Logic and Circuit Complexity

Anuj Dawar

University of Cambridge Computer Laboratory

Amsterdam, 23 June 2016

Is There a Logic for P?

The question of whether or not there is a logic expressing exactly the P properties of *(unordered) relational structures* is the central problem in *Descriptive Complexity*.

If we assume structures are *ordered*, then FP, the extension of first-order logic with least fixed points suffices. **(Immerman; Vardi 1982)**

In the absence of order FP fails to express simple cardinality properties such as *evenness*.

Fixed-point Logic with Counting

Immerman had proposed FPC—the extension of FP with a mechanism for *counting*.

Most *"obviously"* polynomial-time algorithms can be expressed in FPC.

This includes P-complete problems such as

> CVP—*the Circuit Value Problem*
> *Input: a circuit, i.e. a labelled DAG with source labels from* $\{0, 1\}$, *internal node labels from* $\{\vee, \wedge, \neg\}$.
> *Decide: what is the value at the output gate.*

CVP is expressible in FP.

It is expressible in FPC for circuits that may include *threshold or counting gates*.

Expressive Power of FPC

Many non-trivial polynomial-time algorithms can be expressed in FPC:

- FPC captures all of P over any *proper minor-closed class of graphs*
  (Grohe 2010)
- FPC can express *linear programming* problems; *max-flow* and *maximum matching* on graphs. **(Anderson, D., Holm 2015)**

But some cannot be expressed:

- There are polynomial-time decidable properties of graphs that are not definable in FPC. **(Cai, Fürer, Immerman, 1992)**
- Solvability of a system of linear equations over a finite field cannot be expressed in FPC. **(Atserias, Bulatov, D. 2009)**

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 computed by a *circuit* C_n made up of

- Gates labeled by Boolean operators: $\wedge, \vee, \neg$,
- Boolean inputs: $x_1, \ldots, x_n$, and
- A distinguished gate 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, 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.

Note: For these classes it makes no difference whether the circuits only use $\{\wedge, \vee, \neg\}$ or a richer basis with *threshold* or *majority* gates.

Circuit Lower Bounds

It is conjectured that $\mathrm{NP} \not\subseteq \mathrm{P/poly}$.

Lower bound results have been obtained by putting further restrictions on the circuits:

- No *constant-depth* (unbounded fan-in), *polynomial-size* family of circuits decides *parity*. **(Furst, Saxe, Sipser 1983)**.
- No *polynomial-size* family of *monotone* circuits decides *clique*. **(Razborov 1985)**.
- No *constant-depth*, $O(n^{\frac{k}{4}})$-*size* family of circuits decides *k-clique*. **(Rossman 2008)**.

No known result separates NP from *constant-depth*, *polynomial-size* families of circuits with *majority gates*.

Circuits for Graph Properties

We want to study families of circuits that decide properties of *graphs* (or other relational structures—for simplicity of presentation we restrict ourselves to graphs).

We have a family of Boolean circuits $(C_n)_{n\in\omega}$ where there are n^2 inputs labelled $(i,j) : i,j \in [n]$, corresponding to the *potential edges*.
Each input takes value 0 or 1;

Graph properties in P are given by such families where:

- the size of C_n is bounded by a polynomial $p(n)$; and
- the family is uniform, so the function $n \mapsto C_n$ is in P (or $\mathrm{DLogTime}$).

Invariant Circuits

C_n is *invariant* if, for every input graph, the output is unchanged under a permutation of the inputs induced by a permutation of $[n]$.

That is, given any input $G : [n]^2 \to \{0, 1\}$, and a permutation $\pi \in S_n$,

C_n accepts G if, and only if, C_n accepts the input πG given

$$(\pi G)(i, j) = G(\pi(i), \pi(j)).$$

Note: this is not the same as requiring that the result is invariant under *all* permutations of the input. That would only allow us to define functions of the *number* of 1s in the input. The functions we define include all *isomorphism-invariant* graph properties such as *connectivity*, *perfect matching*, *Hamiltonicity*, *3-colourability*.

Symmetric Circuits

Say C_n is *symmetric* if any permutation of $[n]$ applied to its inputs can be extended to an automorphism of C_n.

> *i.e., for each $\pi \in S_n$, there is an automorphism of C_n that takes input (i, j) to $(\pi i, \pi j)$.*

Any symmetric circuit is invariant, but *not* conversely.

> *Consider the natural circuit for deciding whether the number of edges in an n-vertex graph is even.*

Any invariant circuit can be converted to a symmetric circuit, but with potentially *exponential blow-up*.

Logic and Circuits

Any formula of φ *first-order logic* translates into a uniform family of circuits C_n

> *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.*

The circuit C_n is:

- of *constant* depth (given by the depth of φ);
- of size at mose $c \cdot n^k$ where c is the number of subformulas of φ and k is the *maximum number of free variables* in any subformula of φ.
- *symmetric* by the action of $\pi \in S_n$ that takes $\psi[\overline{a}]$ to $\psi[\pi(\overline{a})]$.

FP and Circuits

For every sentence φ of FP there is a k such that for every n, there is a formula φ_n of L^k that is equivalent to φ on all graphs with at most n vertices.

The formula φ_n has

- *depth* n^c for some constant c;
- at most k free variables in each sub-formula for some constant k.

It follows that every graph property definable in FP is given by a family of *polynomial-size, symmetric* circuits.

FPC and Counting

For every sentence φ of FP there is a k such that for every n, there is a formula φ_n of C^k that is equivalent to φ on all graphs with at most n vertices.

The formula φ_n has

- *depth* n^c for some constant c;
- at most k free variables in each sub-formula for some constant k.

It follows that every graph property definable in FP is given by a family of *polynomial-size, symmetric* circuits in a basis with *threshold gates*.

Note: we could also alternatively take a basis with *majority* gates.

Main Results

The following are established in **(Anderson, D. 2014)**:

Theorem

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

Theorem

A class of graphs is accepted by a P*-uniform, polynomial-size, symmetric family of threshold circuits if, and only if, it is definable in* FPC.

Some Consequences

We get a natural and purely circuit-based characterisation of FPC definability.

Inexpressibility results for FP and FPC yield lower bound results against natural circuit classes.

- There is no polynomial-size family of symmetric Boolean circuits deciding if an n vertex graph has an even number of edges.
- Polynomial-size families of uniform symmetric *threshold circuits* are more powerful than Boolean circuits.
- Invariant circuits *cannot* be translated into equivalent symmetric threshold circuits, with only polynomial blow-up.

Symmetric Circuits for non-Boolean Queries

Instead of circuits computing *Boolean* (i.e. $0/1$) queries, we can consider circuits C that compute an m-ary relation on an input graph.

The output gate is not unique. Instead, we have an *injective function* $\Omega : [n]^m \to C$.
The range of Ω forms the *output gates*.

The requirement that $\pi \in S_n$ extends to an automorphism $\hat{\pi}$ of C includes the condition:

$$\hat{\pi}(\Omega(x)) = \Omega(\pi(x))$$

Automorphisms of Symmetric Circuits

For a symmetric circuit C_n we can assume *w.l.o.g.* that the automorphism group is the symmetric group S_n acting in the natural way.

That is:

- Each $\pi \in S_n$ gives rise to a *non-trivial* automorphism of C_n (otherwise C_n would compute a constant function).
- There are no *non-trivial* automorphisms of C_n that fix all the inputs (otherwise there is redundancy in C_n that can be eliminated).

By abuse of notation, we use $\pi \in S_n$ both for permutations of $[n]$ and automorphisms of C_n.

Stabilizers

For a gate g in C_n, $\mathrm{Stab}(g)$ denotes the *stabilizer group of* g, i.e. the *subgroup* of S_n consisting:

$$\mathrm{Stab}(g) = \{\pi \in S_n \mid \pi(g) = g\}.$$

The *orbit* of g is the set of gates $\{h \mid \pi(g) = h \text{ for some } \pi \in S_n\}$

By the *orbit-stabilizer* theorem, there is one gate in the orbit of g for each *co-set* of $\mathrm{Stab}(g)$ in S_n.
Thus the size of the *orbit* of g in C_n is $[S_n : \mathrm{Stab}(g)] = \frac{n!}{|\mathrm{Stab}(g)|}$.

So, an upper bound on $\mathrm{Stab}(g)$ gives us a lower bound on the orbit of g.

Conversely, knowing that the orbit of g is at most polynomial in n tells us that $\mathrm{Stab}(g)$ is *big*.

Supports

For a group $G \subseteq S_n$, we say that a set $X \subseteq [n]$ is a *support* of G if

For every $\pi \in S_n$, *if* $\pi(x) = x$ *for all* $x \in X$, *then* $\pi \in G$.

In other words, G contains all permutations of $[n] \setminus X$.
So, if $|X| = k$, $[S_n : G]$ is at most $\frac{n!}{(n-k)!} \leq n^k$.

Groups with small support are *big*.

The converse is clearly false since $[S_n : A_n] = 2$, but A_n has no support of size less than $n - 1$.

Note: For the family of circuits $(C_n)_{n \in \omega}$ obtained from an FPC formula there is a constant k such that all gates in each C_n have a support of size at most k.

Support Theorem

In *polynomial size* symmetric circuits, all gates have (stabilizer groups with) *small* support:

Theorem

For any polynomial p, there is a k such that for all sufficiently large n, if C is a symmetric circuit on $[n]$ of size at most $p(n)$, then every gate in C has a support of size at most k.

The general form of the support theorem in **(Anderson, D. 2014)** gives bounds on the size of supports in *sub-exponential* circuits.

Proof sketch – Supporting Equivalence Relations

Say that a permutation $\pi \in S_n$ respects an *equivalence relation* $\sim$ on $[n]$ if

$$\pi(x) \sim x \text{ for all } x \in [n].$$

Say that an equivalence relation $\sim$ on $[n]$ *supports* a group $G \subseteq S_n$ if every permutation that respects $\sim$ is in G.

We can show that every group $G \subseteq S_n$ has a *unique, coarsest* equivalence relation $\sim_G$ that supports it.

Supporting Equivalence Relations

Lemma: There is a *coarsest* equivalence relation that supports G.

Proof sketch: For two equivalence relations $\sim_1$ and $\sim_2$, let $\mathcal{E}(\sim_1, \sim_2)$ denote the finest partition that is coarser than $\sim_1$ and $\sim_2$.
Then, any permutation that fixes each equivalence class $\mathcal{E}(\sim_1, \sim_2)$ can be expressed as a composition of permutations fixing all classes of $\sim_1$ and $\sim_2$ respectively.

Essentially, every permutation in G can be expressed as a composition of permutations that respect $\sim_G$ and those that permute the equivalence classes of $\sim_G$.
Call such permutations $\sim_G$*-permutations*

Proof Sketch – Counting Equivalence Classes

If $[S_n : G] < p(n)$, then there is a constant c so that the number of equivalence classes of $\sim_G$ is either $< c$ or $> n - c$.

This is a computation of an upper bound on the number $\sim_G$-permutations when the number of $\sim_G$-equivalence classes is in the range $[c, n - c]$.

Say that $\sim_G$ is *small* if it has at most c parts and *big* otherwise.

A_n is an example of a group with *small index* where $\sim_{A_n}$ is *big*.

Proof Sketch – Largest Equivalence Class

If $[G : S_n] < p(n)$, then there is a constant c' such that if $\sim_G$ is small, then the largest equivalence class has size at least $n - c'$.

This is again proved by showing that if $\sim_G$ has fewer than c equivalence classes and all of them are smaller than $n - c'$, then there are too few $\sim_G$-permutations.

Proof Sketch – Small Supports

Claim: For a gate g in C_n, $\sim_{\mathrm{Stab}(g)}$ is small.

Suppose that g is a minimal gate (in the *DAG*-order of the circuit) with $\sim_{\mathrm{Stab}(g)}$ large.

We can show that this implies that g has a large number of immediate predecessors which (*by assumption*) have small supporting equivalence relations.

Using bounds from the previous claims, we can find a large enough subset of these, and *independently* combine automorphisms that move them.

This is used to show that $\mathrm{Orb}(g)$ must be big.

Support Theorem

In *polynomial size* symmetric circuits, all gates have (stabilizer groups with) *small* support:

Theorem

For any $1 > \epsilon \geq \frac{2}{3}$, *let* C *be a symmetric* s*-gate circuit over* $[n]$ *with* $n \geq 2^{\frac{56}{\epsilon^2}}$, *and* $s \leq 2^{n^{1-\epsilon}}$. *Then every gate* g *of* C *has a support of size at most* $\frac{33}{\epsilon}\frac{\log s}{\log n}$.

We write $\mathrm{sp}(g)$ for the small support of g given by this theorem and note that it can be computed in polynomial time from a symmetric circuit C.

Translating Symmetric Circuits to Formulas

Given a polynomial-time function $n \mapsto C_n$ that generates symmetric circuits:

1. There are formulas of FP interpreted on $([n], <)$ that define the structure C_n.
2. We can also compute in polynomial time (and therefore in FP on $([n], <)$) $\mathrm{sp}(g)$ for each gate g.
3. For an input structure $\mathbb{A}$ and an assignment $\gamma : [n] \to \mathbb{A}$ of the inputs of C_n to elements of $\mathbb{A}$, whether g is made true depends only on $\gamma(\mathrm{sp}(g))$.
4. We define, by induction on the structure of C_n, the set of tuples $\Gamma(g) \subseteq \mathbb{A}^{\mathrm{sp}(g)}$ that represent assignments γ making g true.
5. This inductive definition can be turned into a formula (of FP for a Boolean circuit, of FPC for one with threshold gates.)

Upper and Lower Bounds

The class of properties decided by *symmetric*, *polynomial size*, *threshold* circuits is FPC—a proper subset of P.
This has interesting *upper* and *lower* bounds which makes it an interesting object of study.

Upper Bounds	*Lower Bounds*
CVP	SAT
2-Colourability	3-Colourability
2-SAT	3-SAT
Perfect Matching	Hamiltonian Cycle
Linear Programming	XOR-SAT
Isomorphism on planar graphs	Isomorphism on bounded-degree graphs

FP with Rank Operators

FPrk is fixed-point logic with an operator for *matrix rank* over finite fields.

(D., Grohe, Holm, Laubner, 2009)

We have, as with FPC, terms of *element sort* and *numeric sort*.

We interpret $\eta(x, y)$—a term of numeric sort—in $G = (V, E)$ as defining a matrix with rows and columns indexed by elements of V with entries $\eta[a, b]$.

$\mathrm{rk}_{x,y}\eta$ is a term denoting the number that is the rank of the matrix defined by $\eta(x, y)$.

To be precise, we have, for each finite field $\mathbb{F}_q$ (q prime), an operator rk^q which defines the rank of the matrix with entries $\eta[a, b](\mathrm{mod}\, q)$.

Choiceless Polynomial Time

Choiceless Polynomial Time with counting ($\tilde{\mathrm{C}}\mathrm{PT(Card)}$) is a class of computational problems defined by **(Blass, Gurevich and Shelah 1999)**.

It is based on a *machine model (Gurevich Abstract State Machines)* that works directly on a graph or relational structure (rather than on a string representation).

The machine can access the collection of hereditarily finite sets with the *vertices of the graph* as atoms, and can perform counting operations.

$\tilde{\mathrm{C}}\mathrm{PT(Card)}$ is the polynomial time and space restriction of the machines.

Beyond FPC

FPrk can express the *CFI property* and solvability of systems of linear equations on finite fields. **(D., Grohe, Holm, Laubner, 2009)**

$\tilde{\mathrm{C}}$PT(Card) can express the *CFI property* (but requires sets of unbounded rank). **(D.,Richerby, Rossman, 2008)**

The relationship between the two (and their relationship to P) remains open.

Big Picture

Logic	*Circuits*
FP on structures with a disjoint number sort $([n], <)$.	Poly-size *symmetric* Boolean circuits.
Additional predicates on number sort.	Non-uniformity (of function $n \mapsto C_n$).
Connections between element sort and number sort (FPC and FPrk).	Additional gates (*counting* and *rank*).
$\tilde{\mathrm{C}}$PT(Card).	Breaking symmetry (how?).