Symmetric Circuits with Non-Symmetric Gates

Anuj Dawar

Department of Computer Science and Technology, University of Cambridge

based on joint work with Gregory Wilsenach

HellaFest, Murikanranta, 5 July 2018

Lauri Hella and Generalized Quantifiers

Workshop on *Finite Model Theory and Databases*, San Diego, 1992.

Anuj Dawar and Lauri Hella: "*The Expressive Power of Finitely Many Generalized Quantifiers*", LICS 1994, Inf. Comp 1995.

Workshop on *Finite Model Theory*, Helsinki 1994.

British Council/CIMO Academic Cooperation Grant, 1997–98.

FPC and Symmetric Circuits

FPC—*Fixed Point Logic with Counting* is a reference logic in descriptive complexity theory. It captures a large and natural fragment of *polynomial-time computable* properties.

(Anderson, D. 2014/7) give a characterization of FPC in terms of *symmetric circuits*.

Circuits

A circuit C is a *directed acyclic graph* with:

- source nodes (called *inputs*) labelled $x_1, \ldots, x_n$;
- any other node (called a *gate*) with k incoming edges is labelled by a Boolean function $g : \{0,1\}^k \to \{0,1\}$ from some fixed basis (*e.g.* AND/OR/NOT);
- some gates designated as *outputs*, $y_1, \ldots, y_m$.

C computes a function $f_C : \{0,1\}^n \to \{0,1\}^m$ as expected.

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 may be given by a *circuit* C_n made up of Boolean gates, with n Boolean inputs and one output.

If the size of C_n is bounded by a polynomial in n, the language L is in the class $\mathrm{P/poly}$.

If, in addition, the function $n \mapsto C_n$ is computable in polynomial time, L is in P.

Circuit Complexity Classes

For the definition of $\mathrm{P/poly}$ and P, it makes no difference if the circuits only use $\{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$ or a richer basis with *ubounded fan-in; threshold; or counting gates*.

However,

> *AC_0 — languages accepted by bounded-depth, polynomial-size families of circuits with unbounded fan-in AND and OR gates and NOT gates;*

and

> *TC_0 — languages accepted by bounded-depth, polynomial-size families of circuits with unbounded fan-in AND and OR and threshold gates and NOT gates;*

are different.

A *threshold gate* $\mathsf{Th}_t^k : \{0,1\}^k \to \{0,1\}$ evaluates to 1 iff at least t of the inputs are 1.

Symmetric Functions

We say a function $g\{0,1\} \to \{0,1\}$ is *symmetric* if its value is invariant under *all* permutations of the k inputs.

k-input AND, OR and theshold gates all evaluate symmetric functions, as do *majority gates*.

Since a circuit C is a *DAG*, rather than, say, an *ordered DAG*, it is important that the labels on gates are symmetric functions.

Invariant Circuits

Instead of a language $L \subseteq \{0,1\}^*$, consider a class $\mathcal{C}$ of directed *graphs*. This can be given by a family of *Boolean functions*:

$$(f_n)_{n\in\omega} : \{0,1\}^{n^2} \to \{0,1\}.$$

A graph on vertices $\{1,\ldots,n\}$ has n^2 *potential* edges.
So the graph can be treated as a string in $\{0,1\}^n$.

Since $\mathcal{C}$ is closed under isomorphisms, each function f_n is invariant under the natural action of S_n on n^2.

We call such functions *graph invariant*.

Symmetric Circuits

More generally, for any *relational vocabulary* τ, let

$$\tau(n) = \sum_{R \in \tau} n^{\mathsf{arity}(R)}$$

We take an encoding of n-element τ-structures as strings in $\{0,1\}^{\tau(n)}$ and this determines an action of S_n on such strings.

A function $f : \{0,1\}^{\tau(n)} \to \{0,1\}$ is τ*-invariant* if it is invariant under this action.

We say that a circuit C with inputs labelled by $\tau(n)$ is *symmetric* if every $\pi \in S_n$ acting on the inputs of C can be extended to an *automorphism* of C.

Every symmetric circuit computes an invariant function, but the converse is false.

Formulas to Circuits

Any formula of *first-order logic* translates into a uniform family of *constant-depth*, *polynomial-size symmetric* Boolean circuits.

> *For each subformula $\psi(\overline{x})$ and each assignment $\overline{a}$ of values to the free variables, we have a gate.*
> *Existential quantifiers translate to big disjunctions, etc.*

Any formula φ of FP translates into a uniform family of polynomial-size *symmetric* Boolean circuits.

> *For each n, φ translates into a first-order formula of depth polynomial in n and with a constant bound k on the number of free variables in a sub-formula.*

Any formula of FPC translates into a uniform family of polynomial-size *symmetric* threshold (or majority) circuits.

Symmetric Circuits and Bases

Theorem (Anderson-D.)
A class of structures is definable in FPC if, and only if, it is decided by a P-uniform family of symmetric circuits, using AND, OR, and *majority* gates.

The gates are *unbounded fan-in*.

It is important that we have *majority* or *threshold* gates. Having only the standard Boolean functions gives us something strictly weaker than FPC.

Adding further *symmetric functions* to the basis does not further increase the expressive power of such symmetric circuit families.

Support Theorem

A key technical took in the proof is the *support theorem*.

Say a set $X \subseteq [n]$ is a *support* of a group $G \leq S_n$ if the pointwise stabilizer of X is included in G.

For a symmetric circuit C with automorphism group S_n, we say that $X \subseteq [n]$ is a *support* of a gate g iff it is a support of the stabilizer of g.

Support Theorem: If $(C_n)_{n \in \omega}$ is a P-uniform family of symmetric circuits, then there is a k such that every $g \in C_n$ has a support of size at most k.

FPrk

FPrk is *fixed-point logic with rank*.

This properly extends the expressive power of FPC while still being inside P.

The logic has *rank operators* which allow us to define the rank of a matrix over a finite field.

For our purposes, it is sufficient that every formula of FPrk translates, over structures of size n to a formula of first-order logic extended with *rank quantifiers*, using a constant number of variables.

Rank quantifier:

$$\mathrm{rk}(p, t, x, y)\varphi$$

is true if the 0-1-matrix (interpreted over the finite field $\mathbb{F}_p$) defined by $\varphi(x, y)$ has rank at least t.

Circuits with Rank Gates

Define *rank gates* as Boolean functions:

$$\mathrm{rk}_p^t : \{0,1\}^{m\times n} \to \{0,1\}$$

where the result is 1 if the input, seen as an $m \times n$ matrix over $\mathbb{F}_p$ has rank at least t.

We want to translate formulas of FPrk to circuits using such gates.

Note that such a function is *not symmetric*.

We have to put more structure on the circuit than just a *directed acyclic graph*.

Circuits for FPrk

In **(D., Wilsenach 2018)**, we

- generalize the notion of circuit to allow such *non-symmetric* gates;
- define the notion of *symmetric circuits* in this more general context; and
- give a circuit characterizaton of FPrk.

τ-invariant gates

In general, we consider a *multi-sorted* vocabulary τ with sorts $U_1, \ldots, U_l$ and relations $R_1, \ldots, R_m$, each with a type $i_1, \ldots, i_r$ with $i_j \in [l]$.

This defines a polynomial $\tau k_1, \ldots, k_l$ which gives the length of a string encoding a structure in which the sorts of sizes $k_1, \ldots, k_l$.

A function $g : \{0,1\}^{\tau(k_1,\ldots,k_l)} \to \{0,1\}$ is *τ-invariant* if it is invarinant under the natural action of $S_{k_1} \times \cdots \times S_{k_l}$ on the strings.

Circuits with τ-invariant gates

We consider circuits with gates that compute τ-invariant functions.

Now, the structure of the circuit is not simply a DAG.

A gate computing a τ-invariant function must have its incoming edges labelled with the elements that make up $\tau(k_1, \ldots, k_l)$.

We also need to refine the notion of *automorphism* of a circuit. It must not only preserve the graph structure, but when it takes g to g', it needs to preserve the τ-structure on the children of g.

With this, we can define the notion of a *symmetric circuit* again, as one where every permutation in S_n extends to an automorphism of the circuit.

Circuits for Logics with Generelized Quantifiers

A *generalized quantifier* Q now translates into a natural family of gates g_Q (one for each input size).

And, we can easily see that any formula of the logic $\mathsf{FP}(Q)$ gives rise to a family of P-uniform *symmetric* circuits using gates from AND, OR, NOT and g_Q.

Can we get the *converse*?

Translating Circuits to Formulas

The proof from **(Anderson, D.)** translating symmetric circuits to FPC relies on some technical ingredients.

The first is the *support theorem*. The proof in **(Anderson, D.)** relies heavily on the fact that each gate computes a symmetric function.

We are able to prove a more general support theorem using different techniques.

This yields a translation of P-uniform families of symmetric circuits using gates from AND, OR, NOT and g_Q to $L^{\omega}_{\infty\omega}(Q)$.

To get the translation to FP(Q), there is another obstacle to be overcome.

Detecting Circuit Automorphisms

The proof of **(Anderson, D.)** uses the P-uniformity of the circuit family to conclude that important properties of the circuit C_n are *polynomial-time* decidable and therefore expressible in FP on *ordered structures*.

In the more general context, some of these properties are not in P, unless *graph isomorphism* is.

> *For instance, to decide if a given $\pi \in S_n$ extends to an automorphism of C_n may require checking isomorphism of τ-structures at individual gates.*

We get around this by introducing a further restriction of *transparency*.

Transparent Circuits

A circuit C is *transparent* if

- whenever g is a gate evaluating a τ-invariant function, the labelling of the inputs of g by $\tau(k_1, \ldots, k_l)$ is *injective*; and
- whenever g, h are *distinct* τ-invariant gates, the subcircuits below them are *not syntactically identical*.

We can show that any formula of FP(Q) translates to a P-uniform family of symmetric, *transparent* circuits using gates from AND, OR, NOT and g_Q.

And now, we can also show the converse.

This is shown for FPrk in **(D., Wilsenach 2018)** but holds more generally.

Generalized Gates

The translation of *generalized quantifiers* to *generalized gates* suggests a further generalization.

For a group $G \leq S_n$, we say that a function $g : \{0,1\}^n \to \{0,1\}$ is G-invariant if it is invariant under the action of G on its inputs.

> *So, a τ-invariant function is $S_{k_1} \times \cdots \times S_{k_l}$-invariant where we treat this as a subgroup of $S_{\tau(k_1,\ldots,k_l)}$.*

We can define a suitable notion of *automorphism* of circuits where the inputs of G-invariant gates are mapped by G-isomorphisms.

What logics do families of symmetric circuits in this context give rise to?