Expressiveness and Complexity of Database Query Languages

Todd Millstein
CSE590DB
January 26, 1999

Fundamental Assumptions on Queries

- *well-typed*
  - the schema of the result of a query is fixed
- *computable*
  there is a TM M for query q such that
  - if $q(\mathbf{I})$ is undefined, then TM M does not terminate on input $enc(\mathbf{I})$
  - if $q(\mathbf{I})$ is defined, then TM M halts on input $enc(\mathbf{I})$ with output $enc(q(\mathbf{I}))$ on the tape

Fundamental Assumptions (continued)

- *generic*
  - data independence principle
  - "names of constants don't matter":
    let ρ be a permutation on **dom**

$$\begin{tikzcd}
\mathbf{I} \arrow[r, "q"] \arrow[d, "\rho"] & q(\mathbf{I}) \arrow[d, "\rho"] \\
\rho(\mathbf{I}) \arrow[r, "q"] & \rho(q(\mathbf{I})) = q(\rho(\mathbf{I}))
\end{tikzcd}$$

Complexity of Queries

- *data complexity*: the complexity of evaluating $q(\mathbf{I})$ for a *fixed query* q, as a function of $|\mathbf{I}|$
- *query (expression) complexity*: the complexity of evaluating $q(\mathbf{I})$, for a *fixed instance* $\mathbf{I}$, as a function of $|q|$

Complexity of What?

- the *recognition problem* for a query:
  - Given an instance **I** and a tuple u, determine if $u \in q(\mathbf{I})$
- the *size of the result* of the query

Example: given a binary relation G

$cross(G) = \pi_1(G) \times \pi_2(G)$
$path(G) = \{<x,y> \mid y \text{ is reachable from } x \text{ in } G\}$
$id(G) = G$

Languages and Complexity

For each complexity class C, QC denotes the class of queries whose recognition problem is in C.

A query language L *expresses* QC if L expresses precisely the queries in QC.
- difficulty: TMs are not generic

A query language L is *complete w.r.t.* C if
- each query in L is in QC
- there is a query in L for which the associated recognition problem is complete for C

First-Order Queries

$e ::= \text{atom} \mid t_1 = t_2 \mid t_1 \neq t_2 \mid e_1 \wedge e_2 \mid e_1 \vee e_2 \mid \sim e \mid \forall x.e \mid \exists x.e$

equivalent to
- nonrecursive datalog with negation
- relational algebra
  - select
  - project
  - cartesian product
  - union
  - difference

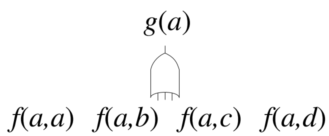
Complexity of First-Order Queries

Theorem: $FO \subseteq QL$
Proof [sketch]: Let φ be a query in FO in prenex form and u be the tuple we're looking for.
Base case: φ has no quantifiers
Just need to scan the input instance, looking for each literal in φ, with the constants in u substituted for the corresponding free variables in φ.
Inductive case: $\varphi = \exists x.\psi$ ($\forall x.\psi$ is handled similarly)
Try all possible values for x that appear in **I**. Accept iff some value for x makes ψ true.

Complexity of First-Order Queries (cont.)

AC_0 is the class of problems computable with a constant-depth circuit using polynomially many logic gates (with unbounded fan-in).

Theorem: $FO \subseteq AC_0$
Proof [basic idea]:
Can implement select, project, etc. with constant-depth circuits.



Ehrenfeucht-Fraissé Game

given instances **I** and **J** over the same schema, repeat r times:
- *spoiler* picks a constant in **I** or **J**
- *duplicator* picks a constant in the other instance

- let a_i (b_i) be the ith constant picked in **I** (**J**).
- **I**/$\{a_1,\ldots,a_r\}$ consists of all facts in **I** using only these constants.
- duplicator *wins the round* iff the mapping $a_i \to b_i$ is an isomorphism of **I**/$\{a_1,\ldots,a_r\}$ and **J**/$\{b_1,\ldots,b_r\}$
- duplicator *wins the game of length r* associated with **I** and **J**, denoted **I** $\equiv_r$ **J**, if he has a winning strategy.

E-F Game Example

I = {E(c,d), E(d,c), E(d,e)}
J = {E(f,g), E(g,h)}

duplicator wins the game of length 1:
duplicator's strategy: pick a random constant
for any a_1 and b_1, **I**/$\{a_1\}$ = **J**/$\{b_1\}$ = { }

duplicator does not win the game of length 2:
spoiler's strategy: choose c, then d (or vice versa)

E-F Games and Negative Results for FO

Theorem: Let **I** and **J** be instances over the same schema. Let φ be a query in *FO* with no free variables and quantifier depth *r*. If **I** $\equiv_r$ **J**, then either **I** and **J** both satisfy φ or neither does.

Therefore, to show that some property is not expressible in *FO,* do the following:
for each *r*, exhibit an **I** and **J** such that
- **I** $\equiv_r$ **J**
- **I** satisfies the property, but **J** does not

Example: Graph Connectivity

Theorem: Graph connectivity is not expressible in *FO*
Proof [idea]: for each r, there is a sufficiently large n s.t. an *FO* query of quantifier depth r cannot distinguish between one $2n$-cycle and two n-cycles.

example: $r = 3$, $n = 5$

Corollary: Transitive closure is not expressible in *FO*

0-1 Laws

Let q be a boolean query (with no free variables) over some schema $\mathbf{R}$.

$$\text{Define } \mu_n(q) = \frac{\{\mathbf{I} \mid q(\mathbf{I}) = \mathit{true} \text{ and } \mathbf{I} \text{ has } n \text{ constants}\}}{\{\mathbf{I} \mid \mathbf{I} \text{ has } n \text{ constants}\}}$$

We say q is *almost surely true* (*false*) if $\lim_{n\to\infty}\mu_n(q)$ exists and equals 1 (0).

If every boolean query (with no free variables) in a language L is almost surely true or almost surely false, *L has a 0-1 law.*

0-1 Laws (continued)

Theorem: Datalog (and hence *FO*) has a 0-1 law.
Proof: See the book.

Let U be a unary relation. Define the parity query $\mathit{even}(U) = \mathit{true}$ if $|U|$ is even, and *false* otherwise

Corollary: *even* is not expressible in datalog (and hence *FO*).