Aspects of Descriptive Complexity

Anuj Dawar

Department of Computer Science and Technology, University of Cambridge

Contextuality as a Resource in Quantum Computation
Oxford, 5 July 2018

Descriptive Complexity

Descriptive Complexity is an attempt to study the complexity of problems and classify them, not on the basis of how difficult it is to *compute* solutions, but on the basis of how difficult it is to *describe* the problem.

This gives an alternative way to study complexity, independent of particular machine models.

Based on *definability in logic*.

Graph Properties

As an example, consider the following three decision problems on *graphs*.

1. Given a graph $G = (V, E)$ does it contain a *triangle*?
2. Given a directed graph $G = (V, E)$ and two of its vertices $a, b \in V$, does G contain a *path* from a to b?
3. Given a graph $G = (V, E)$ is it *3-colourable*? That is,

    is there a function $\chi : V \to \{1, 2, 3\}$ *so that whenever* $(u, v) \in E$, $\chi(u) \neq \chi(v)$.

Graph Properties

1. Checking if G contains a triangle can be solved in *polynomial time* and *logarithmic space*.

2. Checking if G contains a path from a to b can be done in *polynomial time*.
Can it be done in *logarithmic space*?

Unlikely. It is NL-complete.

3. Checking if G is 3-colourable can be done in *exponential time* and *polynomial space*.
Can it be done in *polynomial time*?

Unlikely. It is NP-complete.

Logical Definability

In what kind of formal language can these decision problems be *specified* or *defined*?

The graph $G = (V, E)$ contains a triangle.

$$\exists x, y, z \in V (x \neq y \wedge y \neq z \wedge x \neq z \wedge E(x, y) \wedge E(x, z) \wedge E(y, z))$$

The other two properties are *provably* not definable with only first-order quantification over vertices.

First-Order Logic

Consider *first-order predicate logic*.

A collection of variables $x, y, \ldots$, *and formulas:*

$$x = y \mid E(x, y) \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x\varphi \mid \forall x\varphi$$

Any property of graphs that is expressible in *first-order logic* is in L.

The problem of deciding whether $G \models \varphi$ for a first-order φ is in time $O(ln^m)$ and $O(m \log n)$ space.

where, l is the *length* of φ and n the *order* of G and m is the nesting depth of quantifiers in φ.

Second-Order Quantifiers

3-Colourability and *Reachability* can be defined with quantification over *sets of vertices*.

$$\begin{aligned}
&\exists R \subseteq V\ \exists B \subseteq V\ \exists G \subseteq V \\
&\qquad \forall x (Rx \lor Bx \lor Gx) \land \\
&\qquad \forall x (\neg(Rx \land Bx) \land \neg(Bx \land Gx) \land \neg(Rx \land Gx)) \land \\
&\qquad \forall x \forall y (Exy \to (\neg(Rx \land Ry) \land \\
&\qquad\qquad \neg(Bx \land By) \land \\
&\qquad\qquad \neg(Gx \land Gy)))
\end{aligned}$$

$$\forall S \subseteq V (a \in S \land \forall x \forall y ((x \in S \land E(x,y)) \to y \in S) \to b \in S)$$

Existential Second-Order Logic

Second-order logic is obtained by adding to the defining rules of first-order logic two further clauses:

> *atomic formulae* – $X(t_1, \ldots, t_a)$, *where* X *is a second-order variable*
>
> *second-order quantifiers* – $\exists X \varphi$, $\forall X \varphi$

Existential Second-Order Logic (ESO) consists of formulas of the form

$$\exists X_1 \cdots \exists X_k \varphi$$

where φ is *first-order*

Fagin's Theorem

Theorem (Fagin 1974)
A class of graphs is definable by a formula of *existential second-order logic* if, and only if, it is decidable by a *nondeterminisitic machine* running in polynomial time.

$$\mathsf{ESO} = \mathsf{NP}$$

One direction is easy: Given G and $\exists X_1 \ldots \exists X_k \varphi$.

> *a nondeterministic machine can guess an interpretation for* $X_1, \ldots, X_k$ *and then verify* φ.

The other direction translates a non-deterministic Turing machine M into a sentence φ_M which asserts, in a structure $\mathbb{A}$ the existence of relations coding an acccepting run of M on an input coding $\mathbb{A}$.

A Logic for P?

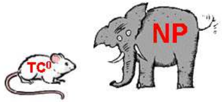
Is there a logic, intermediate between first and second-order logic that expresses exactly graph properties in P?

This is an open question, first posed by **(Chandra and Harel, 1982)** and has been the motor of significant research in descriptive complexity.

Does P admit a syntactic characterisation?

Can the class P be "built up from below" by finitely many operations?

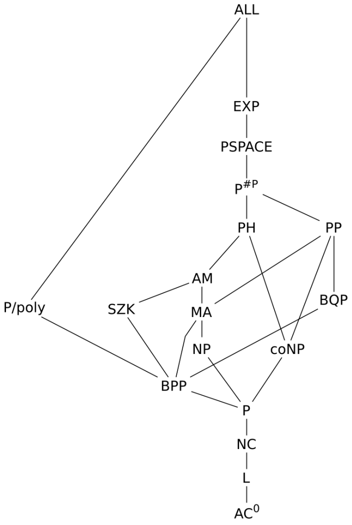
Zoo of Complexity Classes



Scott Aaronson and others have compiled an online *zoo* of complexity classes, which has 535 entries and counting.

P—languages decidable by a deterministic Turing machine running in polynomial time.
NP—languages decidable by a nondeterministic Turing machine running in polynomial time.
coNP—languages whose complements are in NP.
BQP—quantum polynomial time.
AC^0—languages decided by a uniform family of constant-depth, polynomial-size Boolean circuits.

Some Inclusions among Classes



Note: P/poly and All are uncountable classes that necessarily contain undecidable languages.

AC^0 is properly included in L.

L is known to be properly included in PSpace but none of the individual inclusions in between is known to be proper.

Enumerating Complexity Classes

Given a complexity class $\mathcal{C}$, can we enumerate its members?

Fix an enumeration of *Turing machines* and write L_i for the language accepted by machine M_i.

Can we decide the set $\{i \mid L_i \in \mathcal{C}\}$?

No—Rice's theorem.

Say that $\mathcal{C}$ is *weakly indexed* by a set $I \subseteq \mathbb{N}$ if:

- $i \in I \quad \Rightarrow \quad L_i \in \mathcal{C}$
- $L \in \mathcal{C} \quad \Rightarrow \quad \exists i \in I \; L = L_i$

Can we computably enumerate a weak index set for $\mathcal{C}$?

Syntactic Classes

Usually we want something more of a syntactic characterisation of $\mathcal{C}$ then just a computably enumerable weak index set. We want the machines M_i to *"witness"* that L_i is in $\mathcal{C}$.

For instance, fix an enumeration of pairs (M, p) where M is a deterministic Turing machine and p is a polynomial.

Let I be the range of the function that takes (M, p) to the code of the Turing machine that simulates M for $p(n)$ steps on inputs of length n.

I is an *effective syntax* for P.

Syntactic Classes

NP can similarly be indexed by pairs (M, p) where M is a *nondeterministic* Turing machine and p is a polynomial.

What about NP ∩ coNP?
An index set is obtained by taking

$$(M, M', p) \quad \text{such that} \quad L(M) = L(M')$$

↑ *undecidable condition*

So we say P and NP are *syntactic* classes, while NP ∩ coNP is a *semantic* class.

Graph Problems

Consider decision problems where the input is a graph: *Connectedness*, *3-Colouring*, *Hamiltonicity*.

We can encode graphs as strings over $\{0, 1\}$ by, for instance, enumerating the *adjacency matrix*.

There are up to $n!$ distinct encodings of a given n vertex graph G.

For $x, y \in \{0, 1\}^*$ write $x \sim y$ to indicate that they are encodings of the same graph.

And, for a language L, we say it is $\sim$-invariant if

$$x \in L \text{ and } x \sim y \quad \Rightarrow \quad y \in L.$$

Invariant Complexity Classes

Let inv-NP and inv-P be the classes of all languages that are in NP and P respectively and are $\sim$-invariant.

inv-NP is indexed by the set of pairs (M, p) where M is a nondeterminsitic Turing machine, p is a polynomial *and* the language accepted by M when clocked by p is $\sim$-invariant.

The invariance condition is undecidable.

Fagin's theorem tells us that, nonetheless, inv-NP has an *effective syntax*. It is indexed by machines obtained from existential second-order sentences.

Whether inv-P has an effective syntax is *the* open question.

BQP

BQP is the complexity class of problems solvable by *quantum* polynomial time algorithms.

Formally, a language L is in BQP if there is a *quantum* Turing machine M, running in polynomial time such that

- if $x \in L$ then M accepts x with probability $> \frac{2}{3}$; and
- if $x \notin L$ then M accepts x with probability $< \frac{1}{3}$.

Say that a machine M is *well-formed* for BQP if, for every string x, it is the case that the probability of M accepting is either $< \frac{1}{3}$ or $> \frac{2}{3}$.

Index set for BQP

We can obtain an index set for BQP by enumerating all pairs

$$(M, p)$$

where M is a quantum Turing machine and p is a polynomial such that, M clocked by p is *well-formed* for BQP.

The well-formedness condition is *undecidable*.

BQP is a *semantic class*, at least by definition.

inv-BQP

Is there an *effective syntax* for inv-BQP?

There are *two* undecidable conditions in the definition of the class: *well-formedness* and $\sim$*-invariance*.

The second condition might not be an obstacle if there is a polynomial-time quantum algorithm that can produce a $\sim$-canonical representation of a graph *with high probability*.

The first *is* a serious obstacle, and getting a logic for BQP would require a radically different characterization that was *extensionally* the same as BQP.

Fixed-Point Logic with Counting

FPC—*Fixed-Point with Counting* is an extension of first-order logic with a *recursion operator* and a mechanism for *counting*.

FPC was first proposed by **Immerman** as a possible logic for P.

It was shown by **(Cai, Fürer, Immerman 1992)** that there are graph properties in P not in FPC.

FPC captures a *large* and *natural* fragment of P and is worthy of study in its own right.

> *Many powerful polynomial-time algorithms can be expressed in* FPC *and at the same time, we can prove unconditional lower bounds on it.*

Counting Quantifiers

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.

We write $\mathbb{A} \equiv^k \mathbb{B}$ to denote that no sentence of C^k distinguishes $\mathbb{A}$ from $\mathbb{B}$.

This *family of equivalence relations* (also known as k-dimensional Weisfeiler-Leman equivalences) has many different natural formulations in *combinatorics, algebra, logic* and *linear optimization* among others.

FPC and C^k

For every sentence φ of FPC, there is a k such that if $\mathbb{A} \equiv^k \mathbb{B}$, then

$$\mathbb{A} \models \varphi \quad \text{if, and only if,} \quad \mathbb{B} \models \varphi.$$

Essentially, FPC can be understood as those problems decided by polynomial-time algorithms that are inviariant under $\equiv^k$ for some k.

3SAT, *XOR-Sat*, *Hamiltonicity*, *3-Colourability* are not $\equiv^k$-invariant for any constant k.
The same is true of solving systems of equations over *any* finite field.

Counting Width

For any class of structures $\mathcal{C}$, we define its *counting width* $\nu_{\mathcal{C}} : \mathbb{N} \to \mathbb{N}$ so that

> $\nu_{\mathcal{C}}(n)$ *is the least* k *such that* $\mathcal{C}$ *restricted to structures with at most* n *elements is closed under* $\equiv^k$*.*

Every class in FPC has counting width bounded by a *constant*.

3SAT, *XOR-Sat*, *Hamiltonicity*, *3-Colourability* all have counting width $\Omega(n)$.

Constraint Satisfaction Problems

Fix $\mathbb{A}$ and $\mathbb{B}$, two relational structures in the same *relational vocabulary* τ.

A *homomorphism* from $\mathbb{A}$ to $\mathbb{B}$ is a map $h : \mathbb{A} \to \mathbb{B}$ so that for any tuple $\mathbf{a}$ and any relation R,

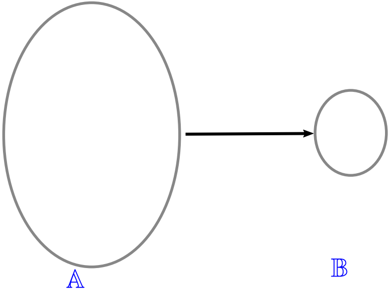
$$R^{\mathbb{A}}(\mathbf{a}) \quad \Rightarrow \quad R^{\mathbb{B}}(h(\mathbf{a})).$$

For a structure $\mathbb{B}$: $\mathrm{CSP}(\mathbb{B}) = \{\mathbb{A} \mid \mathbb{A} \longrightarrow \mathbb{B}\}$

3SAT, *XOR-Sat*, *3-Colourability* all can be naturally formulated as CSP.

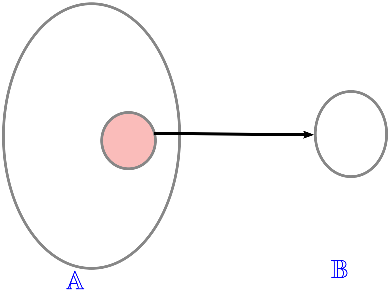
Local Consistency Methods

Deciding, given $\mathbb{A}$ and $\mathbb{B}$, whether $\mathbb{A} \to \mathbb{B}$ is NP-complete.
In some cases, we can test instead for *local consistency*.



Local Consistency Methods

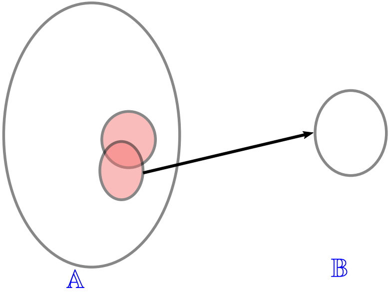
Deciding, given $\mathbb{A}$ and $\mathbb{B}$, whether $\mathbb{A} \to \mathbb{B}$ is NP-complete.
In some cases, we can test instead for *local consistency*



For a k-element subset.

Local Consistency Methods

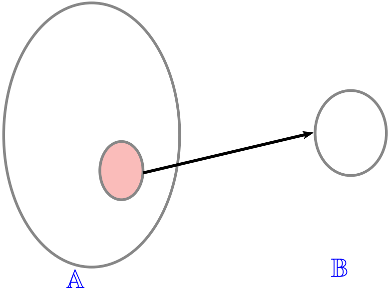
Deciding, given $\mathbb{A}$ and $\mathbb{B}$, whether $\mathbb{A} \to \mathbb{B}$ is NP-complete.
In some cases, we can test instead for *local consistency*



The *extension property*.

Local Consistency Methods

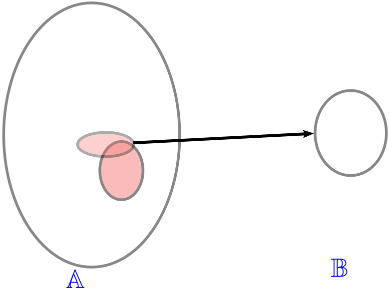
Deciding, given $\mathbb{A}$ and $\mathbb{B}$, whether $\mathbb{A} \to \mathbb{B}$ is NP-complete.
In some cases, we can test instead for *local consistency*



The *extension property*.

Local Consistency Methods

Deciding, given $\mathbb{A}$ and $\mathbb{B}$, whether $\mathbb{A} \to \mathbb{B}$ is NP-complete.
In some cases, we can test instead for *local consistency*



The *extension property*.

The Pebbling Co-Monad

In **(Abramsky, D., Wang, 17)** we give a construction of a *graded co-monad* $\mathbb{T}_k$.

This gives rise to a *co-Kleisli category* in which the morphisms are exactly the k-local consistency tests.

$$\mathbb{T}_k\mathbb{A} \longrightarrow \mathbb{B} \quad \text{if, and only if,} \quad \mathbb{A} \xrightarrow{k} \mathbb{B}.$$

Isomorphism in this category is exactly $\equiv^k$.

That is, $\mathbb{A} \equiv^k \mathbb{B}$ if, and only if, $\mathbb{T}_k\mathbb{A} \cong \mathbb{T}_k\mathbb{B}$.

Lower Bounds from Counting Width

A *CSP* has counting width either $O(1)$ or $\Omega(n)$.
The former if, and only if, it is definable in *Datalog*.

(Atserias, Bulatov, D. '09); (Barto-Kozik '14)

For a CSP of *unbounded counting width*, the corresponding *maximization* problem is intractable. **(Thapper, Živný '16); (D.,Wang '15)**

3SAT, *XOR-Sat*, *Vertex Cover* cannot be *approximated* by any class of bounded counting width. **(Atserias, D. '18)**

An $\Omega(n)$ lower bound on the counting width of a class implies *exponential* lower bounds on the size of *symmetric circuits* and *symmetric linear programs* deciding it. **(Anderson, D. 2017)**

(Atserias, D., Ochremiak '19)

Symmetric Linear Programs

Fix $X = \{x_{ij} \mid i,j \in [n]\}$ for a fixed n.
Consider a class $\mathcal{C}$ of graphs.
We identify a graph on n vertices with a function $G : X \to \{0,1\}$.

We say that a polytope $Q \subseteq \mathbb{R}^X \times \mathbb{R}^Y$ *recognizes* $\mathcal{C}$ if its projection on $\mathbb{R}^X$ includes $\mathcal{C}|_n$ and excludes its complement.

Say $Q \subseteq \mathbb{R}^X \times \mathbb{R}^Y$ is *symmetric* if for every $\pi \in S_V$, there is a $\sigma \in S_Y$ such that

$$Q^{(\pi,\sigma)} = Q$$

Here, we extend the action of π to $V \times V$, and hence to $\mathbb{R}^X$.

Symmetric Linear Programs

Theorem (Atserias, D., Ochremiak '19)
If a family of symmetric polytopes of size $s = O(2^{n^{1-\epsilon}}), \epsilon > 0$ recognizes $\mathcal{C}$, then $\mathcal{C}$ has *counting width* at most $\frac{\log s}{\log n}$.

In particular, classes of counting width $\Omega(n)$ are not recognized by any *subexponential* size symmetric linear programs.