Fixed-Point Logic with Counting.

Anuj Dawar

University of Cambridge Computer Laboratory

ALC 2014, Mumbai, 5 January 2015
Finite Model Theory

In *finite model theory* we are concerned with studying the *definability* of classes of finite relational structures by means of formulas of logic.

$$\text{Mod}(\varphi)$$ denotes the *finite* models of $\varphi$

Specifically, the study of *descriptive complexity* relates definability to the *computational complexity* of the decision problem:

*Given $A$ decide if $A \models \varphi$*

Many of the examples in this talk concern finite structures over a vocabulary with *one binary relation*, which we think of as *finite graphs*:

$$G = (V, E)$$
$\forall x \forall y \forall z (\neg E(x, y) \lor \neg E(x, z) \lor \neg E(y, z))$

defines the graphs that do not contain a triangle.

For any first-order sentence $\varphi$, $\text{Mod}(\varphi)$ is trivially decidable (in polynomial time and logarithmic space).

There are computationally easy classes that are not defined by any first-order sentence.

- The class of graphs with an even number of vertices.
- The class of graphs that are connected.
For every finite structure $\mathbb{A}$, there is a first-order sentence $\varphi_{\mathbb{A}}$ defining the structures isomorphic to $\mathbb{A}$.

Every *isomorphism-invariant* class $S$ of finite structures is definable by a *first-order theory* $T$:

$$T = \{ \neg \varphi_{\mathbb{A}} \mid \mathbb{A} \not\in S \}$$

The interesting definability questions are obtained by considering:

- *extensions* of first-order logic; or equivalently
- *restricted* first-order theories.
FPC is the extension of first-order logic with a mechanism for *iteration* and a mechanism for *counting*.

It was proposed by Immerman as a possible logic for *capturing* $P$:

It was proved *Cai, Fürer, Immerman 1992* that there are polynomial-time graph properties that are *not* expressible in FPC.
A number of other results about the limitations of FPC followed.

Still, FPC forms a *natural* and *powerful* fragment of $P$.
In this talk, we look at three recent, *positive* results on the expressive power of FPC.
Fixed-Point Logic

The logic **FP** is formed by closing first-order logic under the rule:

*If $\varphi$ is a formula, positive in the relational variable $R$, then so is $[\text{lfp}_{R,x} \varphi](t)$."

The formula is read as:

*the tuple $t$ is in the least fixed point of the operator that maps $R$ to $\varphi(R, x)$.*
Connectivity

The formula

$$\forall u \forall v [\text{lfp}_{T,xy}(x = y \lor \exists z (E(x, z) \land T(z, y)))](u, v)$$

is satisfied in a graph $(V, E)$ if, and only if, it is connected.

The expressive power of FP properly extends that of first-order logic.

On structures which come equipped with a linear order FP expresses exactly the classes that are decidable in \textit{polynomial time}.  

\textbf{(Immerman; Vardi)}

In the \textit{absence} of order, there is no formula of FP that defines the graphs with an even number of vertices.
Fixed-Point Logic with Counting

FPC is a logic formulated to add the ability to count to FP.

If $\varphi(x)$ is a formula with free variable $x$, then $\#_x \varphi$ is a term denoting the number of elements satisfying $\varphi$.

Formulae of FPC:

- all atomic formulae as in FP;
- $\tau_1 < \tau_2; \tau_1 = \tau_2$ where $\tau_i$ is a term of numeric sort;
- $\exists x \varphi; \exists \nu \varphi$; where $\nu$ is a variable ranging over numbers up to the size of the domain;
- $\text{lfp}_{x,x,\nu} \varphi(t)$; and
- $\varphi \land \psi; \neg \varphi$. 
$C^k$ is the logic obtained from first-order logic by allowing:

- **counting quantifiers**: $\exists^i x \varphi$; and
- only the variables $x_1, \ldots, x_k$.

Every formula of $C^k$ is equivalent to a formula of first-order logic, albeit one with more variables.

For every sentence $\varphi$ of FPC, there is a $k$ such that $\varphi$ is equivalent to a theory of $C^k$.

Indeed, for any fixed $n$, there is a formula of $C^k$ equivalent to $\varphi$ on structures with at most $n$ elements.
There are polynomial-time decidable properties of graphs that are not definable in FPC. \((\text{Cai, F"ur"er, Immerman, 1992})\)

Other inexpressibility results for FPC follow, either as a consequence of \((\text{Cai, F"ur"er, Immerman, 1992})\) or by similar methods:

- *Hamiltonian Cycle* and *Satisfiability* are not definable in FPC.
- *3-Colourability* is not definable in FPC. \((\text{D. 1998})\)
- Solvability of systems of linear equations (over any fixed finite Abelian group) is not definable in FPC \((\text{Atserias, Bulatov, D. 2009})\)

All of these are shown, in fact, to be not axiomatizable in \(C^k\), for any \(k\).
If we restrict the class of structures we consider, FPC may be powerful enough to express all polynomial-time decidable properties.

1. FPC captures $P$ on trees. \textit{(Immerman and Lander 1990)}.
2. FPC captures $P$ on any class of graphs of \textit{bounded treewidth}. \textit{(Grohe and Mariño 1999)}.
3. FPC captures $P$ on the class of \textit{planar graphs}. \textit{(Grohe 1998)}.
4. FPC captures $P$ on any \textit{proper minor-closed class of graphs}. \textit{(Grohe 2010)}.

In each case, the proof proceeds by showing that for any $G$ in the class, a \textit{canonical, ordered} representatonon of $G$ can be interpreted in $G$ using FPC.
Graph Minors
We say that a graph $H$ is a minor of graph $G$ (written $H \preceq G$) if $H$ can be obtained from $G$ by repeated applications of the operations:

- delete an edge;
- delete a vertex (and all incident edges); and
- contract an edge
Graph Minors

Alternatively, $H = (U, F)$ is a minor of $G = (V, E)$, if there is a set $V' \subseteq V$ and a surjective map $M : V' \rightarrow U$ such that

- for each $u \in U$, $M^{-1}(u)$ is a connected subgraph of $G$; and
- for each edge $(u, v) \in F$, there is an edge in $E$ between some $x \in M^{-1}(u)$ and some $y \in M^{-1}(v)$.
Recall: $G$ is planar if, and only if, $K_5 \not\preceq G$ and $K_{3,3} \not\preceq G$.

**Theorem (Robertson-Seymour)**
In any infinite collection $\{G_i \mid i \in \omega\}$ of graphs, there are $i, j$ with $G_i \preceq G_j$.

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

A consequence is that any $C$ closed under minors is decidable in polynomial-time.

The proof relies on Robertson and Seymour’s *structure theorem*:

*A graph $G$ that excludes a minor $K_k$ admits a tree-decomposition in which each bag is almost embeddable in a surface of genus $k'$*
Grohe’s proof is a version of the structure theorem with *definable decompositions*.

A *treelike decomposition* of a graph $G$ is a *directed acyclic graph* $D$, with a *bag* $\beta(d) \subseteq V(G)$ of vertices associated with each node of $D$ and satisfying certain *connectedness* and *consistency* conditions.

A treelike decomposition of $G$ can be obtained (for instance) from a *tree decomposition* by closing it under the *automorphisms* of $G$—starting at leaves and working upwards.
Treelike Decomposition of a 5-cycle

The picture shows a treelike decomposition of a 5-cycle $C_5$. The grey nodes form a tree decomposition.

picture credit: M. Grohe: JACM, 59(5), 27.
Definable Treelike Decompositions

Grohe shows that there is an FPC-definable decomposition of planar graphs into their 3-connected components.
This is lifted into a decomposition of graphs embeddable in an arbitrary surface.
More heavy lifting is required to obtain a definable treelike decomposition of the class of graphs excluding a $K_k$-minor into components that can be almost embedded in a surface.
This is used to show that for each $k$, there is a $k'$ such that on graphs excluding $K_k$ as a minor, $C^{k'}$ defines isomorphism.
As a consequence, every class of graphs closed under taking minors is definable in FPC.
Linear Programming
We can represent an instance of a linear programming feasibility problem as a *relational structure* over a suitable vocabulary.

We have a set $C$ of *constraints* over a set $V$ of *variables*. Each $c \in C$ consists of $a_c \in \mathbb{Q}^V$ and $b_c \in \mathbb{Q}$. The numbers are encoded in *binary* over an ordered set of *bit positions*.

**Feasibility Problem:** Given a linear programming instance, determine if there is an $x \in \mathbb{Q}^V$ such that:

$$a_c^T x \leq b_c \quad \text{for all} \quad c \in C$$

In *Anderson, D., Holm (2013)* we show that this, and the corresponding *optimization problem* are expressible in FPC.
Ellipsoid Method

The set of constraints determines a polytope.
Ellipsoid Method

Start at the origin and calculate an *ellipsoid* enclosing it.
Ellipsoid Method

If the centre is not in the polytope, choose a constraint it *violates.*
Ellipsoid Method

Calculate a new *centre*. 
And a new ellipsoid around the centre of at most half the volume.
We can encode all the calculations involved in FPC. This relies on expressing algebraic manipulations of unordered matrices.

What is not obvious is how to choose the violated constraint on which to project.

However, the ellipsoid method works as long as we can find, at each step, some separating hyperplane.
Ellipsoid Method in FPC
Ellipsoid Method in FPC

We can encode all the calculations involved in FPC. This relies on expressing algebraic manipulations of *unordered* matrices.

What is not obvious is how to *choose* the violated constraint on which to project.

However, the ellipsoid method works as long as we can find, at each step, some *separating hyperplane*.

So, we can take:

\[
(\sum_{c \in S} a_c)^T x \leq \sum_{c \in S} b_c
\]

where \( S \) is the *set* of all violated constraints.
More generally, the ellipsoid method can be used, even when the constraint matrix is not given explicitly, as long as we can always determine a separating hyperplane.

In particular, the polytope represented may have exponentially many facets.

Anderson, D., Holm (2013) shows that as long as the separation oracle can be defined in FPC, the corresponding optimization problem can be solved in FPC.
Matching

In a graph $G = (V, E)$ a matching $M \subset E$ is a set of edges such that each vertex is incident on at most one edge in $M$.

The problem of finding a maximum matching in $G$ can be represented by a linear program with exponentially many constraints.

We show that a separation oracle for this polytope is definable by an FPC formula interpreted in the graph $G$.

As a consequence, there is an FPC formula defining the size of the maximum matching in $G$.

Blass, Gurevich and Shelah (2001) had shown that matching on bipartite graphs is definable in FPC and conjectured that this was not true on general graphs.
Symmetric Circuits
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: \( \land, \lor, \neg \),
- 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 \( P/\text{poly} \).

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 \( \{\land, \lor, \neg\} \) or a richer basis with threshold or majority gates.
A property of *graphs* (or other relational structures) in $P$ is recognised by a family of Boolean circuits $(C_n)_{n \in \omega}$ where:

- inputs to $C_n$ are $n^2$ potential edges, each taking value 0 or 1;
- 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 DLogTime).

$C_n$ is *invariant* if, for every input graph, the output is unchanged under a permutation of the inputs induced by a permutation of $[n]$. 
Say $C_n$ is symmetric if any permutation of $[n]$ applied to its inputs can be extended to an automorphism of $C_n$.

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

- Any formula $\varphi$ of FP translates into a uniform family of polynomial-size symmetric Boolean circuits.
- Any formula of FPC translates into a uniform family of polynomial-size symmetric threshold (or majority) circuits.
We established the following in Anderson, D. (2014):

**Theorem**
A class of graphs is accepted by a $P$-uniform 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 symmetric family of threshold circuits *if, and only if*, it is definable in $FPC$. 
Main Technical Tools

For a symmetric circuit $C_n$ we can assume \textit{w.l.o.g.} that the automorphism group is the symmetric group $S_n$ acting in the natural way.

For a gate $g$ in $C_n$, $\text{Stab}(g)$ denotes the \textit{stabilizer group of $g$}, i.e.,

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

Say a set $X \subseteq [n]$ \textit{supports} $g$ if

$$\text{Stab}^\bullet(X) \subseteq \text{Stab}(g),$$

where $\text{Stab}^\bullet(X) := \{ \pi \in S_n \mid \pi(x) = x \text{ for all } x \in X \}$ is the \textit{pointwise stabilizer} of $X$.

\textbf{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

Our main technical theorem shows that in *sub-exponential size* symmetric circuits, all gates have *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 $\frac{33}{\epsilon} \frac{\log s}{\log n}$.

**Corollary**
Polynomial-size symmetric circuits have constant support.
Some Consequences

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.

There is no translation of invariant circuit into equivalent symmetric threshold circuits, with only polynomial blow-up.

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

Inexpressibility results for FPC are also lower bound results against a natural circuit class.
Conclusions

The intuition behind the conjecture that FPC captures P was that algorithmic techniques that are obviously polynomial-time are all expressible in the logic.

The Cai-Fürer-Immerman construction and related results show that one important polynomial-time algorithmic technique—Gaussian elimination—is not captured by the logic.

Recent results show that some very non-trivial and non-obvious polynomial-time problems can be expressed in FPC:

- Linear Programming
- Arbitrary minor-closed classes
- Maximum Matching

And, there is a natural circuit complexity class corresponding to FPC.