class PTime. In this article, I justify this claim by reviewing three recent positive results that demonstrate the expressive power and robustness of this logic.

## 1. INTRODUCTION

Neil Immerman showed in [Immerman 1982] that fixed-point logic—FP, a formalism extending first-order logic with a mechanism for defining predicates recursively—could define exactly the polynomial-time decidable properties of finite structures provided that the formulas had access to a linear order on the elements of the structures (a result established independently by Vardi [1982]). In his paper, Immerman asks the question whether the requirement of having a linear order could be replaced by allowing instead the logic to express the cardinalities of definable sets. This leads to the definition of *Fixed-Point Logic with Counting* (FPC), the extension of first-order logic with a mechanism for iteration and a mechanism for counting. Immerman's question was motivated by two considerations. On the one hand, the requirement for a linear order imposes an extraneous condition on the structures allowing us to express properties that are not really properties of the underlying structure at all, and one would like to avoid this. On the other hand, the only known examples of structural properties that could not be expressed in FP were cardinality properties, which could be expressed once a mechanism for counting is included.

It was not long after the question was raised that it was shown, by Cai et al. [1992] that there are, indeed, polynomial-time properties of graphs that cannot be expressed in FPC. The question of whether there is a natural logic in which one can express all and only the polynomial-time (PTime) decidable properties of finite structures remains open. There have been several proposals of logics that properly extend the expressive power of FPC while remaining within PTime. These include logics with choice operators [Gire and Hoang 1998], choiceless machines [Blass et al. 1999] and logics with matrix rank operators [Dawar et al. 2009]. However, none of these has been as extensively studied as FPC (see, for instance, the monograph by Otto [1997] for an extensive discussion of the logic). FPC, though no longer a candidate for capturing PTime, seems to capture an intriguing class of problems within PTime worthy of study in its own right. That makes the logic an important focus of continuing investigation.

In this article, I take a look at three recent results about the expressive power of FPC that support the case that the FPC-definable properties form a natural and powerful fragment of PTime. First of these is the monumental result of Grohe that shows that on any class $\mathcal{C}$ of structures that excludes some graph minor (precise definitions

of these notions are given in Section 3) FPC captures all of PTime. This is the culmination of a decade of work exploring the expressive power of FPC on restricted classes of structures using the methods of structural graph theory. The second result (explored in Section 4) is one that establishes the surprising power of FPC to express linear-programming problems and uses this to settle a fifteen-year-old conjecture of Blass, Gurevich and Shelah. The third result, reviewed in Section 5, characterizes the expressive power of FPC in terms of families of circuits restricted by a natural symmetry condition. This frames the class of problems definable in FPC in the context of circuit complexity and reinforces the view that this is a natural and robust class. In this article, the aim, of course, is not to prove these results but to explain the context in which they are proved, the nature of their contribution and explore some of the ideas involved in the proofs. But, first of all, we begin by reviewing the definitions and some background of the logic FPC.

## 2. FIXED-POINT LOGIC WITH COUNTING

Fix a relational vocabulary $\sigma$, and let $\mathrm{FO}[\sigma]$ denote first-order logic over this vocabulary. By this we may mean the collection of first-order $\sigma$-formulas or we may also mean the collection of $\sigma$-classes of finite structures that are definable in first-order logic. In general, we may drop the mention of the vocabulary $\sigma$ if it is implicit and just write FO. FP denotes the extension of FO with a fixed-point operator. Fixed-point operators come in different varieties, such as the least-fixed-point operator and the the inflationary-fixed-point operator. For our purposes here, it is most convenient to just consider inflationary fixed-points. For details of the different logics one can obtain with such fixed-point operators and their expressive power, the interested reader may consult [Ebbinghaus and Flum 1999]. By Immerman [1986] and Vardi [1982], we know that on ordered structures, FP expresses exactly those properties that are decidable in polynomial-time. However, in the absence of order, simple counting properties, such as saying that the number of elements in a structure is even, are not definable.

*The Logic.* Fixed-point logic with counting (FPC) extends FP with the ability to express the cardinality of definable sets. The logic has two sorts of variables: $x_1, x_2, \ldots$ ranging over the domain elements of the structure, and $\nu_1, \nu_2, \ldots$ ranging over the non-negative integers. If we allow unrestricted quantification over non-negative integers, the logic would be powerful enough to express undecidable properties. In Immerman's original proposal, number variables were restricted to taking values in the set $\{0, \ldots, n\}$ where $n$ is the number of elements in the domain of the structure of interpretation. In the formal definition below, we adopt another convention (suggested by Grohe) of requiring quantification of number variables to be bounded by a term. In addition, we also have second order variables $X_1, X_2, \ldots$, each of which has a type which is a finite string in $\{\mathrm{element}, \mathrm{number}\}^*$. Thus, if $X$ is a variable of type $(\mathrm{element}, \mathrm{number})$, it is to be interpreted by a binary relation relating elements to numbers. The logic allows us to build up *counting terms* according to the following rule:

> If $\phi$ is a formula and $x$ is a variable, then $\#x\phi$ is a term.

The intended semantics is that $\#x\phi$ denotes the number (i.e. the non-negative integer) of elements that satisfy the formula $\phi$. The formulas of FPC are now described by the following set of rules:

— all atomic formulas of first-order logic are formulas of FPC;
— if $\tau_1$ and $\tau_2$ are counting terms (that is each one is either a number variable or a term of the form $\#x\phi$) then each of $\tau_1 < \tau_2$ and $\tau_1 = \tau_2$ is a formula;
— if $\phi$ and $\psi$ are formulas then so are $\phi \wedge \psi$, $\phi \vee \psi$ and $\neg\psi$;

— if $\phi$ is a formula, $x$ is an element variable, $\nu$ is a number variable, and $\eta$ is a counting term, then $\exists x\,\phi$ and $\exists \nu \leq \eta\,\phi$ are formulas; and
— if $X$ is a relation symbol of type $\alpha$, $\mathbf{z}$ is a tuple of variables whose sorts match the type $\alpha$ and $\mathbf{t}$ is a tuple of terms of type $\alpha$, then $[\mathbf{fp}_{X,\mathbf{z}}\phi](\mathbf{t})$ is a formula.
The intended semantics here is that the tuple of elements denoted by $\mathbf{t}$ is in the inflationary fixed-point of the operator defined by $\phi$, binding the variables in $X, \mathbf{z}$.

For details of the semantics and a lot more about the logic I refer the reader to Otto's excellent monograph [Otto 1997]. Here, I illustrate the use of this logic with some examples. In what follows, we use $\mathbf{A}$ and $\mathbf{B}$ to denote finite structures over a vocabulary $\sigma$ and $A$ and $B$ to denote their respective universes.

A standard example of the power of inductive definitions is that we can use them to say a graph is connected, a property that is not expressible in FO.

*Example* 2.1. The following formula:

$$\forall u \forall v [\mathbf{fp}_{T,xy}(x = y \vee \exists z(E(x,z) \wedge T(z,y)))](u,v)$$

is satisfied in a graph $(V, E)$ if, and only if, the graph is connected. Indeed, the least relation $T$ that satisfies the equivalence $T(x,y) \equiv x = y \vee \exists z(E(x,z) \wedge T(z,y)))$ is the reflexive and transitive closure of $E$. This is, therefore the fixed point defined by the operator $\mathbf{fp}$. The formula can now be read as saying that every pair $u, v$ is in this reflexive-transitive closure.

The standard example of a property not definable in FP is to say that the number of elements in a structure is even.

*Example* 2.2. The following sentence is satisfied in a structure $\mathbf{A}$ if, and only if, the number of elements of $\mathbf{A}$ that satisfy the formula $\phi(x)$ is even.

$$\exists \nu_1 \leq [\#x\phi] \exists \nu_2 \leq \nu_1 (\nu_1 = [\#x\phi] \wedge (\nu_2 + \nu_2 = \nu_1))$$

In particular, taking $\phi$ to be a universally true formula such as $x = x$, we get a sentence that defines evenness. Here we have used the addition symbol in the subformula $\nu_2 + \nu_2 = \nu_1$. It should be noted that this denotes a relation that is easily definable by induction on the domain of numbers.

Besides using formulas of FPC to define classes of structures, we also use it to define a new structure from a given one. An FPC-*interpretation* of a vocabulary $\tau$ in a vocabulary $\sigma$ is a sequence of $\sigma$-formulas, including a formula $\theta_R$ for each symbol $R$ of $\tau$ which, when interpreted in a $\sigma$-structure $\mathbf{A}$, yield a definition of a $\tau$-structure $\Phi\mathbf{A}$. More precisely, the universe $U$ of $\Phi\mathbf{A}$ is the set of tuples of elements (which may be tuples involving numbers as well as elements of $\mathbf{A}$) satisfying some particular formula $\theta_U$, and the interpretation of each $\tau$ symbol $R$ is given by the set of those tuples in $U$ satisfying $\theta_R$. We omit some technical details here and the interested reader may consult [Immerman 1999] where an interpretation is also called a *query* or [Grohe 2014, Chapter 2], where it is called a *transduction*.

*Finite Variable Logics.* A key tool in analysing the expressive power of FPC is to look at equivalence in a weaker logic—first-order logic with counting quantifiers and a bounded number of variables. For each natural number $i$, we have a quantifier $\exists^i$ where $\mathbf{A} \models \exists^i x\,\phi$ if, and only if, there are at least $i$ distinct elements $a \in A$ such that $\mathbf{A} \models \phi[a/x]$. While the extension of first-order logic with counting quantifiers is no more expressive than FO itself (in contrast to the situation with counting terms), the presence of these quantifiers does affect the number of variables that are necessary to express a query. Let $C^k$ denote the $k$-variable fragment of first-order logic with counting quantifiers. That is, $C^k$ consists of those formulas in which no more than $k$ variables appear, free or bound. For two structures $\mathbf{A}$ and $\mathbf{B}$, we write $\mathbf{A} \equiv^{C^k} \mathbf{B}$ to denote

that the two structures are not distinguished by any sentence of $C^k$. The link between this and FPC is the following fact, established by Immerman and Lander [1990]:

THEOREM 2.3. *For every sentence $\phi$ of* FPC*, there is a $k$ such that if $\mathbf{A} \equiv^k \mathbf{B}$, then $\mathbf{A} \models \phi$ if, and only if, $\mathbf{B} \models \phi$.*

Indeed, this theorem follows from the fact that for any formula $\phi$ of FPC, there is a $k$ so that on structures with at most $n$ elements, $\phi$ is equivalent to a formula $\theta_n$ of $C^k$ Additionally, it can be shown that the quantifier depth of $\theta_n$ is bounded by a polynomial function of $n$, but the important bound for us is that the number of variables $k$ is bounded by a constant that only depends on $\phi$.

The equivalence relations $\equiv^{C^k}$ have many different characterisations, some of which arose in contexts removed from the connection with logic, such as the Weisfeiler-Lehman family of equivalences arising in the study of graph isomorphism (see the discussion in [Cai et al. 1992] for the connection). Particularly interesting are the characterisations of $\equiv^{C^k}$ in terms of two-player games, including the counting game of Immerman and Lander [1990] and the bijection game of Hella [1992]. These provide a means of arguing that two structures $\mathbf{A}$ and $\mathbf{B}$ are $\equiv^{C^k}$-equivalent. By Theorem 2.3, this can then be used to show that some property is not definable in FPC, by showing that it is not closed under $\equiv^{C^k}$ for any fixed $k$.

*Inexpressibility Results.* The suggestion that FPC might be sufficient to express all properties in PTime arose from the intuition that it addresses the two obvious shortcomings of FO by providing a means to express inductive definitions and a means of counting. In a sense, all algorithms that are "obviously" polynomial-time can be translated into the logic. Nonetheless, an ingenious construction by Cai et al. [1992] uses games to give a polynomial-time decidable class of graphs that is not definable in FPC. More precisely, they show how to construct a sequence of pairs of graphs $(G_k, H_k)$, one for each $k \in \omega$ such that:

— for each $k$, $G_k \equiv^{C^k} H_k$;
— for each $k$, $G_k \ncong H_k$, i.e. the graphs are not isomorphic; and
— $G_k$ and $H_k$ have maximum degree 3.

It is an immediate consequence of these facts that the problem of graph isomorphism for graphs of degree 3 is not expressible in FPC. However, this problem is known to be in PTime by a result of Luks [1982].

A number of other conclusions on the limitations of the expressive power of FPC can be drawn from the result of Cai et al. Such results typically fall into one of two classes. In the first category are non-expressiblity results that follow from the result of Cai et al. by means of reductions. For instance, there is no FPC sentence that defines the graphs with a Hamiltonian cycle. This is because, by a result of Dahlhaus [1984], this problem is NP-complete under first-order reductions, and FPC is closed under such reductions. Hence, as long as we have some problem in NP that is not definable in FPC, it follows that Hamiltonicity is not definable. By an older result of Lovász and Gács [1977] we also know that satisfiability of Boolean formulas in CNF (suitably encoded as a relational structure) is NP-complete under first-order reductions and therefore this is also not definable in FPC. Interestingly, it remains an open question whether 3-SAT is NP-complete under first-order reductions.

In the second category of non-expressibility results that follow on from Cai et al. are those proved by adapting their methods. For instance, it is known that graph 3-colourability is *not* NP-complete under first-order reductions (indeed, the class of problems that reduce to it obeys a 0-1-law, see [Dawar and Grädel 2010]). Still, it has

been shown [Dawar 1998] that 3-colourability is not definable in FPC by a construction of graphs adapting that of Cai et al. More recently, Atserias et al. [2009] show that the problem of determining whether a system of linear equations (modulo 2) is solvable cannot be expressed in FPC, though this is a problem in PTime. By means of first-order reductions, they then show that this implies that a host of constraint satisfaction problems (characterised by algebraic properties) are not definable in FPC, including 3-colourability and 3-SAT.

It should be noted that in all cases mentioned, the problem is shown to be not definable in FPC by establishing the stronger statement that the problem is not invariant under $\equiv^{C^k}$ for any fixed $k$. Of course, there are problems that are invariant under $\equiv^{C^k}$ for some $k$ but still not definable in FPC, for instance, problems that are not in PTime at all. Could it be that every problem in PTime that is invariant under $\equiv^{C^k}$ for some $k$ is also in FPC? This remains an open question and I refer the interested reader to Otto [1997] for a thorough discussion.

*Capturing Results.* Alongside results establishing the limits of the expressive power of FPC, there has been a series of results showing that FPC is powerful enough to express all polynomial-time decidable properties, provided that we restrict the class of structures in some way. Of course, we have already noted above that FPC (even without the counting extension) suffices to capture PTime on ordered structures. One important line of investigation has sought to examine classes of structures based on *sparse graphs*. An early result in this vein was due to Immerman and Lander, who showed that FPC captures PTime on trees. This was generalized in two distinct directions by Grohe [1998], who showed that FPC capture PTime on planar graphs and Grohe and Mariño [1999] who show that FPC captures PTime on graphs of bounded treewidth. These results were the beginning of a long progression which leads to Grohe's theorem establishing that FPC captures PTime on any class of structures whose adjacency graphs form a proper minor-closed class. This generalizes the prevous mentioned results and that is the subject of the next section.

### 3. MINOR-CLOSED CLASSES

With any $\sigma$-structure $\mathbf{A}$, we associate a (loop-free, undirected) graph $\Gamma\mathbf{A}$, which we call the *adjaceny graph* of $\mathbf{A}$ (also known as the Gaifman graph of $\mathbf{A}$). This is the graph with vertex set $A$ and in which two vertices $u$ and $v$ are adjacent if there is some relation $R$ in $\sigma$ and some tuple $\mathbf{t} \in R^{\mathbf{A}}$ such that both $u$ and $v$ occur in $\mathbf{t}$. A very useful methodology for studying well-behaved classes of finite structures is to restrict attention to structures with adjacency graphs from some restricted graph class. This has enabled the deployment of techniques from structural graph theory in the study of finite model theory (see [Dawar 2007] for a short survey). In particular, well-behaved classes of finite structures can be defined using the graph minor relation. In the rest of this section, I talk of graphs and classes of graphs. All the results about definability and capturing PTime that are stated for a class of graphs $\mathcal{C}$ apply equally well to a class of relational structures all of whose adjacency graphs are in $\mathcal{C}$.

*Graph Minors.* We say that a graph $H$ is a *minor* of a graph $G$ and write $H \preceq G$, if $H$ can be obtained from $G$ by means of repeated applications of the operations of *(i)* deleting an edge; *(ii)* deleting a vertex, and all edges incident on it; and *(iii)* contracting an edge. Here, the last operation is one where we remove an edge $(u, v)$ and replace the two vertices $u$ and $v$ by a new vertex whose neighbours are all neighbours of $u$ and $v$ (other than $u$ and $v$ themselves). A more formal definition is that $H = (U, F)$ is a minor of $G = (V, E)$ if there is a set $U' \subseteq V$ and a surjective map $M : U' \to U$ such that

— for each $u \in U$, $M^{-1}(u)$ induces a connected subgraph of $G$; and

— for each edge $(u, v) \in F$, there is an edge in $E$ between some $x \in M^{-1}(u)$ and some $y \in M^{-1}(v)$.

Intuitively, $H$ is a minor of $G$ if the "structure" of $H$ can be found inside $G$. As a simple example, if $H$ is a triangle (denoted $K_3$), then $H$ is a minor of $G$ if, and only if, $G$ contains a cycle.

Say that a class $\mathcal{C}$ of graphs excludes $H$ as a minor if $H$ is not a minor of any graph in $\mathcal{C}$. Thus, the class of graphs that exclude $K_3$ as a minor is exactly the class of acyclic graphs. Say $\mathcal{C}$ is *minor-closed* if any minor of any graph in $\mathcal{C}$ is also in $\mathcal{C}$. It is a proper minor-closed class if it is *minor-closed* and is not the class of all graphs. Clearly, any proper minor-closed class of graphs excludes some graph as a minor. Indeed it is characterised by the set of minor-minimal graphs that it excludes. Also, any graph class that excludes some graph $H$ as a minor is included in a proper minor-closed class. In particular, it is included in the class of all graphs that do not have $H$ as a minor, which is clearly minor-closed, by transitivity of the minor relation.

A famous theorem of Wagner [1937] (building on earlier work by Kuratowski [1930]) states that a graph $G$ is planar if, and only if, neither $K_5$ (the clique on 5 vertices) nor $K_{3,3}$ (the complete bipartite graph on two sets of three vertices) is a minor of $G$. Thus, planar graphs are a proper minor-closed class that is characterised by two *forbidden minors*. The theory of graph minors was developed through a series of papers by Robertson and Seymour culminating in the proof of the graph minor theorem:

THEOREM 3.1 ([ROBERTSON AND SEYMOUR 2004]). *In any infinite collection $\{G_i \mid i \in \omega\}$ of graphs, there are $i, j$ with $G_i \preceq G_j$.*

In other words, there are no infinite anti-chains in the graph minor relation. Since the set of minor-minimal elements that are excluded from some proper minor-closed class $\mathcal{C}$ form an anti-chain, an immediate corollary to the theorem is that any such class is characterised by a finite set of forbidden minors.

COROLLARY 3.2. *For any minor-closed class $\mathcal{C}$, there is a finite collection $\mathcal{F}$ of graphs such that $G \in \mathcal{C}$* if, and only if, *$F \not\preceq G$ for all $F \in \mathcal{F}$.*

This corollary has important algorithmic consequences. In particular, since Robertson and Seymour also show [Robertson and Seymour 1995] that for every $H$, there is a cubic time algorithm that decides if a given $G$ has $H$ as a minor, it follows that every proper minor-closed class is decidable by a cubic time algorithm. Indeed, if $\mathcal{C}$ is such a class and $\mathcal{F}$ the finite collection of forbidden minors that characterize it, an algorithm for testing membership in $\mathcal{C}$ is obtained by taking a graph $G$ and checking for each $H \in \mathcal{F}$ that $H$ is not a minor of $G$.

*Capturing* PTime. We can now state the theorem of Grohe.

THEOREM 3.3 (GROHE [GROHE 2014]). FPC *captures* PTime *on any proper minor-closed class $\mathcal{C}$.*

Note that, since any class $\mathcal{C}$ that excludes a minor is included in some proper-minor closed class, it follows that FPC captures PTime on any class of structures whose adjacency graphs exclude some minor.

The full proof of Theorem 3.3 has not yet been published but appears in a monograph of more than 400 pages that is available from the author's website [Grohe 2014]. The statement of the result, and a proof of the important special case of classes of graphs $\mathcal{C}$ embeddable in a fixed surface, have been published in [Grohe 2012]. It is clearly not possible, in the confines of this column, to even sketch the proof of Theorem 3.3, but I will try to highlight some of the important ideas.

Say that a class $\mathcal{C}$ of $\sigma$-structures admits $\mathrm{FPC}$ *canonization* if there is an $\mathrm{FPC}$ interpretation that maps each structure $\mathbf{A}$ in $\mathcal{C}$ to a $\sigma \uplus \{\leq\}$-structure $\mathbf{A}^{\leq}$ whose $\sigma$-reduct is isomorphic to $\mathbf{A}$ and on which $\leq$ is interpreted as a linear order of the universe. In other words, the canonization allows us to define an ordered version of each structure in $\mathcal{C}$ from the structure itself. In particular, by the invariance properties of $\mathrm{FPC}$, a canonization has the property that two structures are isomorphic if, and only if, the canonization takes them to isomorphic *ordered* structures. Checking isomorphism of ordered structures is trivial, so an $\mathrm{FPC}$ canonization on a class $\mathcal{C}$ also gives us an $\mathrm{FPC}$-definable isomorphism test on $\mathcal{C}$. Since $\mathrm{FPC}$ captures $\mathrm{PTime}$ on ordered structures, it is not difficult to see that for any class $\mathcal{C}$ that admits $\mathrm{FPC}$ canonization, $\mathrm{FPC}$ captures $\mathrm{PTime}$ on $\mathcal{C}$. Grohe shows that for any graph $H$, there is an $\mathrm{FPC}$ interpretation that is a canonization on the class of graphs that omit $H$ as a minor.

In order to prove the result, Grohe develops a version of the structure theory that Robertson and Seymour developed in the proof of the graph minor theorem, but in a version that respects definability. An essential ingredient of Robertson and Seymour's proof of Theorem 3.1 is known as the structure theorem [Robertson and Seymour 2003]. Introducing all the elements required for a formal statement of the theorem would take us too far afield, so we will be content with an informal statement. The structure theorem essentially says that for each $H$, there is a $k$ such that a graph $G$ that excludes $H$ as a minor admits a tree decomposition in which each bag is *almost embeddable* in a surface of genus $k$. The difficulty in using this decomposition theorem to derive an $\mathrm{FPC}$ canonization for graphs excluding $H$ as a minor is that the particular tree decomposition constructed by Robertson and Seymour is not *generic*. That is to say, the construction relies on a particular presentation of the graph, is not invariant under automorphisms of the graph and so cannot be defined by formulas that necessarily respect such automorphisms. Indeed, it is not clear that any generic form of such a tree decomposition can be defined.

To get around this difficulty, Grohe defines the notion of a *treelike decomposition* of a graph $G$. This is not a tree decomposition at all. Rather, it is a directed acyclic graph decorated with *bags* that are sets of vertices of $G$. The definition includes technical conditions on connectivity and consistency of these bags that ensure that it includes within it a tree decomposition of $G$. Indeed, a treelike decomposition of $G$ can be obtained from a tree decomposition by taking the images of all bags under automorphisms of $G$. An essential step in Grohe's construction is a definable version of Robertson and Seymour's structure theorem, showing that there is an $\mathrm{FPC}$ interpretation which, on graphs that exclude $H$ as a minor, defines a treelike decomposition into bags that are almost embeddable in a fixed surface.

The definable structure theorem is arrived at by a series of steps. First, Grohe shows that there is an $\mathrm{FPC}$-definable decomposition of planar graphs into their $3$-connected components. This is lifted to graphs embeddable in an arbitrary surface, by an inductive argument on the surface. More heavy lifting is required, based on elements of Robertson and Seymour's structure theory, to obtain from this a definable treelike decomposition. The final element is that from such a definable treelike decomposition, one can show that the class of graphs excluding $H$ as a minor admits $\mathrm{FPC}$ canonization, leading to Theorem 3.3.

There are two significant consequences of this construction to note here. One is that every proper minor-closed class of graphs is itself definable by some $\mathrm{FPC}$ sentence. The other is that for any such class $\mathcal{C}$, there is a $k$ such that $\equiv^{C^k}$ is the same as isomorphism on graphs in $\mathcal{C}$. In other words, the $k$-dimensional Weisfeiler-Lehman algorithm suffices to decide graph isomorphism on $\mathcal{C}$. Of course, the exact value of $k$ depends on the finite set of excluded minors that characterise the class $\mathcal{C}$ and for many proper minor-

closed classes, we do not know this set. But, in other cases, such as planar graphs, it would be interesting to pinpoint the precise value of $k$ for which this happens.

## 4. LINEAR PROGRAMMING AND MATCHING

We noted above (see page 11) that the original intuition that led to the question of whether FPC captures PTime was that all problems that admit "obvious" polynomial-time algorithms are expressible in FPC. The last section showed a whole class of problems—the proper minor-closed classes of graphs—whose membership in PTime is far from obvious, but which nonetheless turn out to be definable. In this section, we turn to another fundamental algorithmic technique which is used in a rather non-obvious way to establish that some problems are in PTime.

Linear programming is a widely used approach to solving combinatorial optimization problems. It provides a powerful framework within which optimization problems can be represented, as well as efficient methods for solving the resulting programs. In particular, it is known since the work of [Khachiyan 1980] that there are polynomial-time algorithms that solve linear programs.

*Linear Programs as Structures.* In general, a linear programming instance consists of a set $C$ of constraints over a set $V$ of variables. Each constraint $c$ consists of a vector $a_c \in \mathbb{Q}^V$ and a rational $b_c \in \mathbb{Q}$. The *feasibility problem* is to determine for such an instance if there exists a vector $x \in \mathbb{Q}^V$ such that $a_c^T x \leq b_c \quad$ for all $c \in C$. An instance of the *optimization problem* is obtained if we have, in addition, an objective function represented by a vector $f \in \mathbb{Q}^V$. The aim is then to find an $x \in \mathbb{Q}^V$ which satisfies all constraints in $C$ and maximizes the value of $f^T x$.

We want to consider instances of linear programming as relational structures that may act as interpretations for formulas of FPC. For this, we consider structures whose universe consists of three disjoint sets: $V$, $C$ and $B$. The last of these is equipped with a linear order and may be thought of as $\{0, \ldots, t\}$, i.e. an initial segment of the natural numbers where $t$ is large enough that all the rational numbers required to represent our instance can be written down using at most $t$ bits for both numerator and denominator. We can then encode the linear programming instance through suitable relations for the numerators, denominators and signs of the values involved. For instance, we have a ternary relation $N$ so that for $c \in C$, $v \in V$ and $b \in B$, $N(c, v, b)$ holds if the $b$th bit in the numerator of $a_c(v)$ is 1. It should be stressed that while the set $B$ is ordered, there is no order on the sets $V$ and $C$. If there were, then feasibility of such linear programming instances would be definable in FPC simply by the fact that all polynomial-time decidable properties of ordered structures are definable.

*Ellipsoid Method.* In [Anderson et al. 2013], it is shown that the feasibility problem and, indeed, the optimization problem, for linear programming, is definable in FPC. The proof proceeds by means of expressing in the logic a version of Khachiyan's ellipsoid method [Khachiyan 1980]. In short, the ellipsoid method for determining the feasibility of a linear program proceeds by choosing a vector $x \in \mathbb{Q}^V$ (one may as well begin from the vector of all zeroes) and calculating, based on the bit complexity of the instance, an ellipsoid around $x$ which is guaranteed to include the polytope defined by the constraints $C$. Now, if $x$ itself does not satisfy all the constraints in $C$, we can choose one $c \in C$ for which $a_c^T x > b_c$ and use it to calculate a new vector $x'$ and an ellipsoid centred on $x'$ which still includes the polytope defined by $C$. The construction guarantees that the volume of the new ellipsoid is at most half that of the original one. This means that in a number of steps that is bounded by a polynomial in the size of the instance, the process converges to a centre $x$ that satisfies all the constraints in $C$ *or* the volume of the ellipsoid is small enough that we know for certain that the polytope is empty.

It is not difficult to show that all the calculations involved in computing the ellipsoid and centres can be expressed in $\mathrm{FPC}$ (see [Holm 2010] for details on how linear algebra using matrices without an order on the rows and columns can be expressed). The one step in the algorithm outlined above that causes difficulty is the *choice* of a violated constraint, which is used to define a new centre. There is no way, in any logical formalism that respects automorphisms of the structure on which it is interpreted (as any reasonable logic must) to choose an arbitrary element. However, the ellipsoid method is quite robust and it still works as long as, at each stage, we can construct some hyperplane that separates our current centre $x$ from all points in the polytope defined by $C$. And, as shown in [Anderson et al. 2013], such a construction can be done canonically by taking the set of all constraints in $C$ that are violated by $x$ and taking their sum as our separating hyperplane. We thus get that feasibility of linear programs (and also the linear programming optimization problem) are definable in $\mathrm{FPC}$.

*Separation Oracles.* So far, we have considered linear programs represented *explicitly*. That is, all the constraints are written out as part of the input structure. Relying on the robustness of the ellipsoid method, we can take this method further. In many applications of linear programming, we are not given an explicit constraint matrix, but some succinct description from which explicit constraints can be derived. In particular, the number of constraints may be exponential in the size of the input. The ellipsoid method can be used in such cases as long as we have a means of determining, for any vector $x$, whether it is in the polytope $P$ described by the constraint matrix and, if it is not, a hyperplane that separates $x$ from the polytope. This is known as a *separation oracle* for $P$. It is shown in [Anderson et al. 2013] that, as long as a separation oracle for a polytope $P$ is itself definable in $\mathrm{FPC}$, then the corresponding linear programming optimization problem can also be defined in $\mathrm{FPC}$.

This reduction of optimization to separation reveals an interesting relationship of linear programming with the equivalence relations $\equiv^{C^k}$. We are given a polytope $P \subseteq \mathbb{Q}^V$ in the form of some relational structure $\mathbf{A}$ and an $\mathrm{FPC}$ interpretation $\Phi$ which acts as a separation oracle. That is, given an $x \in \mathbb{Q}^V$ (and we assume that $V$ is part of the universe of $\mathbf{A}$, though the constraint set $C$ is not and may, in general, be exponentially larger than $\mathbf{A}$), $\Phi$ interpreted in a suitable expansion of $\mathbf{A}$ with $x$ either determines that $x$ is in the polytope $P$ or defines a hyperplane separating $x$ from $P$. Then, there is some $k$ (the exact value will depend on $\Phi$) so that we can take the quotient $V' = V/_{\equiv^{C^k}}$ (where $\equiv^{C^k}$ is defined with respect to the structure $\mathbf{A}$) and project $P$ to a polytope $P' \subseteq \mathbb{Q}^{V'}$ in such a way that feasibility and optimization with respect to $P$ can be reduced to similar questions about $P'$. This should be compared with [Grohe et al. 2014] where it is shown, essentially, that in the special case of explictly represented constraint matrices, taking $k = 2$ gives a similar result. This is then used as a preprocessing tool for optimizing the performance of linear programming solvers.

*Maximum Matching.* An important application given in [Anderson et al. 2013] of the $\mathrm{FPC}$ definability of linear programming is to the *maximum matching problem*. Recall that a matching in a graph $G$ is a set of edges $M$ so that each vertex in $G$ is incident on at most one edge in $M$. The maximum matching problem is then to find a matching in $G$ of maximum size. Note that it would not be possible, in $\mathrm{FPC}$ or any other logic, to give a formula that would actually define the set $M$ that is a maximum matching. This is because $M$ would not, in general, be invariant under automorphisms of $G$. For instance, if $G$ is $K_n$—the $n$-vertex complete graph—then it contains an exponential (in $n$) number of distinct maximum matchings all of which map to each other under automorphisms of $G$. However, it is shown in [Anderson et al. 2013] that the *size* of a maximum matching in $G$ can be defined in $\mathrm{FPC}$. It is known, thanks to Edmonds [1965]

that we can associate with a graph $G$ a *matching polytope* with an exponential number of constraints (and Rothvoß [2014] has shown that this is essentially optimal) so that optimizing over this polytope yields a maximum matching. Anderson et al. [2013] show that a separation oracle for the matching polytope can be defined in $\mathrm{FPC}$ interpreted on the graph $G$ and hence deduce that the size of the maximum matching is definable. This settles a fifteen-year-old question posed by Blass et al. [1999] who conjectured that the existence of a perfect matching in $G$ was not definable even in the stronger formalism of Choiceless Polynomial Time with Counting.

## 5. SYMMETRIC CIRCUITS

Let us now turn to the relationship between definability in $\mathrm{FPC}$ and circuit complexity. We start with a brief introduction to the formalism of circuit complexity.

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

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

Each $f_n$ can be represented by a *circuit* $C_n$ which is a directed acyclic graph where we think of the vertices as gates suitably labeled by Boolean operators $\wedge, \vee, \neg$ for the internal gates and by inputs $x_1, \ldots, x_n$ for the gates without incoming edges. One gate is distinguished as determining the output. If there is a polynomial $p(n)$ bounding the size of $C_n$ (i.e. the number of gates in $C_n$), then the language $L$ is said to be in the complexity class $\mathrm{P/poly}$. If, in addition, the family of circuits is *uniform*, meaning that the function that takes $n$ to $C_n$ is itself computable in polynomial time, then $L$ is in $\mathrm{PTime}$. For the definition of either of these classes, it does not make a difference if we expand the class of gates that we can use in the circuit beyond the Boolean basis to include, for instance, *threshold* or *majority* gates. The presence of such gates can make a difference for more restricted circuit complexity classes, for instance when we limit the depth of the circuit to be bounded by a constant, but not when we allow arbitrary polynomial-size circuits. Also, in the circuit characterization of $\mathrm{PTime}$, it does not make a difference if we replace the uniformity condition with a stronger requirement. Say, we might require that the function taking $n$ to $C_n$ is computable in $\mathrm{DLogTime}$.

*Circuits on Graphs.* We are interested in languages that represent properties of relational structures such as graphs. For simplicity, in what follows, let us restrict attention to directed graphs, i.e. structures in a vocabulary with one binary relation. A property of such graphs that is in $\mathrm{PTime}$ can be recognised by a family $(C_n)_{n\in\omega}$ of Boolean circuits of polynomial size and uniformity, as before, where now the inputs to $C_n$ are labelled by the $n^2$ *potential edges* of an $n$-vertex graph, each taking a value of $0$ or $1$. Of course, given an $n$-vertex graph $G$, there are many ways that it can be mapped onto the inputs of the circuit $C_n$. So, to ensure that the family of circuits is really defining a property of graphs, we require it to be *invariant* under the choice of this mapping. That is, each input of $C_n$ carries a label of the form $(i,j)$ for $i,j \in [n]$ and we require the output to be unchanged under any permutation $\pi \in S_n$ acting on the inputs by the action $(i,j) \mapsto (\pi(i),\pi(j))$. It is clear that any property of graphs that is invariant under isomorphisms of graphs and is in $\mathrm{PTime}$ is decided by such a family of circuits.

*From Logic to Circuits.* It turns out that the properties of graphs that are definable in logics such as $\mathrm{FPC}$ are decided by circuits with a stronger invariance condition. Say that a circuit $C_n$ is *symmetric* if any permutation $\pi \in S_n$ can be extended to an *automorphism* of $C_n$ which takes each input $(i,j)$ to $(\pi(i),\pi(j))$. It is clear that symmetric circuits are necessarily invariant and it is not difficult to come up with examples that show that the converse is not true.

It is also not difficult to show we can translate a formula of first-order logic, say, in the language of graphs into a family of symmetric circuits. Given a first-order sentence $\phi$ and a positive integer $n$, we define the circuit $C_n^\phi$ by taking as gates the set of pairs $(\psi, \mathbf{a})$ where $\psi$ is a subformula of $\phi$ and a is a tuple of values from $[n]$, one for each free variable occurring in $\psi$. The input gates correspond to the atomic formulas and the output gate is the one labelled by $\phi$. If the outermost connective in $\psi$ is a Boolean operation, the gate $(\psi, \mathbf{a})$ is labelled by that operator and it is connected to the gates corresponding to subformulas in the obvious way. If $\psi$ is $\exists x\theta$, then $(\psi, \mathbf{a})$ would be an OR gate with inputs from $(\theta, \mathbf{a}a)$ for all values $a \in [n]$ and similarly a universal formula attaches to a large AND gate. It is not difficult to see that the circuits constructed are symmetric as for any $\pi \in S_n$, the map that takes $(\psi, \mathbf{a})$ to $(\psi, \pi(\mathbf{a}))$ is an automorphism of the circuit. Moreover, the *depth* of the circuit is a function of $\phi$, so a fixed formula yields a family of circuits of constant depth. On the other hand, we can take a formula $\phi$ of fixed-point logic, FP, and for each $n$, obtain a first-order formula $\theta_n$ with $k$ (independent of $n$) variables and quantifier depth bounded by a polynomial in $n$ so that $\phi$ and $\theta_n$ are equivalent on structures with at most $n$ elements. Converting this into a circuit by the translation above yields a polynomial-size family of symmetric Boolean circuits (the size is bounded by $c \cdot n^k$ where $c$ is the number of sub-formulas of $\phi$) that is equivelent to $\phi$. If we start with a sentence $\phi$ of FPC instead, we can use the translation to $C^k$ (see Theorem 2.3) to obtain a family of symmetric circuits with threshold (or majority) gates. We need these additional gates to translate the counting quantifiers that appear in the $C^k$ formula and Boolean gates will no longer suffice. Symmetric circuits have been studied for the repesentation of properties defined in logic under different names in the literature (see [Denenberg et al. 1986; Otto 1996]).

*From Circuits to Logic.* The recent paper [Anderson and Dawar 2014] establishes that the correspondence between decidability by a family of symmetric circuits and definability in logic is tight by giving translations in the other direction. In particular, the following is shown there.

THEOREM 5.1. *A class of graphs is accepted by a polynomially uniform symmetric family of Boolean circuits* if, and only if, *it is definable by an* FP *formula interpreted in* $G \uplus ([n], <)$.

Here, $G \uplus ([n], <)$ denotes a structure consisting of a graph $G$ together with a disjoint set of elements $[n]$ of the same cardinality as the set of vertices of $G$. In this structure, we have the edge relation on the vertices of $G$ and a linear order on the elements $[n]$ and no other relations. In particular, the vertices of $G$ are not ordered and there is no relation that connects the vertices of $G$ with the elements of $[n]$. One can write an FP formula for such structures that states that the number of vertices is even, since this amounts to saying that the length of the linear order $([n], <)$ is even. But, it is not difficult to show by a pebble game argument that no FP formula can say that the number of edges in the graph $G$ is even, even in the presence of the linear order on the side. It follows from Theorem 5.1 that this simple *invariant* property of graphs is not decidable by any polynimially-uniform family of *symmetric* circuits. It also follows that allowing *threshold* or *counting* gates in symmetric circuits leads to a model that is more powerful than just Boolean circuits, in contrast to the situation of general polynomial-size circuits. The exact power of polynomially uniform families of symmetric circuits with threshold gates is determined by the following theorem from [Anderson and Dawar 2014]:

THEOREM 5.2. *A class of graphs is accepted by a polynomially uniform symmetric family of threshold circuits* if, and only if, *it is definable in* FPC.

A consequence of Theorems 5.2 and 5.1 is that any translation of *invariant* Boolean circuits into equivalent *symmetric* circuits, even allowing additional threshold gates in the latter, will necessarily involve a super-polynomial blow-up in the size of the circuits.

Theorem 5.2 gives a natural and purely circuit-based characterization of $\mathrm{FPC}$ definability, justifying the claim that this is a natural and robust class. It also shows that inexpressibility results for $\mathrm{FPC}$ can be understood as lower bound results against a natural class of circuits. The holy grail of circuit complexity is to prove lower bounds against the class of polynomial-size circuit families. So far, lower bounds have been proved by imposing various restrictions on the families, such as monotonicity (where a famous result of Razborov [1985] showed that no polynomial size family of *monotone* circuits can decide the clique problem) or bounded depth. We can now add symmetric circuits to the catalogue of circuit classes against which we are able to prove lower bounds. It is instructive to put this in the context of the expressibility and inexpressibility results for $\mathrm{FPC}$ mentioned previously. In particular, it follows that there is a polynomially uniform family of symmetric threshold circuits for deciding whether a graph has a perfect matching. But, there is no such family that decides whether a graph contains a Hamiltonian cycle. A natural circuit model that separates these two problems is certainly an intriguiing development.

*Proof Idea.* To conclude this section, I present some of the ideas underlying the proofs of Theorems 5.1 and 5.2. One direction is easy, by the argument given above about translating formulas into circuits. In the other direction, we want to use the symmetry of the circuit to derive an equivalent formula in the fixed-point logic. If $C_n$ is a symmetric circuit taking $n$-vertex graphs as input, we can assume without loss of generality, that the automorphism group of $C_n$ is exactly the symmetric group $S_n$ acting in the natural way on its inputs. For a gate $g$ in $C_n$, we say that a set $X \subseteq [n]$ ***supports*** $g$ if for every $\pi \in S_n$ such that $\pi(x) = x$ for all $x \in X$, we also have $\pi(g) = g$. Now, when we consider the circuit families $(C_n)_{n\in\omega}$ that arise as translations of $\mathrm{FPC}$ formulas, we can note that there is a constant $k$ such that all gates have a support of size at most $k$. This is because the gates are labelled by pairs $(\psi, \mathbf{a})$ where $\mathbf{a}$ is a $k$-tuple of elements from $[n]$ and it is easily checked that the set of elements that appear in $\mathbf{a}$ is a support of the gate. The main technical lemma in the proof of Theomem 5.2 establishes that this is, in fact, necessary for all symmetric circuits: if $(C_n)_{n\in\omega}$ is a family of symmetric circuits of polynomial size then there is a $k$ such that all gates in $C_n$ have a support of at most $k$ elements. Moreover, given a description of the circuit $C_n$, there is a polynomial-time algorithm that will determine a minimal size support of each gate. Now, given a graph $G$ on $n$ vertices and a bijection $\gamma : [n] \to V(G)$ that determines how this graph is mapped to the inputs of $C_n$, whether a gate $g$ in $C_n$ evaluates to TRUE is completely determined by $\gamma$ restricted to the support of $g$. We can thus represent the set of all maps $\gamma$ that make $g$ true as a $k$-ary relation on $G$. These $k$-ary relations admit an inductive definition (by induction on the construction of the circuit $C_n$) which allows us to turn the circuit family into a formula of $\mathrm{FP}$ (for Boolean circuits) or $\mathrm{FPC}$ (if the circuit also has threshold gates). This relies on the fact that the map that takes $n$ to $C_n$ is polynomial-time computable and therefore definable in $\mathrm{FP}$ on ordered structures. We use the linear order available "on the side" in Theorem 5.1 for this. It is not necessary in the proof of Theorem 5.2 as it can be replaced by the use of numerical variables.

## 6. CONCLUSION

I noted in the introduction that the conjecture that $\mathrm{FPC}$ captures $\mathrm{PTime}$ was based on the intuition that algorithmic techniques that are obviously polynomial-time are all ex-

pressible in this logic. The result of Cai et al. and subsequent results on inexpressiblity in FPC essentially show that one important algorithmic technique, that of Gaussian elimination, is not expressible in the logic. Nonetheless, the recent results surveyed in this article show that many powerful and certainly non-obvious polynomial-time algorithmic techniques are indeed expressible in FPC. In particular, we have seen that FPC can express maximum matching in graphs, feasibility of linear programs and arbitrary minor-closed classes of graphs. For each of these, the fact that the problem is in polynomial-time was a major result of its day. Finally, the results of Section 5 show further that FPC is a natural and robustly defined class by giving a characterization of it by a natural and independently-motivated circuit model. Taken together, these justify the claim that FPC is a *natural* and *powerful* class of problems.

## ACKNOWLEDGMENTS

This article is a write-up of a talk I first gave as part of the workshop on *Logic and Computational Complexity* at Vienna in July 2014, devoted to Neil Immerman's 60th birthday. I would like to thank the organizers of that workshop, and I would like to thank Neil Immerman for inviting me to write it up in this column.

## REFERENCES

M. Anderson and A. Dawar. 2014. On Symmetric Circuits and Fixed-Point Logics. In *31st Intl. Symp. Theoretical Aspects of Computer Science (STACS 2014)*. 41–52.

M. Anderson, A. Dawar, and B. Holm. 2013. Maximum Matching and Linear Programming in Fixed-Point Logic with Counting. In *28th Annual ACM/IEEE Symp. Logic in Computer Science*. 173–182.

A. Atserias, A. Bulatov, and A. Dawar. 2009. Affine Systems of Equations and Counting Infinitary Logic. *Theoretical Computer Science* 410, 18 (2009), 1666–1683.

A. Blass, Y. Gurevich, and S. Shelah. 1999. Choiceless Polynomial Time. *Annals of Pure and Applied Logic* 100 (1999), 141–187.

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.

E. Dahlhaus. 1984. Reduction to NP–Complete Problems by Interpretation. In *LNCS 171*. Springer-Verlag, 357–365.

A. Dawar. 1998. A Restricted Second Order Logic for Finite Structures. *Information and Computation* 143 (1998), 154–174.

A. Dawar. 2007. Finite Model Theory on Tame Classes of Structures. In *MFCS (Lecture Notes in Computer Science)*, Vol. 4708. Springer, 2–12.

A. Dawar and E. Grädel. 2010. Properties of Almost All Graphs and Generalized Quantifiers. *Fundam. Inform.* 98, 4 (2010), 351–372.

A. Dawar, M. Grohe, B. Holm, and B. Laubner. 2009. Logics with Rank Operators. In *Proc. 24th IEEE Symp. on Logic in Computer Science*. 113–122.

L. Denenberg, Y. Gurevich, and S. Shelah. 1986. Definability by Constant-depth Polynomial-size Circuits. *Information and Control* 70 (1986), 216–240.

H-D. Ebbinghaus and J. Flum. 1999. *Finite Model Theory* (2nd ed.). Springer.

J. Edmonds. 1965. Maximum Matching and a Polyhedron with $0, 1$ Vertices. *J. Research National Bureau of Standards* 69 B (1965), 125–130.

F. Gire and H. Hoang. 1998. An Extension of Fixpoint Logic with a Symmetry-Based Choice Construct. *Information and Computation* 144 (1998), 40–65.

M. Grohe. 1998. Fixed-Point Logics on Planar Graphs. In *Proc. 13th IEEE Annual Symp. Logic in Computer Science*. 6–15.

M. Grohe. 2012. Fixed-point definability and polynomial time on graphs with excluded minors. *J. ACM* 59, 5 (2012), 27:1–27:64.

M. Grohe. 2014. Descriptive Complexity, Canonisation, and Definable Graph Structure Theory. (2014). http://www.automata.rwth-aachen.de/~grohe/cap/index.en Draft of Monograph.

M. Grohe, K. Kersting, M. Mladenov, and E. Selman. 2014. Dimension Reduction via Colour Refinement. In *Algorithms - ESA 2014 - 22nd Annual European Symposium*. 505–516.

M. Grohe and J. Mariño. 1999. Definability and Descriptive Complexity on Databases of Bounded Tree-Width. In *Proc. 7th International Conference on Database Theory (LNCS)*, Vol. 1540. Springer, 70–82.

L. Hella. 1992. Logical hierarchies in PTIME. In *Proc. 7th IEEE Symp. on Logic in Computer Science*. 360–368.

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

N. Immerman. 1982. Relational Queries Computable in Polynomial Time (Extended Abstract). In *Proc. 14th Annual ACM Symp. Theory of Computing*. 147–152.

N. Immerman. 1986. Relational Queries computable in Polynomial Time. *Information and Control* 68 (1986), 86–104.

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

N. Immerman and E. S. Lander. 1990. Describing Graphs: A First-order Approach to Graph Canonization. In *Complexity Theory Retrospective*, A. Selman (Ed.). Springer-Verlag.

L. G. Khachiyan. 1980. Polynomial Algorithms in Linear Programming. *U. S. S. R. Comput. Math. and Math. Phys.* 20, 1 (1980), 53–72.

K. Kuratowski. 1930. Sur le Probléme des Courbes Gauches en Topologie. *Fundamenta Mathematicae* 15 (1930), 271283.

L. Lovász and P. Gács. 1977. Some Remarks on Generalized Spectra. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* 23 (1977), 27–144.

E. M. Luks. 1982. Isomorphism of Graphs of Bounded Valence Can be Tested in Polynomial Time. *J. Comput. System Sci.* 25 (1982), 42–65.

M. Otto. 1996. The Logic of Explicitly Presentation-Invariant Circuits. In *Computer Science Logic, 10th International Workshop, CSL '96, Annual Conference of the EACSL*. 369–384.

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

A. A. Razborov. 1985. Lower Bounds on the Monotone Complexity of some Boolean Functions. *Dokl. Akad. Nauk. SSSR* 281 (1985), 798–801.

N. Robertson and P. D. Seymour. 1995. Graph Minors. XIII. The Disjoint Paths Problem. *J. Comb. Theory, Ser. B* 63 (1995), 65–110.

N. Robertson and P. D. Seymour. 2003. Graph Minors. XVI. Excluding a non-planar graph. *J. Comb. Theory, Ser. B* 89 (2003), 43–76.

N. Robertson and P. D. Seymour. 2004. Graph Minors. XX. Wagner's conjecture. *J. Comb. Theory, Ser. B* 92 (2004), 325–357.

T. Rothvoß. 2014. The Matching Polytope has Exponential Extension Complexity. In *Symp. Theory of Computing, STOC 2014*. 263–272.

M. Y. Vardi. 1982. The Complexity of Relational Query Languages. In *Proc. of the 14th ACM Symp. on the Theory of Computing*. 137–146.

K. Wagner. 1937. Uber eine Eigenschaft der Ebenen Komplexe. *Math. Ann.* 114 (1937), 570–590.