Descriptive and Computational Complexity

Anuj Dawar
University of Cambridge

Leicester, 16 February 2007

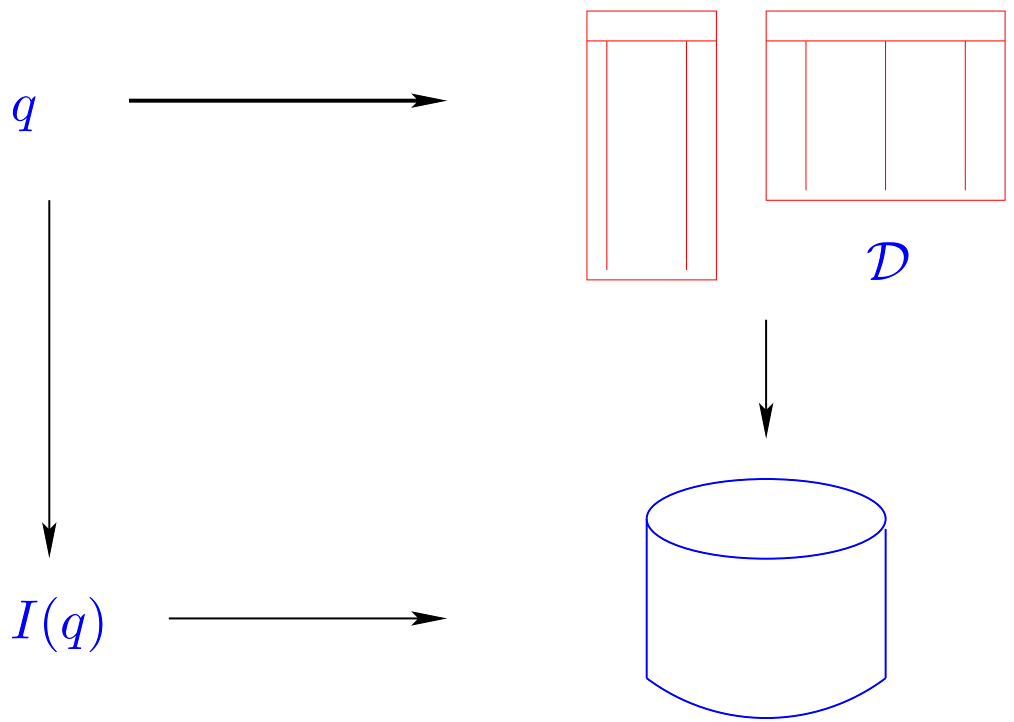
Complexity and Database Theory

Descriptive Complexity Theory arises from questions in computational complexity and in database theory.

In 1974, **Fagin** showed that the collection of problems definable in *existential second-order logic* is exactly the problems in **NP**.

In 1980, **Chandra and Harel** asked whether there a database query language in which one can express exactly the *feasible, generic* queries.

Generic Queries



A query q is *generic* if the answer to q depends only on the abstract view $\mathcal{D}$

q is *feasible* if its implementation $I(q)$ runs in time polynomial in the size of $\mathcal{D}$

Descriptive vs. Computational Complexity

Computational Complexity:

is concerned with measuring space, time or other resources on a machine model of computation.

usually defines complexity of a language - i.e. a set of strings

Descriptive Complexity:

defines the complexity of classes of structures - *e.g.* a collection of graphs, or relations.

concerned with the complexity of describing the collection in a suitable language.

Relational Databases

$$Cinema = \{Movies[3], Location[3], Guide[3]\}$$

Movies	Title	Director	Actor
	Volver	Almodovar	Cruz
	Volver	Almodovar	Maura
	Casino Royale	Campbell	Craig
	Casino Royale	Campbell	Green
	...		
	Rocky	Stallone	Stallone

Guide	Title	Cinema	Time
	Rocky	Vue	12:00
	Volver	Picturehouse	19:00
	...		
	Casino Royale	Cineworld	19:00
	Rocky	Cineworld	22:00

Location	Cinema	Address	Tel
	Picturehouse	Cambridge	504444
	Vue	Leicester	240240
	Cineworld	Cambridge	560225

Relational Algebra

In relational algebra, queries are built up from

Base relations: R

Singleton constant relations: $\{\langle a \rangle\}$

using

select: $\sigma_{j=a}(q)$ or $\sigma_{j=k}(q)$

project: $\pi_{j_1,\ldots,j_k}(q)$

join: $q_1 \bowtie q_2$

union: $q_1 \cup q_2$

difference: $q_1 - q_2$

Relational Calculus

Codd in 1972 introduced the relational calculus (based on first-order logic) and equivalent to the relational algebra.

Conjunctive Queries:

$$q(x, y) \leftarrow \textit{Movies}(z_1, \text{“Almodovar”}, z_2), \textit{Guide}(x, z_1, z_3), \textit{Location}(x, y, z_4)$$

expresses the query

$$\{x, y \mid \exists z_1, \ldots, z_4\ \textit{Movies}(z_1, \text{“Almodovar”}, z_2) \wedge \textit{Guide}(x, z_1, z_3) \wedge \textit{Location}(x, y, z_4)\}$$

Disjunction is expressed by *multiple rules*.

First-Order Logic

Adding negation and universal quantification gives us the full-power of relational algebra, or equivalently, *first-order logic*.

Note: closed-world assumption.

From now on, we speak of finite relational structures:

$$\mathbb{A} = (A, R_1, \ldots, R_m)$$

where A is a finite *domain* and each R_i is a relation on A.

And queries are given by formulas of predicate logic:

atomic formulas – $R(t_1, \ldots, t_m)$, $t_1 = t_2$

Boolean operations – $\varphi \wedge \psi$, $\varphi \vee \psi$, $\neg\varphi$

first-order quantifiers – $\exists x \varphi$, $\forall x \varphi$

Complexity of First-Order Logic

A query expressed by a first-order formula φ can be evaluated in time polynomial in the size of the structure $\mathbb{A}$.

If $\psi(x_1, \ldots, x_k)$ is a sub-formula of φ,

there are at most n^k tuples satisfying this formula.

where n is the number of elements in A.

In fact, it can be shown that the query can be computed in *logarithmic space*.

Limitations of First-Order Logic

There are *polynomial-time computable* and *generic* queries that are not computable in first-order logic.

Evennness:

Is the number of elements in A even?

Transitive Closure:

In a structure (A, R) with a binary relation R, give the set of pairs (x, y) such that there is an R-path from x to y.

Second-Order Quantifiers

Existential Second-Order Quantification:

$$\exists P_1 \ldots \exists P_m \varphi$$

A structure $\mathbb{A}$ satisfies $\exists P \varphi$ if there is a relation R on the universe of $\mathbb{A}$ such that $(\mathbb{A}, R)$ satisfies φ.

ESO – existential second order logic

$$\text{ESO} \subseteq \text{NP}$$

An existential second order quantifier represents a polynomial amount of non-determinism.

Examples

Evennness

This formula is true in a structure if, and only if, the size of the domain is even.

$$
\begin{aligned}
\exists B \exists S \quad & \forall x \exists y B(x,y) \wedge \forall x \forall y \forall z B(x,y) \wedge B(x,z) \rightarrow y = z \\
& \forall x \forall y \forall z B(x,z) \wedge B(y,z) \rightarrow x = y \\
& \forall x \forall y S(x) \wedge B(x,y) \rightarrow \neg S(y) \\
& \forall x \forall y \neg S(x) \wedge B(x,y) \rightarrow S(y)
\end{aligned}
$$

Examples

Transitive Closure

This formula is true of a pair of elements a, b in a structure if, and only if, there is an R-path from a to b.

$$\begin{array}{ll} \exists P & \forall x \forall y\, P(x,y) \rightarrow R(x,y) \\ & \exists x P(a,x) \land \exists x P(x,b) \land \neg\exists x P(x,a) \land \neg\exists x P(b,x) \\ & \forall x(x \neq a \land \exists y(P(x,y) \rightarrow \forall z(P(x,z) \rightarrow y = z))) \\ & \forall x(x \neq b \land \exists y(P(y,x) \rightarrow \forall z(P(z,x) \rightarrow y = z))) \end{array}$$

Examples

3-Colourability

The following formula is true in a graph (V, E) if, and only if, it is 3-colourable.

$$
\begin{aligned}
\exists R \exists B \exists G \quad & \forall x (Rx \vee Bx \vee Gx) \wedge \\
& \forall x (\quad \neg(Rx \wedge Bx) \wedge \neg(Bx \wedge Gx) \wedge \neg(Rx \wedge Gx)) \wedge \\
& \forall x \forall y (Exy \rightarrow (\quad \neg(Rx \wedge Ry) \wedge \\
& \qquad\qquad\qquad\qquad \neg(Bx \wedge By) \wedge \\
& \qquad\qquad\qquad\qquad \neg(Gx \wedge Gy)))
\end{aligned}
$$

Note, this is an NP-complete problem and so unlikely to be computable in polynomial-time.

Fagin's Theorem

Fagin proved that *every* problem that is in the complexity class NP is definable by a formula of ESO.

NP can be defined as the class of problems decidable by guessing a polynomial number of bits, and then running a polynomial-time verification algorithm

Fagin's theorem says that the verification phase can always be replaced by a first-order formula.

Chandra and Harel's question asks whether we can similarly characterise the class P.

Recursion

We are looking for logical formalisms intermediate in expressive power between first-order and second-order logic.

One idea, considered by Chandra and Harel, is to add a recursion mechanism to first-order logic.

Example:

$$\begin{aligned} T(x,y) &\leftarrow R(x,y) \\ T(x,y) &\leftarrow R(x,z), T(z,y). \end{aligned}$$

This recursively defines a relation T that is the transitive closure of the relation R.

LFP

More generally, we allow any first-order formula on the right-hand side of the rule:

$S(\mathbf{x}) \leftarrow \varphi(S)$ where φ is positive in the symbol S.

This rule has a *least* solution for S, and this solution can be constructed in time polynomial in the size of the structure $\mathbb{A}$.

If we allow S to occur inside a negation symbol on the right, the rule may not have a solution (*viz.* $S(x) \leftarrow \neg S(x)$).

LFP is the logic that is obtained by adding a recursion operator to first-order logic.

It can still not express *Evenness*.

Counting

LFP + C is a logic formulated to add the ability to count to LFP.

A second *sort* of variables: $\nu_1, \nu_2, \ldots$ which range over *numbers* in the range

$$0, \ldots, |A|$$

If $\varphi(x)$ is a formula with free variable x, then $\nu = \#x\varphi$ denotes that ν is the number of elements of A that satisfy the formula φ.

We also have the order $\nu_1 < \nu_2$, which allows us (using recursion) to define arithmetic operations.

Evenness

There are an even number of elements satisfying $\varphi(x)$.

$$\exists\nu_1\exists\nu_2(\nu_1 = [\#x\varphi] \land (\nu_2 + \nu_2 = \nu_1))$$

Cai-Fürer-Immerman

Cai, Fürer and Immerman (1992) showed that $\mathrm{LFP} + \mathrm{C}$ is not powerful enough to express all properties in *P*.

The proof involved a contrived construction of a class of graphs on which the graph isomorphism problems is solvable in polynomial time but not definable in $\mathrm{LFP} + \mathrm{C}$.

They conjectured that adding some "group-theoretic operators" may be a solution.

Group-theoretic Operators

We (Atserias, Bulatov, D., 2007) have recently exhibited natural feasibly computable problems that are not definable in LFP $+$ C.

- Solving linear equations over a finite field; or more simply
- Solving additive equations over a finite Abelian group.

These suggest natural operators that could be added to LFP $+$ C to obtain a logic that can still only express feasibly computable properties.

Linear Equations

Consider systems of equations (with three variables per equation), over the integers $\mathrm{mod}\ 2$.

$$a_1 + a_2 + a_3 = 0$$
$$a_2 + a_3 + a_4 = 1$$

has the solution $a_1 = a_2 = a_3 = 0$, $a_4 = 1$.

This can be coded as a structure with domain $\{a_1, \ldots, a_n\}$ and ternary relations R_0 and R_1, with:

$(a_i, a_j, a_k) \in R_m$ iff $a_i + a_j + a_k = m$ is an equation in the system

There is no formula of LFP $+$ C that defines the *solvable* systems of equations.

Challenges

Prove that the extension of LFP + C with an operator for determining the *rank of a matrix* still does not express all properties in P.

Other operators have also been defined in the literature (*e.g.* symmetric choice). It remains an open problem to show that these don't capture all of P.

It's possible that P cannot be "generated from below" by a finite collection of operators. To prove this would also separate P from NP.