Symmetric Circuits for Rank Logic

Anuj Dawar
Department of Computer Science and Technology, University of Cambridge, UK
anuj.dawar@cl.cam.ac.uk
https://orcid.org/0000-0003-4014-8248

Gregory Wilsenach[1]
Department of Computer Science and Technology, University of Cambridge, UK
gregory.wilsenach@cl.cam.ac.uk

Abstract

Fixed-point logic with rank (FPR) is an extension of fixed-point logic with counting (FPC) with operators for computing the rank of a matrix over a finite field. The expressive power of FPR properly extends that of FPC and is contained in P, but it is not known if that containment is proper. We give a circuit characterization for FPR in terms of families of symmetric circuits with rank gates, along the lines of that for FPC given by [Anderson and Dawar 2017]. This requires the development of a broad framework of circuits in which the individual gates compute functions that are not symmetric (i.e., invariant under all permutations of their inputs). This framework also necessitates the development of novel techniques to prove the equivalence of circuits and logic. Both the framework and the techniques are of greater generality than the main result.

2012 ACM Subject Classification Theory of computation → Circuit complexity, Theory of computation → Finite Model Theory, Theory of computation → Complexity theory and logic

Keywords and phrases fixed-point logic with rank, circuits, symmetric circuits, uniform families of circuits, circuit characterization, circuit framework, finite model theory, descriptive complexity

Digital Object Identifier 10.4230/LIPIcs.CSL.2018.20

Related Version A full version of this paper is available at [7], `https://arxiv.org/abs/1804.02939`.

1 Introduction

The study of extensions of fixed-point logics plays an important role in the field of descriptive complexity theory. In particular, fixed-point logic with counting (FPC) has become a reference logic in the search for a logic for polynomial-time (see [2]). In this context, Anderson and Dawar [1] provide an interesting characterization of the expressive power of FPC in terms of circuit complexity. They show that the properties expressible in this logic are exactly those that can be decided by polynomially-uniform families of circuits (with threshold gates) satisfying a natural *symmetry* condition. Not only does this illustrate the robustness of FPC as a complexity class within P by giving a distinct and natural characterization of it, it also demonstrates that the techniques for proving inexpressibility in the field of finite model theory can be understood as lower-bound methods against a natural circuit complexity class. This raises an obvious question (explicitly posed in the concluding section of [1]) of how to obtain circuit characterizations of logics more expressive than FPC, such as choiceless polynomial time (CPT) and fixed-point logic with rank (FPR). It is this last question that we address in this paper.

[1] Funding provided by the Gates Cambridge Scholarship.

Fixed-point logic with rank extends the expressive power of FPC by means of operators that allow us to define the rank of a matrix over a finite field. Such operators are natural extensions of counting – counting the dimension of a definable vector space rather than just the size of a definable set. At the same time they make the logic rich enough to express many of the known examples that separate FPC from P. Rank logics were first introduced in [5]. The version FPR we consider here is that defined by Grädel and Pakusa [9] where the prime characteristic is a parameter to the rank operator, and we do not have a distinct operator for each prime number. Formal definitions of these logics are given in Section 2. We give a circuit characterization, in terms of symmetric circuits, of FPR. One might think, at first sight, that this is a simple matter of extending the circuit model with gates for computing the rank of a matrix. It turns out, however, that the matter is not so simple as the symmetry requirement interacts in surprising ways with such rank gates. It requires a new framework for defining classes of such circuits, which yields remarkable new insights.

The word *symmetry* is used in more than one sense in the context of circuits (and also in this paper). We say that a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is symmetric if the value of the function on a string s is determined by the number of 1s in s. In other words, f is invariant under *all* permutations of its input. In contrast, when we consider the input to a Boolean function to be the adjacency matrix of an n-vertex graph, for example, and $f : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ decides a graph property, then f is invariant under all permutations of its input induced by permutations of the n vertices of the graph. We call such a function *graph-invariant*. More generally, for a relational vocabulary τ and a standard encoding of n-element τ-structures as strings over $\{0,1\}$, we can say that function taking such strings as input is τ-invariant if it is invariant under permutations induced by the n elements. A circuit C computing such an invariant function is said to be *symmetric* if every permutation of the n elements extends to an automorphism of C. It is families of symmetric circuits in this sense that characterize FPC in [1]. The restriction to symmetric circuits arises naturally in the study of logics and has appeared previously under the names of generic circuits in the work of [8] and explicitly order-invariant circuits in the work of Otto [15]. In this paper, we use the word "symmetric", and context is used to distinguish the meaning of the word as applied to circuits from its meaning as applied to Boolean functions.

The main result of [1] says that the properties of τ-structures definable in FPC are exactly those that can be decided by P-uniform families of symmetric circuits using AND, OR, NOT and majority gates. Note that each of these gates itself computes a Boolean function that is symmetric in the strong sense identified above. On the other hand, a gate for computing a rank threshold function, e.g. one that takes as input a $n \times n$ matrix and outputs 1 if the rank of the matrix is greater than a threshold t, is *not* symmetric. In our circuit characterization of FPR we necessarily have to consider such non-symmetric gates. Indeed, we can show that P-uniform families of symmetric circuits using gates for *any* symmetric functions do not take us beyond the power of FPC. This is a further illustration of the robustness of FPC. In order to go beyond it, we need to introduce gates for Boolean functions that are not symmetric. We construct a systematic framework for including functions computing τ-invariant functions for arbitrary multi-sorted relational vocabularies τ in Section 3. We also explore what it means for such circuits to be symmetric.

The proof of the circuit characterization of FPC relies on the *support theorem* proved in [1]. This establishes that for any P-uniform family of circuits using AND, OR, NOT and majority gates there is a constant k such that every gate has a support of size at most k. That is to say that we can associate with every gate g in the circuit C_n (the circuit in the family that works on n-element structures) a subset X of $[n]$ of size at most k such that any permutation of $[n]$ fixing X pointwise extends to an automorphism of C_n that fixes g. This theorem is crucial to the translation of the family of circuits into a formula of FPC, which is the difficult (and novel) direction of the equivalence. In attempting to do the same with circuits that now use rank-threshold gates we are faced with the difficulty that the proof of the support theorem in [1] relies in an essential way on the fact that the Boolean function computed at each gate is symmetric. We are able to overcome this difficulty and prove a support theorem for circuits with rank gates but this requires substantial, novel technical machinery.

Another crucial ingredient in the proof of Anderson and Dawar is that we can eliminate redundancy in the circuit C_n by making it *rigid*. That is, we can ensure that the *only* automorphisms of C_n are those that are induced by permutations of $[n]$. Here we face the difficulty that identifying the symmetries and eliminating redundancy in a circuit that involves gates computing τ-invariant functions requires us to solve the isomorphism problem for τ-structures. This is a hard problem (or, at least, one that we do not know how to solve efficiently) even when the τ-structures are 0-1-matrices. We overcome this difficulty by placing a further restriction on circuits that we call *transparency*. Circuits satisfying this condition have the property that their lack of redundancy is transparent.

In the characterization of FPC, the translation from formulas into families of circuits is easy and, indeed, standard. In our case, we have to show that formulas of FPR translate into uniform families of circuits using rank-threshold gates that are symmetric and transparent. This is somewhat more involved technically and presented in Section 5. Finally, with all these tools in place, the translation of such P-uniform families of circuits into formulas of FPR given in Section 6 completes the characterization. This still requires substantial new techniques. The translation of circuits to formulas in [1] relies on the fact that in order to evaluate a gate computing a symmetric Boolean function, it suffices to count the number of inputs that evaluate to true and there is a bijection between the orbits of a gate and tuple assignments to its support. When counting is no longer sufficient, this bijection has to preserve more structure and demonstrating this in the case of matrices requires new insight.

Space limitations prevent us from giving details of proofs. These and much more detail can be found in the full version of this paper [7].

2 Background

We write $\mathbf{Sym}_X$ to denote the group of all permutations of the set X. Let G be a group and X be a set on which a group action is defined and let $S \subseteq X$. Let $\mathbf{Stab}_G(S) := \{\pi \in G : \forall s \in S,\ \pi(s) = s\}$. For $n \in \mathbb{N}$ we write $\mathbf{Sym}_n$ to abbreviate $\mathbf{Sym}_{[n]}$ and write $\mathbf{Stab}_n(S)$ to abbreviate $\mathbf{Stab}_{\mathbf{Sym}_n}(S)$. In the event that the group is obvious from context we omit the subscript entirely. We let $A^{\underline{B}}$ denote the set of injections from the set B to the set A.

2.1 Logic

A *vocabulary* is a finite sequence of relation symbols $(R_1, \ldots, R_k)$, each of which has a fixed *arity*. We let $r_i \in \mathbb{N}$ denote the arity of the relation symbol R_i. A *many-sorted vocabulary* is a tuple of the form (R, S, ν), where R is a relational vocabulary, S is a finite sequence of *sort* symbols, and ν is a function that assigns to each $R_i \in R$ a tuple $\nu(R_i) := (s_1, \ldots, s_{r_i})$, where for each $j \in [r_i]$, $s_j \in S$. We call $\nu(R_i)$ the *type* of R_i. A *τ-structure* $\mathcal{A}$ is a tuple $(U, R_1^{\mathcal{A}}, \ldots, R_k^{\mathcal{A}})$ where $U = \uplus_{s \in S} U_s$ is a disjoint union of non-empty sets and is called the *universe* of $\mathcal{A}$, and for all $i \in [k]$, $R_i^{\mathcal{A}} \subseteq U_{s_1} \times \ldots \times U_{s_{r_i}}$, where $(s_1, \ldots, s_{r_i}) = \nu(R_i)$. The size of $\mathcal{A}$, denoted by $|\mathcal{A}|$, is the cardinality of its universe. All structures in this paper are finite.

We assume the reader is familiar with first-order logic (FO), inflationary fixed-point logic (FP), fixed-point logic with counting (FPC), and first-order logic with counting quantifiers (FOC). For details on these logics please see [10, 13].

2.2 Rank Logic

Let FPR$[\tau]$ denote *fixed-point logic with rank* over the vocabulary τ. FPR extends FP with an operator that denotes the rank of a definable matrix over a finite field, as well as other mechanisms for reasoning about quantity. Each variable in a formula of FPR is either a *number* or *vertex* variable, with vertex variables interpreted by elements of the universe and number variables interpreted by natural numbers. All atomic formulas in FP$[\tau]$ are atomic formulas in FPR$[\tau]$. We say that t is a *number term* if t is a number variable or if t is an application of the *rank operator*, i.e. $t := [\mathbf{rk}(\vec{x}, \vec{\nu} \leq \vec{t}, \vec{y}, \vec{\mu} \leq \vec{s}, \pi \leq \eta).\phi]$, where ϕ is a number term or formula, $\vec{t}$ and $\vec{s}$ are tuples of number terms bounding the sequences of number variables $\vec{\mu}$ and $\vec{\nu}$, and η is a number term bounding the number variable π. If t_1 and t_2 are number terms then $t_1 \leq t_2$ and $t_1 = t_2$ are atomic formulas. The formulas of FPR are formed by closing the set of atomic formulas under the usual Boolean connectives, the first-order quantifiers, and the fixed-point operator. When quantifying over number variables we only allow bounded quantification. Second-order variables, such as those that appear in a fixed-point application, may have mixed-type. For more detail on the syntax and semantics of FPR please see [9].

Let FOR$[\tau]$ be the set of formulas in FPR$[\tau]$ without an application of the fixed-point operator. We define for each prime p, and natural number r, a rank quantifier $\mathbf{rk}_p^r$, such that $\mathbf{rk}_p^r \vec{x}\vec{y}.\phi$ is interpreted as $[\mathbf{rk}(\vec{x}, \vec{y}, \pi).\phi] \geq t_r$, where π is assigned to p and t_r is a number term that evaluates to r. Let $\mathcal{R}$ be the set of all such quantifiers and FO+rk$[\tau]$ be the closure of FO$[\tau]$ under $\mathcal{R}$. For more details on rank quantifiers see [5].

3 Generalizing Symmetric Circuits

A *Boolean basis* is a set of Boolean functions. We always use $\mathbb{B}$ to denote a basis. Let $\mathbb{B}_{\mathrm{std}}$ denote the *standard basis* containing the Boolean functions computing AND, OR and NOT for each arity. Let $\mathbb{B}_{\mathbf{maj}}$ denote the *majority basis*, i.e. the extension of $\mathbb{B}_{\mathrm{std}}$ with functions computing majority for each arity.

A Boolean circuit C over a basis $\mathbb{B}$ is a directed acyclic graph in which each internal gate g is labelled with a function $f_g : \{0,1\}^q \to \{0,1\} \in \mathbb{B}$ where q is the fan-in of g. Notice that if f_g were allowed to be arbitrary then an order would need to be imposed on the children of g to ensure unambiguous evaluation. As such, the usual notion of a circuit as a directed acyclic graph with no structure on the children of any gate g implicitly assumes that f_g is invariant under all permutations of its inputs – i.e. f_g is a symmetric function. It is easy to see that the standard basis and majority basis contain only symmetric functions.

Anderson and Dawar [1] characterize the expressive power of FPC in terms of symmetric circuits over the majority basis. This circuit model cannot be strengthened by extending the basis by symmetric functions (see [7]). As our ultimate aim is a circuit characterisation of FPR, which is strictly more expressive than than FPC, we would like to consider circuits defined over bases containing non-symmetric Boolean functions. In particular, we are interested in bases containing rank-threshold functions – i.e. functions that take in a matrix and decide if the matrix understood as having entries in some prime field has rank less than some threshold. While these functions are not symmetric in the full sense, they are symmetric in the sense of being invariant under row-column permutations.

To lead up to this we first develop a general framework of structured Boolean functions. These are functions whose inputs naturally encode τ-structures, rather than just matrices or strings, and where the output is invariant under the natural symmetries of such structures. We therefore define symmetric circuits in a general form where gates can be labelled by *isomorphism-invariant* structured functions.

3.1 Structured Functions

Let X be a finite set and $F : \{0,1\}^X \to \{0,1\}$. It is common to consider Boolean functions that take strings as input, which would correspond to taking $X = [n]$ for some $n \in \mathbb{N}$. The natural notion of symmetry for such functions is invariance under arbitrary permutations of X, i.e. the usual notion of a symmetric (Boolean) function. Alternatively, we might want to consider Boolean functions that take in more complex algebraic structures as input, which would involve selecting an index set X such that the input to the function encodes an appropriate structure. For example, if we are interested in functions that take directed graphs as inputs we would let $X = V^2$ for some vertex set V. We notice that in this case the natural symmetry condition would not be invariance under arbitrary permutation, but rather invariance under the action of a permutation of V.

In this subsection we formalise this notion and define a class of functions that take in many-sorted structures and define a natural symmetry notion for such functions. Let $\tau := (R, S, \nu)$ be a many-sorted vocabulary and let $D := \biguplus_{s \in S} D_s = \{(s, d) : d \in D_s\}$, be a disjoint union of non-empty sets. Let $\mathrm{str}(\tau, D)$ be the τ-structure with universe D and such that every relation is full (i.e. contains all possible tuples). We let $\mathrm{ind}(\tau, D)$ be the disjoint union of all the relations in $\mathrm{str}(\tau, D)$, i.e. $\mathrm{ind}(\tau, D) = \biguplus_{R_i \in R} R_i^{\mathrm{str}(\tau,D)} := \{(\vec{a}, R_i) : \vec{a} \in R_i^{\mathrm{str}(\tau,D)}, R_i \in R\}$. We often abbreviate $(\vec{a}, R_i) \in \mathrm{ind}(\tau, D)$ by $\vec{a}_{R_i}$. We call $\mathrm{ind}(\tau, D)$ the *index defined by* (τ, D).

We think of $\mathrm{ind}(\tau, D)$ as containing all those tuples that may appear in a relation in a τ-structure or, equivalently, the collection of ground atoms in the vocabulary τ with elements from the domain D. In this way each element of $\{0,1\}^{\mathrm{ind}(\tau,D)}$ encodes a τ-structure with universe D. We call a function $F : \{0,1\}^{\mathrm{ind}(\tau,D)} \to \{0,1\}$ a *(τ, D)-structured function*, or just a *structured function*, and we call τ and D the *vocabulary* and *universe* of F, and denote them by $\mathrm{voc}(F)$ and $\mathrm{unv}(F)$ respectively. We call $\mathrm{ind}(\tau, D)$ the *index* of F, and denote it by $\mathrm{ind}(F)$. We see that F defines a class of τ-structures with universe D. We are especially interested in structured functions that are symmetric in some sense, and hence decide properties of τ-structures, i.e. isomorphism-closed classes of structures.

Let H be a set. We think of a function $f : \mathrm{ind}(\tau, D) \to H$ as defining a labelling of $\mathrm{str}(\tau, D)$ by H and we identify f with this labelled instance of $\mathrm{str}(\tau, D)$. Let $f : \mathrm{ind}(\tau, D) \to H$ and $g : \mathrm{ind}(\tau, D') \to H$. We say that f and g are *isomorphic* if there is an isomorphism $\pi : \mathrm{str}(\tau, D) \to \mathrm{str}(\tau, D')$ such that $f(\vec{a}_R) = g((\pi\vec{a})_R)$ for all $\vec{a}_R \in \mathrm{ind}(\tau, D)$. In other words, f and g are isomorphic if, and only if, they are isomorphic as (labelled) structures. Notice that if $H = \{0,1\}$ then f and g define τ-structures and f and g are isomorphic if, and only if, the τ-structures they define are isomorphic.

We say that $F : \{0,1\}^{\mathrm{ind}(\tau,D)} \to \{0,1\}$ is *isomorphism-invariant* if for all $f, g : \mathrm{ind}(\tau, D) \to \{0,1\}$ whenever f and g are isomorphic then $F(f) = F(g)$.

3.2 Symmetric Circuits

We now generalise the circuit model in [1] in order to allow for circuits to be defined over bases that include non-symmetric (structured) functions. In this model each gate g is not only associated with an element of the basis, but also with a labelling function. This labelling function maps an appropriate set of labels (i.e. the index of the structured function associated with g) to the input gates of g. In concord with this generalisation, we also update the circuit-related notions from [1], e.g. circuit automorphisms, symmetry, etc. Moreover, we briefly discuss some of the important complications introduced by our generalisation, and introduce some of the important tools we use to address these complications.

▶ **Definition 1** (Circuits on Structures). Let $\mathbb{B}$ be a basis of structured functions and ρ be a relational vocabulary, we define a $(\mathbb{B}, \rho)$-*circuit* C of order n computing a q-ary query Q as a structure $\langle G, \Omega, \Sigma, \Lambda, L\rangle$.

- G is called the set of gates of C.
- Ω is an injective function from $[n]^q$ to G. The gates in the image of Ω are called the output gates. When $q = 0$, Ω is a constant function mapping to a single output gate.
- Σ is a function from G to $\mathbb{B} \uplus \rho \uplus \{0, 1\}$ such that $|\Sigma^{-1}(0)| \leq 1$ and $|\Sigma^{-1}(1)| \leq 1$. Those gates mapped to $\rho \uplus \{0, 1\}$ are called input gates, with those mapped to ρ called relational gates and those mapped to $\{0, 1\}$ called constant gates. Those gates mapped to $\mathbb{B}$ are called internal gates.
- Λ is a sequence of injective functions $(\Lambda_{R_i})_{R_i \in R}$ such that Λ_{R_i} maps each relational gate g with $\Sigma(g) = R_i$ to the tuple $\Lambda_{R_i}(g) \in [n]^{r_i}$. When no ambiguity arises we write $\Lambda(g)$ for $\Lambda_{R_i}(g)$.
- L associates with each internal gate g a function $L(g) : \text{ind}(\Sigma(g)) \to G$ such that if we define a relation $W \subseteq G^2$ by $W(h_1, h_2)$ iff h_2 is an internal gate and h_1 is in the image of $L(h_2)$, then (G, W) is a directed acyclic graph.

The definition requires some explanation. Each gate in G computes a function of its inputs and the relation W on G is the set of "wires". That is, $W(h, g)$ indicates that the value computed at h is an input to g. However, since the functions are structured, we need more information on the set of inputs to g and this is provided by the labelling L. $\Sigma(g)$ tells us what the function computed at g is, and thus $\text{ind}(\Sigma(g))$ tells us the structure on the inputs and $L(g)$ maps this to the set of gates that form the inputs to g. We let $H_g = \{h \in G : W(h, g)\}$ denote the set of inputs to the gate g. We let $\text{unv}(g)$ denote the universe of $\Sigma(g)$. We call a gate g a *symmetric gate* if $\Sigma(g)$ is a symmetric function and g a *non-symmetric gate* otherwise.

Let ρ be a relational vocabulary, $\mathcal{A}$ be a ρ-structure with universe U of size n, and $\gamma \in [n]^{\underline{U}}$. Let $\gamma\mathcal{A}$ be the structure with universe $[n]$ formed by mapping the elements of U in accordance with γ. The evaluation of a $(\mathbb{B}, \rho)$-circuit C of order n computing a q-ary query Q proceeds by recursively evaluating the gates in the circuit. The evaluation of the gate g for the bijection γ and input structure $\mathcal{A}$ is denoted by $C[\gamma\mathcal{A}](g)$, and is given as follows:

- if g is a constant gate then it evaluates to the bit given by $\Sigma(g)$,
- if g is a relational gate then g evaluates to true iff $\gamma\mathcal{A} \models \Sigma(g)(\Lambda(g))$, and
- if g is an internal gate let $L^{\gamma\mathcal{A}}(g) : \text{ind}(g) \to \{0, 1\}$ be defined by $L^{\gamma\mathcal{A}}(g)(x) = C[\gamma\mathcal{A}](L(g)(x))$, for all $x \in \text{ind}(g)$. Then g evaluates to true if, and only if, $\Sigma(g)(L^{\gamma\mathcal{A}}(g)) = 1$.

We say that C defines the q-ary query $Q \subseteq U^q$ under γ where $\vec{a} \in Q$ if, and only if, $C[\gamma\mathcal{A}](\Omega(\gamma\vec{a})) = 1$.

We now define a circuit automorphism for a circuit.

▶ **Definition 2** (Automorphism). Let $C = \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a $(\mathbb{B}, \tau)$-circuit of order n computing a q-ary query, and where $\mathbb{B}$ is a basis of isomorphism-invariant structured functions. Let $\sigma \in \mathbf{Sym}_n$ and $\pi : G \to G$ be a bijection such that

- for all output tuples $x \in [n]^q$, $\pi\Omega(x) = \Omega(\sigma x)$,
- for all gates $g \in G$, $\Sigma(g) = \Sigma(\pi g)$,
- for each relational gate $g \in G$, $\sigma\Lambda(g) = \Lambda(\pi g)$, and
- For each pair of gates $g, h \in G$, we have $W(h, g)$ if, and only if, $W(\pi h, \pi g)$ and for each internal gate g we have that $L(\pi g)$ and $\pi \cdot L(g)$ are isomorphic.

We call π an *automorphism* of C, and we say that σ *extends to an automorphism* π. The group of automorphisms of C is called $\mathbf{Aut}(C)$.

We are particularly interested in circuits that have the property that *every* permutation in $\mathbf{Sym}_n$ extends to an automorphism of the circuit.

▶ **Definition 3** (Symmetry). A circuit C on structures of size n is called *symmetric* if every $\sigma \in \mathbf{Sym}_n$ extends to an automorphism on C.

Suppose C does not contain a relational gate labelled by a relation symbol with non-zero arity. In that case C computes a constant function. For this reason, we always assume a circuit contains at least one relational gate with non-zero arity. Now, by assumption there exists a relational gate in C such that some element of $[n]$ appears in the tuple labelling that gate. By symmetry it follows that every element of $[n]$ appears in a tuple labelling a relational gate in C. It follows that no two distinct elements of $\mathbf{Sym}_n$ agree on all input gates and so we can associate with each $\pi \in \mathbf{Aut}(C)$ a unique $h(\pi) \in \mathbf{Sym}_n$ that it extends and it is easily seen that $h : \mathbf{Aut}(C) \to \mathbf{Sym}_n$ is a surjective group homomorphism. If h is also injective then we have that each element of σ extends uniquely to an automorphism of the circuit. In this case we say that a circuit has *unique extensions*.

▶ **Definition 4.** We say that a circuit C of order n has *unique extensions* if for every $\sigma \in \mathbf{Sym}_n$ there is at most one $\pi_\sigma \in \mathbf{Aut}(C)$ such that π_σ extends σ.

Many important technical tools, e.g. the support theorem, are only applicable to circuits with unique extensions. It is for this reason that a notion of a *rigid* circuit is introduced in [1]. Such circuits have unique extensions and it is shown that a symmetric circuit over the basis $\mathbb{B}_{\mathbf{maj}}$ can be converted in polynomial-time to an equivalent rigid one.

We should like to develop a property analogous to rigidity for our framework, as well as a similar polynomial-time translation. However, in our framework the value a gate g computes depends not just on the *set* of gates input to g but also on the structure of this set. This structure must be preserved by the action of an automorphism, and so we require that if π is an automorphism that maps g to g' then $\pi L(g)$ and $L(g')$ are isomorphic. Following from this observation, it can be shown that deciding if a function on the circuit is an automorphism, and indeed deciding almost any symmetry-related property, for circuits with non-symmetric gates is at least as hard as the graph-isomorphism problem. As such, constructing an argument analogous to [1], as well as establishing the numerous other crucial results whose proofs rely on the polynomial-time decidability of various circuit properties, would require the development of a polynomial-time algorithm for graph-isomorphism.

In order to proceed we explicitly restrict our attention to *transparent* circuits. We will define this term below, but before we do we need to define a notion of 'structural similarity' between gates that we call *syntactic-equivalence*.

▶ **Definition 5.** Let $C := \langle G, \Omega, \Sigma, \Lambda, L\rangle$ be a $(\mathbb{B}, \rho)$-circuit of order n. We recursively define the equivalence relation *syntactic-equivalence*, which we denote using the symbol '$\equiv$', on G as follows. If g and h are both input gates or both output gates then $g \equiv h$ if, and only if, $g = h$. Suppose g and h are both non-output internal gates and we have defined the syntactic-equivalence relation for all gates of depth less than the depth of either g or h. Then $g \equiv h$ if, and only if, $\Sigma(g) = \Sigma(h)$ and $L(g)/_{\equiv}$ and $L(h)/_{\equiv}$ are isomorphic (as labelled structures).

The intuition is that two gates are syntactically-equivalent if the circuits underneath these two gates are structurally equivalent. The important point is that if two gates are mapped to one another by an automorphism that extends the trivial permutation, then these gates are syntactically-equivalent. In fact, we prove a slightly stronger result.

▶ **Lemma 6.** *Let C be a circuit of order n and $\sigma \in \boldsymbol{Sym}_n$. Let $\pi, \pi' \in \boldsymbol{Aut}(C)$ be automorphisms extending σ. For every gate g in C we have $\pi(g) \equiv \pi'(g)$.*

In this way syntactic-equivalence constrains the automorphism group. We use syntactic-equivalence to establish sufficient conditions for a circuit to have unique extensions and, moreover, for various circuit-properties that reference automorphism to be polynomial-time decidable. With these two ideas in mind we define the following classes of circuits.

▶ **Definition 7.** Let C be a circuit and g be an internal gate in C. We say g has *injective labels* if $L(g)$ is an injection. We say g has *unique labels* if g has injective labels and no two gates in $W(g, \cdot)$ are syntactically-equivalent. We say C has *injective labels* (resp. *unique labels*) if every gate in C has injective labels (resp. unique labels). We say C is *transparent* if every non-symmetric gate in C has unique labels.

We can translate transparent circuits into circuits with unique labels in polynomial-time. We prove this by first showing that syntactic-equivalence can be computed for transparent circuits in polynomial-time. This follows from a straightforward inductive on depth, starting from the input gates and noting that the syntactic-equivalence classes of the next layer can be computed so long as you can solve the isomorphism problem for the gates in this next layer. This is easy to do for symmetric gates, as we can check set-equivalence easily, and in the case the gate is non-symmetric then this gate has unique labels, and so there is at most one candidate isomorphism, and it is easy to check if a given function is an isomorphism.

▶ **Lemma 8.** *There is an algorithm that takes as input a transparent circuit C and outputs the syntactic-equivalence relation on the gates of C. The algorithm runs in time polynomial in the size of C.*

The translation from transparent circuits to circuits with unique labels is defined as follows. We define a circuit by collapsing the gates of the input circuit into its syntactic-equivalence classes, i.e. taking a quotient of the circuit by syntactic-equivalence. The resultant circuit almost has unique labels, but for the fact that certain gates computing symmetric functions might not have injective labels. For each offending gate g and each $h \in W(\cdot, g)$ that has t wires to g we add in a sequence of $t-1$ single-input AND-gates and replace $t-1$ wires from h to g with wires from each of these AND-gates to g. This construction gives the following result.

▶ **Lemma 9.** *There is an algorithm that takes as input a $(\mathbb{B}, \rho)$-transparent circuit C and outputs a $(\mathbb{B} \cup \mathbb{B}_{\boldsymbol{std}}, \rho)$-circuit C' such that C and C' compute the same function, C' has unique labels, and if C is symmetric then C' is symmetric. Moreover, this algorithm runs in time polynomial in the size of the input circuit.*

We have that transparent circuits can be transformed into circuits with unique labels. We should like to show that circuits with unique labels are analogous to rigid circuits in that

(i) circuits with unique labels have unique extensions and (ii) we can compute the action of an automorphism on a circuit with unique labels in polynomial-time.

Let C be a circuit of order n with unique labels of order n and let $\sigma \in \mathbf{Sym}_n$. We can define $\pi \in \mathbf{Aut}(C)$ as follows. If g is an output or input gate then the image of g is entirely determined by σ. Suppose g is an internal gate, and suppose we have constructed π for all gates h of depth greater than g. We start from the input gates and inductively construct a gate g' that, from Lemma 6, must be syntactically-equivalent to the image of g under π. We notice that, since C has unique-labels, there is at most one child of $\pi(h)$ syntactically-equivalent to g'. We can compute which child using Lemma 8, and we assign $\pi(g)$ to be this child. The above construction can be implemented as a polynomial-time algorithm, with the additional requirement that we halt and output that no automorphism exists if at any stage the construction fails. It is also important to note that at each point in this inductive definition there is always a unique extension of the automorphism to the next layer of gates. We thus have the following two results.

▶ **Lemma 10.** *If C is a circuit with unique labels then C has unique extensions.*

▶ **Lemma 11.** *There is an algorithm takes as input a $(\mathbb{B}, \rho)$-circuit C of order n with unique labels and $\sigma \in \mathbf{Sym}_n$ and outputs for each gate g the image of g under the action of the unique automorphism extending σ (if it exists). This algorithm runs in time polynomial in the combined size of the input circuit and the encoding of the permutation.*

It remains to use our framework to define a class of circuits with gates that can compute rank. Let $a, b, r, p \in \mathbb{N}$, with p prime. Let $\mathrm{RANK}^r_p[a, b] : \{0,1\}^{[a]\times[b]} \to \{0,1\}$ be the (isomorphism-invariant) structured function with universe $[a] \uplus [b]$, and such that $\mathrm{RANK}^r_p[a, b](M) = 1$ if, and only if, the matrix $M \in \{0,1\}^{[a]\times[b]}$ has rank at least r over $\mathbb{F}_p$ when the entries of M are interpreted as elements of $\mathbb{F}_p$. Let $\mathrm{RANK} = \{\mathrm{RANK}^r_p[a, b] : a, b, r, p \in \mathbb{N},\ p \text{ prime}\}$ and let the *rank basis* be $\mathbb{B}_{\mathbf{rk}} := \mathbb{B}_{\mathbf{maj}} \cup \mathrm{RANK}$. We call a circuit defined over the rank basis a *rank-circuit*.

We are now ready to state the main theorem of this paper.

▶ **Theorem 12** (Main Theorem). *A graph property is decidable by a* P*-uniform family of transparent symmetric rank-circuits if, and only if, it is definable by an* FPR *sentence.*

4 Symmetry and Supports

In this section we introduce the definition of a support and supporting partition from [1] and extend the results about supports to our framework.

▶ **Definition 13.** Let $G \leq \mathbf{Sym}_n$ and let $S \subseteq [n]$. Then S is a *support* for G if $\mathbf{Stab}_n(S) \leq G$.

An important generalisation of the notion of a support is a *supporting partition*.

▶ **Definition 14.** Let $G \leq \mathbf{Sym}_n$ and $\mathcal{P}$ be a partition of $[n]$. Then $\mathcal{P}$ is a *supporting partition* for G if $\mathbf{Stab}_n(\mathcal{P}) \leq G$.

Let $\mathcal{P}$ and $\mathcal{P}'$ be supporting partitions for G. We say that $\mathcal{P}'$ is as *coarse* as $\mathcal{P}$, denoted by $\mathcal{P}' \preceq \mathcal{P}$, if every part in $\mathcal{P}$ is contained in a part in $\mathcal{P}'$. Every group $G \leq \mathbf{Sym}_n$ has a unique coarsest supporting partition [1]. We call this partition the *canonical supporting partition*, and denote it by $\mathbf{SP}(G)$.

It is easy to show that if $\mathcal{P}$ is a supporting partition for $G \leq \mathbf{Sym}_n$ and P is the largest part of $\mathcal{P}$ then $[n] \setminus P$ is a support for G. We say that G has *small support* if there exists

$P \in \mathbf{SP}(G)$ such that $|P| > \frac{n}{2}$, and if G has small support then we call $\mathrm{sp}(G) := [n] \setminus P$ the *canonical support* for G.

We apply the language of supports to circuits. Let C be a symmetric circuit of order n with unique extensions and let g be a gate in C. There is a group action of $\mathbf{Sym}_n$ on the gates of C, given by the isomorphism from $\mathbf{Sym}_n$ to $\mathbf{Aut}(C)$. We say that a partition $\mathcal{P}$ of $[n]$ (resp. a set $S \subseteq [n]$) is a supporting partition (resp. support) for g if $\mathcal{P}$ is a supporting partition for $\mathbf{Stab}(g)$ (resp. S is a support for $\mathbf{Stab}(g)$). We abuse notation and write $\mathbf{SP}(g)$ and $\mathrm{sp}(g)$ for the canonical supporting partition and canonical support for g. Let $\|\mathbf{SP}(g)\|$ denote the smallest value of $|[n] \setminus P|$ for $P \in \mathbf{SP}(g)$. Let $\mathbf{SP}(C)$ denote the largest value of $\|\mathbf{SP}(g)\|$ for g a gate in C. We now state the support theorem and then discuss its proof.

▶ **Theorem 15.** *For any ϵ and n such that $\frac{2}{3} \leq \epsilon \leq 1$ and $n \geq \frac{128}{\epsilon^2}$, if C is a symmetric circuit of order n with unique labels and $s := \max_{g \in C} |\boldsymbol{Orb}(g)| \leq 2^{n^{1-\epsilon}}$, then $\boldsymbol{SP}(C) \leq \frac{33}{\epsilon} \frac{\log s}{\log n}$.*

The proof follows a strategy broadly similar to the one used in [1], and makes use of two lemmas from there. The first lemma gives us that if the index of a group $G \leq \mathbf{Sym}_n$ is small then $\mathbf{SP}(G)$ either has very few or very many parts. The second lemma gives us that for $G \leq \mathbf{Sym}_n$, if $\mathbf{SP}(G)$ has very few parts then it must have a single very large part (and hence a small canonical support). These two results allow us to conclude that every gate g in C has a small canonical support if it has a canonical supporting partition with very few parts. We then prove by structural induction that the canonical supporting partition of every gate has few parts. To be precise, we show that if g is the topologically first gate in the circuit with a canonical supporting partition with too many parts then $|\mathbf{Orb}(g)| > 2^{n^{1-\epsilon}}$, i.e. the orbit is larger than the given bound.

We do this by establishing the existence of a large set H of permutations that each take g to a different gate. To construct H we define a set of triples of the form (σ, h, h') where $\sigma \in \mathbf{Sym}_n$ and $h, h' \in H_g$. Each of these triples is useful in a sense that it guarantees that σ moves g. Moreover, the triples are pairwise independent which means that we can compose them in arbitrary combinations to generate new permutations moving g, while guaranteeing that each such combination gives us a different element in the orbit of g. We have the following as an immediate consequence of the support theorem.

▶ **Lemma 16.** *Let $\mathcal{C} := (C_n)_{n \in \mathbb{N}}$ be a polynomial-size family of symmetric circuits with unique labels. There exists $k \in \mathbb{N}$ such that $\boldsymbol{SP}(C_n) \leq k$ for all $n \in \mathbb{N}$.*

Supports of Indexes

In our analysis we not only need to consider supports for gates but also for elements of the universe of a gate. Let C be a circuit with unique extensions and g be a gate in C. We define an action of $\mathbf{Stab}(g)$ on $\mathrm{unv}(g)$ such that $\sigma \cdot a := (L(g)^{-1}\sigma L(g)(\vec{a}_R))(\vec{a}_R^{-1}(a))$, for $\sigma \in \mathbf{Stab}(g)$ and $a \in \mathrm{unv}(g)$, and where $\vec{a}_R \in \mathrm{ind}(g)$ contains the element a.

Since we have a group action of $\mathbf{Stab}(g)$ on $\mathrm{unv}(g)$, but not $\mathbf{Sym}_n$ on $\mathrm{unv}(g)$, we must speak of the support of $a \in \mathrm{unv}(g)$ relative to $\mathbf{Stab}(g)$. In fact, we are often interested in the action of the subgroup $\mathbf{Stab}(\mathrm{sp}(g))$. We let $\mathbf{Stab}_{\mathrm{sp}g}(a)$ and $\mathbf{Orb}_{\mathrm{sp}g}(a)$ denote the orbit and stabiliser of a under the action of $\mathbf{Stab}(\mathrm{sp}(g))$. We let $\mathrm{sp}_{\mathrm{sp}(g)}(a)$ and $\mathbf{SP}_{\mathrm{sp}(g)}(a)$ denote the canonical support and canonical supporting partition of $\mathbf{Stab}_{\mathrm{sp}(g)}(a)$. In all cases when the choice of g is obvious from context we omit the subscript. The following lemma is a direct consequence of the support theorem and extends the support theorem to the elements of the universe of a gate.

▶ **Lemma 17.** *Let $(C_n)_{n\in\mathbb{N}}$ be a polynomial-size family of symmetric circuits with unique labels. There exists $n_0, k \in \mathbb{N}$ such that for all $n > n_0$, g a gate C_n and $a \in unv(g)$ we have that (i) $\textbf{Stab}_{sp(g)}(a)$ and $\textbf{Stab}_n(g)$ have small support, (ii) if $h \in H_g$ and a appears in $L(g)^{-1}(h)$ then $sp_{sp(g)}(a) \subseteq sp(g) \cup sp(h)$, and (iii) $|sp(g)| \leq k$ and $|sp_{sp(g)}(a)| \leq 2k$.*

5 The Translation from Formulas into Circuits

The standard translation from formulas to families of symmetric circuits does not result in a family of *transparent* circuits. We must thus define a novel translation. We do this in two parts. We first define a translation from P-uniform families of bounded-width FO+rk-formulas to equivalent P-uniform families of transparent symmetric rank-circuits. We then define a translation from formulas of FPR to P-uniform families of bounded-width FO+rk-formulas. The first of these translations is given by the following lemma.

▶ **Lemma 18.** *There is a function that takes as input an* FO+rk*-formula $\theta(\vec{x})$ and $n \in \mathbb{N}$ and outputs a transparent symmetric rank-circuit C of order n defined over the same vocabulary as $\theta(\vec{x})$ and that computes the query defined by $\theta(\vec{x})$ for structures of size n. Moreover, this function is computable and there is a polynomial p such that for an input $(\theta(\vec{x}), n)$ the algorithm computing this function terminates in at most $p(n^{\mathrm{width}(\theta)}|\mathrm{cl}(\theta)|)$ many steps.*

Proof Sketch. The proof follows easily once one understands why the usual translation does not produce a transparent circuit. Consider the following example. Suppose $\psi(\vec{y})$ is a subformula of $\theta(\vec{x})$ of the form $\mathbf{rk}_p^r \vec{w}\vec{z}.\phi$ and suppose that $\phi := \phi'(w_1) \wedge \phi'(w_2)$. In this case the syntactic structure of ϕ is fixed by any permutation of the variables that fixes $\{w_1, w_2\}$ setwise. The usual translation to circuits would preserve symmetries of this form, resulting in many of the input gates of the rank gate being syntactically-equivalent.

In order to address this we first preprocess the formula $\theta(\vec{x})$, defining a new formula $\lambda(\vec{x})$ that decides the same query but is not invariant (in the sense alluded to above) under permutations of the variables. We define $\lambda(\vec{x})$ as follows. Let R be a relation symbol in the vocabulary of $\theta(\vec{x})$ (if the vocabulary is empty the translation is trivial). For a variable y let $\textsc{no-op}(y) := (R(y, y) \vee (\neg R(y, y)))$. For a sequence of variables $\vec{y} = (y_1, \dots, y_m)$ let $\textsc{tag}(\vec{y}) := (\textsc{no-op}(y_1) \wedge (\textsc{no-op}(y_2) \wedge (\textsc{no-op}(y_2) \wedge (\dots \wedge (\textsc{no-op}(y_m)) \dots))))$. Let $\lambda(\vec{x})$ be the formula constructed from $\theta(\vec{x})$ by replacing each sub-formula $\psi(\vec{y})$ of the form $\mathbf{rk}_p^r \vec{w}\vec{z}.\phi$ with the formula $\mathbf{rk}_p^r \vec{w}\vec{z}.((\forall u.u = u) \wedge \phi) \wedge \textsc{tag}(\vec{w} \cup \vec{z})$. Since we always replace a subformula ϕ with a logically equivalent formula, it follows that $\lambda(\vec{x})$ and $\theta(\vec{x})$ are equivalent. The intuition is that $\textsc{tag}(\vec{w} \cup \vec{z})$ appends a tower of conjunctions of tautologies, with each tautology referencing a unique variable from $\vec{w} \cup \vec{z}$. When we construct the circuit, this tower of tautologies acts to 'tag' each input to the rank gate with a unique gadget.

We now construct C using the usual approach. For each subformula $\psi(\vec{y})$ of $\lambda(\vec{x})$ and assignment $\vec{a} \in [n]^{|\vec{y}|}$ to $\vec{y}$ we include a gate $g_{\psi,\vec{a}}$ in C. We wire the circuit such that $g_{\phi,\vec{a}}$ is an input gate to $g_{\psi,\vec{b}}$ iff ϕ is an immediate subformula of ψ and the two assignments never assign the same variable to two different values. For a complete proof see [7]. ◀

The translation from FPR to P-uniform families of bounded-width FO+rk-formulas is a concatenation of the following two translations. First, from [5], we can translate $\theta(\vec{x}) \in \mathrm{FPR}[\tau]$ into an equivalent P-uniform family of FOR$[\tau]$-formulas. Second, from [14], we can translate FOR$[\tau]$-formulas into equivalent P-uniform families of FO+rk$[\tau]$-formulas. Both of these translations increase the width by a constant factor, and so we may apply Lemma 18 to prove the following.

▶ **Theorem 19.** *For each* FPR*-formula* $\theta(\vec{x})$ *there exists a* P*-uniform family of transparent symmetric rank-circuits*$(C_n)_{n\in\mathbb{N}}$ *that defines the same query as* $\theta(\vec{x})$*.*

6 The Translation from Circuits into Formulas

We leverage the support theorem and the various polynomial-time algorithms defined for transparent circuits and circuits with unique labels in order to define a translation from P-uniform families of symmetric rank-circuits to formulas of FPR. Let $\mathcal{C} = (C_n)_{n\in\mathbb{N}}$ denote a P-uniform family of transparent symmetric $(\mathbb{B}_{\mathbf{rk}}, \rho)$-circuits computing a q-ary query Q.

From the Immerman-Vardi theorem [12, 16] and Lemma 9, there is a t-width interpretation Φ such that for each ρ-structure $\mathcal{A}$ of size n the interpretation of Φ in $\mathcal{A}$ defines a symmetric rank-circuit with unique labels (in the number universe) equivalent to C_n. We aim to show that there exists $\theta_Q \in \text{FPR}[\rho]$ that defines Q, i.e. such that $\mathcal{A} \models \theta_Q[\vec{a}]$ if, and only if, $C_n[\gamma\mathcal{A}](\Omega(\gamma\vec{a})) = 1$ for any bijection $\gamma \in [n]^{\underline{U}}$.

Let n_0 and k be the constants in the statement of Lemma 17. Notice that for each $n \leq n_0$ there are only constantly many bijections from the universe of a structure to $[n]$, and so we can explicitly quantify over these constantly many bijections and evaluate the circuit. We thus fix $n > n_0$ and a ρ-structure $\mathcal{A}$ with universe U of size n and show how to evaluate C_n

It follows from Lemma 17 that each gate g has a support of size at most k and each $a \in \text{unv}(g)$ has a support of size at most $2k$. We say that two injections f and g are *compatible* if there is an injection on the union of their domains that agrees with both functions. If there is such a function we denote it by $(f|g)$. We use $\sim$ to denote compatibility. The following result gives us that the evaluation of a gate g for a bijection $\gamma \in [n]^{\underline{U}}$ depends only on those elements γ maps to $\text{sp}(g)$.

▶ **Lemma 20.** *Let g be a gate in C_n. Let $\eta \in U^{\underline{\text{sp}(g)}}$ and $\gamma_1, \gamma_2 \in [n]^{\underline{U}}$ such that $\gamma_1^{-1} \sim \eta$ and $\gamma_2^{-1} \sim \eta$. Then $L^{\gamma_1\mathcal{A}}(g)$ and $L^{\gamma_2\mathcal{A}}(g)$ are isomorphic.*

It follows from Lemma 20 that the evaluation of g is entirely determined by $\text{EV}_g := \{\eta \in U^{\underline{\text{sp}(g)}} : \exists\gamma \in [n]^{\underline{U}} \text{ s.t. } C_n[\gamma\mathcal{A}](g) = 1 \text{ and } \eta \sim \gamma^{-1}\}$. Here we see how the support theorem allows us to characterize the evaluation of a gate succinctly.

The query defined by C_n for $\mathcal{A}$ is $Q = \{\vec{a} \in U^q : \exists g \in G, \eta \in \text{EV}_g \text{ s.t. } \Omega(\eta^{-1} \circ \vec{a})) = g\}$. In order to define Q it is thus sufficient to show that EV_g is FPR-definable. In particular, we show that there is an FPR-definable relation $V \subseteq [n^t] \times U^k$ such that $(g, \vec{x}) \in V$ if, and only if, the assignment that maps $\text{sp}(g)$ to the first $|\text{sp}(g)|$ elements of $\vec{x}$ is in EV_g. We do this by first describing a procedure for recursively defining EV_g, i.e. defining EV_g given $\{\text{EV}_h : h \in H_g\}$, and then arguing that this definition can be implemented in FPR. This suffices as we may then use the fixed-point operator to complete the definition of V. The gate g is either a symmetric gate or a rank gate. If g is a symmetric gate then we have a FPC-definable recursive construction of EV_g from [1]. As such, we assume g is a rank gate.

As an aside, we note that the recursive construction of EV_g in [1] relies on the fact that if g is symmetric then it can be evaluated by counting the number of its inputs that evaluate to 1. Using this fact, along with a bijection between the orbit of a gate and the assignments to the support of that gate, the problem of evaluating g reduces to a counting problem on the assignments to the supports of the inputs to g. The results that underlie this counting argument fail for non-symmetric gates, and so we are forced to use a very different approach for rank gates.

We instead show that for each gate g and $\eta \in U^{\underline{\text{sp}(g)}}$ there is an FPR-definable matrix M that has the same rank as $L^{\gamma\mathcal{A}}(g)$ for any $\gamma \in [n]^{\underline{U}}$ such that $\gamma^{-1} \sim \eta$. We can then check if $\eta \in \text{EV}_g$ by applying the rank operator to M and testing against the threshold.

We introduce some notation. Let $A \times B := \text{ind}(g)$. For $h \in H_g$ let $\text{row}(h) := L(g)^{-1}(h)(1)$ and $\text{col}(h) := L(g)^{-1}(h)(2)$. Let $A_h := \{\vec{x} \in U^{\underline{\text{sp}(h)}} : \eta \sim \vec{x}\}$ and for all $a \in \text{unv}(g)$ let $A_a = \{\vec{x} \in U^{\underline{\text{sp}(a)}} : \eta \sim \vec{x}\}$.

We first define the index sets for the matrix M. Let $R^{\min} := \{\min(\textbf{Orb}(\text{row}(h))) : h \in H_g\}$ and $C^{\min} := \{\min(\textbf{Orb}(\text{col}(h))) : h \in H_g\}$. Let $I := \{(i, \vec{x}) : i \in R^{\min}, \vec{x} \in A_i\}$ and $J := \{(j, \vec{y}) : j \in C^{\min}, \vec{y} \in A_j\}$. We think of $R^{\min}$ and $C^{\min}$ as indexing the orbits of the row and column elements under the action of $\textbf{Stab}(\text{sp}(g))$, with each orbit indexed by the minimal element in A (or B, respectively) that appears in it. We think of I and J as indexing the elements within an orbit instead by elements of A_i and A_j, implicitly using the bijection between these sets and the orbits of $\text{row}(h)$ and $\text{col}(h)$.

We associate with each index $((i, \vec{x}), (j, \vec{y})) \in I \times J$ a gate h and an assignment $\vec{w}$ to the support of h as follows. It can be shown there is a function that maps a given index to a permutation $\sigma \in \textbf{Stab}(g)$ such that $\vec{y}\sigma$ is compatible with both η and $\vec{x}$ (see [7] for details). Let $h = L(g)(i, \sigma j)$ and let $\vec{w} = (\vec{x}|\vec{y}\sigma)$. We define the matrix $M : I \times J \to \{0, 1\}$ by $M((i, \vec{x}), (j, \vec{y})) := \vec{w} \in \text{EV}_h$.

Let x be a gate in H_g or an element of the universe of g. Let $f \in U^{\underline{\text{sp}(x)}}$ and $\gamma \in [n]^{\underline{U}}$ such that $\gamma^{-1} \sim \eta$. Let $\Pi_f^\gamma \in \textbf{Stab}(\text{sp}(g))$ be such that $\Pi_f^\gamma(a) = \gamma(f(a))$ for all $a \in \text{sp}(x)$. It is easy to see that $\Pi_f^\gamma(x)$ is well-defined. For a fixed $h \in H_g$, the mapping $\vec{z} \mapsto \Pi_{\vec{z}}^\gamma(h)$, for $\vec{z} \in A_h$, establishes a correspondence between A_h and the orbit of h. A similar correspondence exists for a fixed $a \in \text{unv}(g)$. It follows that $\vec{z} \in \text{EV}_h$ if, and only if, $C_n[\gamma\mathcal{A}](\Pi_{\vec{z}}^\gamma(h)) = 1$. [7]

We use this correspondence to define a mapping from M to $L^{\gamma\mathcal{A}}(g)$. Let $\alpha^\gamma : I \to A$ and $\beta^\gamma : J \to B$ be defined by $\alpha^\gamma(i, \vec{x}) := \Pi_{\vec{x}}^\gamma(i)$ and $\beta^\gamma(j, \vec{y}) := \Pi_{\vec{y}}^\gamma(j)$, respectively. It is possible to show that $(\alpha^\gamma, \beta^\gamma)$ is a surjective homomorphism from M to $L^{\gamma\mathcal{A}}(g)$. It can be shown that $\alpha^\gamma(i, \vec{x}) = \alpha^\gamma(i, \vec{x}')$ if, and only if, there exists $\pi \in \textbf{Stab}_{\text{sp}(g)}(i)$ such that $\vec{x} = \vec{x}'\pi$ – and a similar result holds for β^γ. It follows that $(\alpha^\gamma, \beta^\gamma)$ is not, in general, injective.

We resolve this problem by quotienting. Let $s \in \text{unv}(g)$ and $\vec{x}, \vec{x}' \in A_s$. We say that $\vec{x} \approx \vec{x}'$ if, and only if, there exists $\pi \in \textbf{Stab}(s)$ such that $\vec{x} = \vec{x}'\pi$. For $(i, \vec{x}), (i', \vec{x}') \in I$ we say that $(i, \vec{x}) \approx (i', \vec{x}')$ if, and only if, $i = i'$ and $\vec{x} \approx \vec{x}'$. We similarly define $\approx$ on J.

It is easy to see that α^γ and β^γ are constant on $\approx$-equivalence classes. As such, the quotient functions $\alpha^\gamma/_\approx$ and $\beta^\gamma/_\approx$ are well-defined. We can also show that $M((i, \vec{x}, (j, \vec{y}))) = M((i', \vec{x}'), (j', \vec{y}'))$ if $(i, \vec{x}) \approx (i', \vec{x}')$ and $(j, \vec{y}) \approx (j', \vec{y}')$. Let $M_\approx : I/_\approx \times J/_\approx \to \{0, 1\}$ be defined by $M_\approx((i, [\vec{x}])_\approx, (j, [\vec{y}])_\approx) := M((i, \vec{x}), (j, \vec{y}))$. It follows from the previous observation that this function is well-defined.

Since $(\alpha^\gamma, \beta^\gamma)$ is a surjective homomorphism, $(\alpha^\gamma/_\approx, \beta^\gamma/_\approx)$ is a surjective homomorphism from $M_\approx$ to $L^{\gamma\mathcal{A}}(g)$. Moreover, it follows from the previous comment on the failure of injectivety that $(\alpha^\gamma/_\approx, \beta^\gamma/_\approx)$ is an injection. We thus have the following result.

▶ **Theorem 21.** *Let $\gamma \in [n]^{\underline{U}}$ such that $\gamma^{-1} \sim \eta$. Then $L^{\gamma\mathcal{A}}(g)$ is isomorphic to $M_\equiv$.*

It is not hard to show that the rows $M((i, \vec{x}), \cdot)$ and $M((i', \vec{x}'), \cdot)$ are equal if $(i, \vec{x}) \approx (i'\vec{x}')$, and so $\textbf{rk}_p(M) = \textbf{rk}_p(M_\equiv)$. From this and Theorem 21 we have the following result.

▶ **Lemma 22.** *Let $\gamma \in U^{\underline{n}}$ be such that $\gamma^{-1} \sim \eta$ and let $p \in \mathbb{N}$ be prime. Then $\boldsymbol{rk}_p(M) = \boldsymbol{rk}_p(M_\equiv) = \boldsymbol{rk}_p(L^{\gamma\mathcal{A}}(g))$.*

It remains to justify our assertion that the above recursive definition of EV_g can be implemented in FPR. It is sufficient to show that there is an FPR-formula that defines M for a rank gate g and assignment $\eta \in U^{\underline{\text{sp}(g)}}$. We first show that the sets $\{(g, \text{sp}(g)) : g \in G\}$, I, and J are FPR-definable. We have the following results as a consequence of Lemma 11.

▶ **Lemma 23.** *There is an algorithm that takes in a circuit C with unique labels and outputs if the circuit is symmetric. If it is symmetric then it outputs for each gate g and $a \in unv(g)$ the orbit $\boldsymbol{Orb}(g)$ and canonical supporting partition $\boldsymbol{SP}(g)$, as well as $\boldsymbol{Orb}_{sp(g)}(a)$ and $\boldsymbol{SP}_{sp(g)}(a)$. This algorithm runs in time polynomial in the size of the circuit.*

From Lemma 23 and the Immerman-Vardi theorem there are FPC-formulas that define the canonical support and orbit for each gate g and each $a \in \text{unv}(g)$. Moreover, it can be shown that compatibility between assignments to supports is FPR-definable. It follows that we can define A_a for each $a \in \text{unv}(g)$ and A_h for each $h \in H_g$. Combining these results we have that I and J are FPR-definable. We then define M using a relation symbol V' that denotes the value of V at a given stage in the recursive construction. This completes the FPR-definition of M and so EV_g, and hence the proof of our main result.

7 Concluding Remarks and Future Work

FPR is one of the most expressive logics we know that is still contained in P and understanding its expressive power is an important question. The main result of this paper establishes an equivalence between the expressive power of FPR and the computational power of uniform families of transparent symmetric rank-circuits. Not only does this establish an interesting characterization of an important logic, it also deepens our understanding of the connection between logic and circuit complexity and sheds new light on foundational aspects of the circuit model.

The circuit characterisation helps emphasise certain important aspects of the logic. Given that P-uniform families of invariant circuits (without the restriction to symmetry) express all properties on P, we can understand the inability of FPC (and, conjecturally, FPR) to express all such properties as essentially down to symmetry. As with other (machine) models of computation, the translation to circuits exposes the inherent combinatorial structure of an algorithm. In the case of logics, we find that a key property of this structure is its symmetry and the translation to circuits provides us with the tools to study it.

Still, the most significant contribution of this paper is not in the main result but in the techniques that are developed to establish it, and we highlight some of these now. The conclusion of [1] says that the support theorem is "largely agnostic to the particular [...] basis", suggesting that it could be easily adapted to include other gates. This turns out to have been a misjudgment. Attempting to prove the support theorem for a basis that includes rank threshold gates showed us the extent to which both the proof of the theorem and, more broadly, the definitions of circuit classes, rest heavily on the assumption that all functions computed by gates are symmetric. Thus, in order to define what the "symmetry" condition might mean for circuits that include rank threshold gates, we radically generalise the circuit framework to allow for gates that take structured inputs (rather than sets of 0s and 1s) and are invariant under isomorphisms. This leads to a refined notion of circuit automorphism, which allows us to formulate a notion of symmetry and prove a version of the support theorem. Again, in that proof, substantial new methods are required.

The condition of *transparency* makes the translation of uniform circuit families into formulas of logic (which is the difficult direction of our characterisation) possible, but it complicates the other direction. Indeed, the natural translation of formulas of FPR into uniform circuit families yields circuits which are symmetric, but not transparent. This problem is addressed by introducing gadgets in the translation – which for ease of exposition, we did in formulas of FO+rk which are then translated into circuits in the natural way. Thus, the restriction to transparent circuits is sufficient to get both directions of the characterisation.

In short, we can represent the proof of our characterisation through the three equivalences in this triangle.

$$\begin{tikzcd}
\text{FPR} \arrow[r, equal] \arrow[dr, equal] & \text{Uniform families of bounded-width FO+rk formulas} \arrow[d, equal] \\
 & \text{Uniform families of transparent symmetric rank-circuits}
\end{tikzcd}$$

This highlights another interesting aspect of our result. The first translation, of FPR to uniform families of FO+rk formulas was given in [5] and used there to establish arity lower bounds. However, this was for a weaker version of the rank logic rather than the strictly more expressive one defined by Grädel and Pakusa [9]. The fact that we can complete the cycle of equivalences with the more powerful logic demonstrates that the definition of Grädel and Pakusa is the "right" formulation of FPR.

Future Work

There are many directions of work suggested by the methods and results developed in this paper. First of all, there is the question of transparency. We introduce it as a technical device that enables our characterisation to go through. Could it be dispensed with? Or are P-uniform families of transparent symmetric rank-circuits strictly weaker than families without the restriction of transparency?

The framework we have developed for working with circuits with structured inputs is very general and not specific to rank gates. It would be interesting to apply this framework to other logics. It appears to be as general a way of extending the power of circuits as Lindström quantifiers are in the context of logic. We would like to develop this link further, perhaps for specific quantifiers such as FP extended by an operator that expresses the solubility of systems of equations over rings as in [4]

At the moment, we have little by way of methods for proving inexpressibility results for FPR, whether we look at it as a logic or in the circuit model. The logical formulation lays emphasis on some parameters (the number of variables, the arity of the operators, etc.) which we can treat as resources against which to prove lower bounds. On the other hand, the circuit model brings to the fore other, more combinatorial, parameters. One such is the fan-in of gates and a promising and novel approach is to try and prove lower bounds for symmetric circuits with gates with bounded fan-in. We might ask if it is possible to compute AND[3] using a symmetric circuit with gates that have fan-in two. Perhaps we could also combine the circuit view with lower-bound methods from logic, such as pebble games. Dawar [3] has shown how the bijection games of Hella [11] can be used directly to prove lower bounds for symmetric circuits without reference to the logic. We also have pebble games for FPR [6], and it would be interesting to know if we can use these on circuits and how the combinatorial parameters of the circuit interact with the game.

Finally, we note that some of the interesting directions on the interplay between logic and symmetric circuits raised in [1] remain relevant. Can we relax the symmetry condition to something in between requiring invariance of the circuit under the full symmetric group (the case of symmetric circuits) and requiring no invariance condition at all? Can such restricted symmetries give rise to interesting logics in between FPR and P? It also remains a challenge to find a circuit characterisation of CPTC. Could the general framework for non-symmetric gates we have developed here help in this respect?

References

1 M. Anderson and A. Dawar. On symmetric circuits and fixed-point logics. *Theory of Computing Systems*, 60(3):521–551, 2017.

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

3 A. Dawar. On symmetric and choiceless computation. In Mohammad Taghi Hajiaghayi and Mohammad Reza Mousavi, editors, *Topics in Theoretical Computer Science*, pages 23–29, Cham, 2016. Springer International Publishing.

4 A. Dawar, E. Grädel, B. Holm, E. Kopczynski, and W. Pakusa. Definability of linear equation systems over groups and rings. *Logical Methods in Computer Science*, 9(4), 2013.

5 A. Dawar, M. Grohe, B. Holm, and B. Laubner. Logics with rank operators. In *2009 24th Annual IEEE Symposium on Logic In Computer Science (LICS)*, pages 113–122, 2009.

6 A. Dawar and B. Holm. Pebble games with algebraic rules. In Artur Czumaj, Kurt Mehlhorn, Andrew Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming*, pages 251–262, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

7 A. Dawar and G. Wilsenach. Symmetric circuits for rank logic. *arXiv*, 2018. `arXiv:1804.02939`.

8 L. Denenberg, Y. Gurevich, and S. Shelah. Definability by constant-depth polynomial-size circuits. *Information and Control*, 70(2):216–240, 1986.

9 E. Grädel and W. Pakusa. Rank logic is dead, long live rank logic! In *2015 24th Annual Conference on Computer Science Logic, (CSL)*, pages 390–404, 2015.

10 M. Grohe. *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory*. Lecture Notes in Logic. Cambridge University Press, 2017. URL: `https://books.google.co.uk/books?id=RLYrDwAAQBAJ`.

11 L. Hella. Logical hierarchies in ptime. *Information and Computation*, 129(1):1–19, 1996.

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

13 N. Immerman. *Descriptive Complexity*. Graduate texts in computer science. Springer New York, 1999.

14 L. Libkin. *Elements of Finite Model Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer Berlin Heidelberg, 2004.

15 M. Otto. The logic of explicitly presentation-invariant circuits. In *1996 10th International Workshop, Annual Conference on Computer Science Logic (CSL)*, pages 369–384. Springer, Berlin, Heidelberg, 1997.

16 M. Vardi. The complexity of relational query languages (extended abstract). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 137–146, New York, NY, USA, 1982. ACM.