Advanced Database Systems (ADBS), University of Edinburgh, 2024/25

Conjunctive Queries: Evaluation and Static Analysis

(Chapter 14 and 15 of DBT)

[DBT] Database Theory, https://github.com/pdm-book/community

A Core Relational Query Language

Conjunctive Queries (CQ)

= {σ,π,⋈}-fragment of relational algebra

= relational calculus using only ∃ and ∧

= simple SELECT-FROM-WHERE SQL queries (only AND and equality in the WHERE clause)

Syntax of Conjunctive Queries

$$Q(\mathbf{x}) \ := \ \exists \mathbf{y}\, (R_1(\mathbf{v_1}) \ \wedge \cdots \wedge \ R_m(\mathbf{v_m}))$$

- $R_1,\dots,R_m$ are relation names
- $\mathbf{x}, \mathbf{y}, \mathbf{v_1},\dots,\mathbf{v_m}$ are tuples of variables
- each variable mentioned in $\mathbf{v_i}$ appears either in $\mathbf{x}$ or $\mathbf{y}$
- the variables in $\mathbf{x}$ are free called distinguished or output variables

It is very convenient to see conjunctive queries as rule-based queries of the form

$$Q(\mathbf{x}) \ :- \ \underbrace{R_1(\mathbf{v_1}),\dots,R_m(\mathbf{v_m})}$$

this is called the body of Q that can be seen as a set of atoms

Pattern Matching Problem

List the airlines that fly directly from London to Glasgow

{
Flight(VIE,LHR,BA),
Flight(LHR,EDI,BA),
Flight(LGW,GLA,U2),
Flight(LCA,VIE,OS),
Airport(VIE,Vienna),
Airport(LHR,London),
Airport(LGW,London),
Airport(LCA,Larnaca),
Airport(GLA,Glasgow),
Airport(EDI,Edinburgh)
}

Q(z) :- Airport(x,London), Airport(y,Glasgow), Flight(x,y,z)

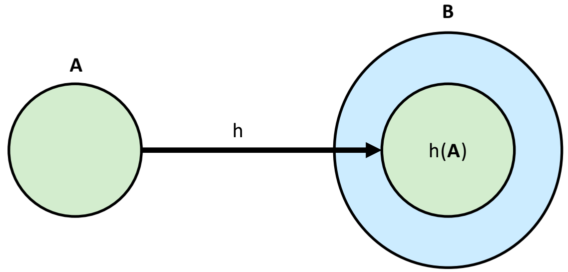
Pattern Matching Problem

List the airlines that fly directly from London to Glasgow

{
Flight(VIE,LHR,BA),
Flight(LHR,EDI,BA),
Flight(LGW,GLA,U2),
Flight(LCA,VIE,OS),
Airport(VIE,Vienna),
Airport(LHR,London),
Airport(LGW,London),
Airport(LCA,Larnaca),
Airport(GLA,Glasgow),
Airport(EDI,Edinburgh)
}

Q(z) :- Airport(x,London), Airport(y,Glasgow), Flight(x,y,z)

Homomorphism



h : terms(**A**) → terms(**B**) that is the identity on constants

Semantics of Conjunctive Queries

- A match of a conjunctive query $Q(x_1,\ldots,x_k)$:- body in a database D is a homomorphism h from the set of atoms body to the set of atoms D

- The answer to $Q(x_1,\ldots,x_k)$:- body over D is the set of k-tuples
  $Q(D) := \{(h(x_1),\ldots,h(x_k)) \mid h \text{ is a match of } Q \text{ in } D\}$

- The answer consists of the witnesses for the distinguished variables of Q

Pattern Matching Problem

List the airlines that fly directly from London to Glasgow

{
Flight(VIE,LHR,BA),
Flight(LHR,EDI,BA),
Flight(LGW,GLA,U2),
Flight(LCA,VIE,OS),
Airport(VIE,Vienna),
Airport(LHR,London),
Airport(LGW,London),
Airport(LCA,Larnaca),
Airport(GLA,Glasgow),
Airport(EDI,Edinburgh)
}

Q(z) :- Airport(x,London), Airport(y,Glasgow), Flight(x,y,z)

Pattern Matching Problem

List the airlines that fly directly from London to Glasgow

{
Flight(VIE,LHR,BA),
Flight(LHR,EDI,BA),
Flight(LGW,GLA,U2),
Flight(LCA,VIE,OS),
Airport(VIE,Vienna),
Airport(LHR,London),
Airport(LGW,London),
Airport(LCA,Larnaca),
Airport(GLA,Glasgow),
Airport(EDI,Edinburgh)
}

{x ↦ LGW, y ↦ GLA, z ↦ U2,
London ↦ London, Glasgow ↦ Glasgow}

Q(z) :- Airport(x,London), Airport(y,Glasgow), Flight(x,y,z)

Query Evaluation

- Understand the complexity of evaluating a conjunctive query over a database
- What to measure? Queries may have a large output and it would be misleading to count the output as "complexity"
- We therefore consider the following decision problem for **CQ**

CQ-Evaluation
Input: a database D, a CQ $Q(x_1,\dots,x_k)$:- body, and a tuple $(a_1,\dots,a_k)$ of values
Question: $(a_1,\dots,a_k) \in Q(D)$?

combined complexity

Data Complexity of Query Evaluation

- Measures the complexity in terms of the size of the database - the query is fixed
- Meaningful in practice since the database is usually much bigger than the query
- We consider the following decision problem for a fixed CQ $Q(x_1,\dots,x_k)$:- body

Q-Evaluation
Input: a database D, and a tuple $(a_1,\dots,a_k)$ of values
Question: $(a_1,\dots,a_k) \in Q(D)$?

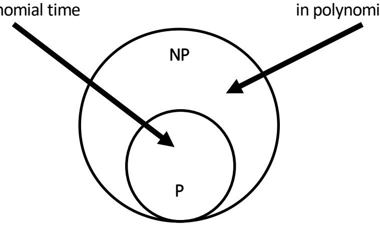
Complexity of Query Evaluation

Theorem: CQ-Evaluation is NP-complete and in PTIME in data complexity

Few Words about NP

a solution can be **found** in polynomial time

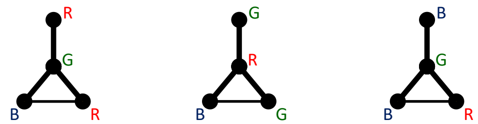
a solution can be **verified** in polynomial time



- $P \subseteq NP$, but it is open whether $P \subset NP$ or $P = NP$ - one of the most important questions in mathematics and theoretical computer science
- One of the Millennium Prize Problems selected by the Clay Mathematics Institute in 2000 (1 million USD prize for the first correct solution)

3-Colorability Problem

> 3COL
> **Input:** an undirected graph **G** = (V,E)
> **Question:** is there a function $c : V \rightarrow \{R,G,B\}$ such that $(v,u) \in E \Rightarrow c(v) \neq c(u)$?



3-Colorability Problem

> 3COL
> **Input:** an undirected graph **G** = (V,E)
> **Question:** is there a function $c : V \rightarrow \{R,G,B\}$ such that $(v,u) \in E \Rightarrow c(v) \neq c(u)$?

Theorem: 3COL is NP-complete

Proof:

- **Guess** a function $c : V \rightarrow \{R,G,B\}$ in polynomial time, and **verify** that $(v,u) \in E \Rightarrow c(v) \neq c(u)$ in polynomial time
- 3COL is one of the most difficult problems in the complexity class NP (it is unlikely to be solvable in polynomial time unless P = NP)

Complexity of Query Evaluation

Theorem: CQ-Evaluation is NP-complete and in PTIME in data complexity

Proof:

(NP-membership) Guess-and-verify:

- Consider a database D, a CQ $Q(x_1,\dots,x_k)$:- body, and a tuple $(a_1,\dots,a_k)$ of values
- Guess a substitution h : terms(body) $\rightarrow$ terms(D) that is the identity on constants
- Verify that h is a match of Q in D, i.e., h(body) $\subseteq$ D and $(h(x_1),\dots,h(x_k)) = (a_1,\dots,a_k)$

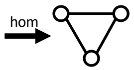
(NP-hardness) Reduction from 3COL

NP-hardness

(NP-hardness) Reduction from 3COL

> 3COL
>
> **Input:** an undirected graph **G** = (V,E)
>
> **Question:** is there a function c : V → {R,G,B} such that (v,u) ∈ E ⇒ c(v) ≠ c(u)?

Lemma: G is 3-colorable iff **G** can be mapped to $\mathbf{K_3}$, i.e., **G**



therefore, **G** is 3-colorable iff there is a match of Q_G in D = {E(x,y),E(y,z),E(z,x)}

the Boolean CQ that represents **G**

Complexity of Query Evaluation

Theorem: CQ-Evaluation is NP-complete and in PTIME in data complexity

Proof:

(NP-membership) Guess-and-verify:

- Consider a database D, a CQ $Q(x_1,\ldots,x_k)$:- body, and a tuple $(a_1,\ldots,a_k)$ of values
- Guess a substitution h : terms(body) → terms(D) that is the identity on constants
- Verify that h is a match of Q in D, i.e., h(body) ⊆ D and $(h(x_1),\ldots,h(x_k)) = (a_1,\ldots,a_k)$

(NP-hardness) Reduction from 3-colorability

(in PTIME) For every substitution h : terms(body) → terms(D) that is the identity on constants, check if h(body) ⊆ D and $(h(x_1),\ldots,h(x_k)) = (a_1,\ldots,a_k)$

Static Analysis

> **CQ**-Satisfiability
>
> **Input:** a conjunctive query Q
>
> **Question:** is there a database D such that Q(D) is non-empty?

- If the answer is no, then the input query Q makes no sense
- **CQ**-Evaluation becomes trivial - the answer is always NO!

Static Analysis

> **CQ**-Equivalence
>
> **Input:** two conjunctive queries Q_1 and Q_2
>
> **Question:** $Q_1 \equiv Q_2$? or $Q_1(D) = Q_2(D)$ for every database D?

- Replace a query Q_1 with a query Q_2 that is easier to evaluate
- But, we have to be sure that $Q_1(D) = Q_2(D)$ for every database D

Static Analysis

CQ-Containment
Input: two conjunctive queries Q_1 and Q_2
Question: $Q_1 \subseteq Q_2$? or $Q_1(D) \subseteq Q_2(D)$ for every database D?

- Equivalence boils down to two containment checks
- Clearly, $Q_1(D) = Q_2(D)$ iff $Q_1(D) \subseteq Q_2(D)$ and $Q_2(D) \subseteq Q_1(D)$

Complexity of Static Analysis

CQ-Satisfiability
Input: a conjunctive query Q
Question: is there a database D such that Q(D) is non-empty?

CQ-Equivalence
Input: two conjunctive queries Q_1 and Q_2
Question: $Q_1 \equiv Q_2$? or $Q_1(D) = Q_2(D)$ for every database D?

CQ-Containment
Input: two conjunctive queries Q_1 and Q_2
Question: $Q_1 \subseteq Q_2$? or $Q_1(D) \subseteq Q_2(D)$ for every database D?

Canonical Database

- Convert a conjunctive query Q into a database D[Q] - the canonical database of Q
- Given a conjunctive query of the form Q(**x**) :- body, D[Q] is obtained from body by replacing each variable x with a new value c(x) = $\underline{x}$
- E.g., given Q(x,y) :- R(x,y), P(y,z,w), R(z,x), then D[Q] = {R($\underline{x}$,$\underline{y}$), P($\underline{y}$,$\underline{z}$,$\underline{w}$), R($\underline{z}$,$\underline{x}$)}
- **Note:** The mapping c : {variables in body} → {new values} is a bijection, where c(body) = D[Q] and c^{-1}(D[Q]) = body

Satisfiability of CQs

CQ-Satisfiability
Input: a conjunctive query Q
Question: is there a database D such that Q(D) is non-empty?

Theorem: A conjunctive query Q is always satisfiable

Proof: Due to its canonical database - Q(D[Q]) is trivially non-empty

Equivalence and Containment of CQs

CQ-Equivalence
Input: two conjunctive queries Q_1 and Q_2
Question: $Q_1 \equiv Q_2$? or $Q_1(D) = Q_2(D)$ for every database D?

CQ-Containment
Input: two conjunctive queries Q_1 and Q_2
Question: $Q_1 \subseteq Q_2$? or $Q_1(D) \subseteq Q_2(D)$ for every database D?

$Q_1 \equiv Q_2$ iff $Q_1 \subseteq Q_2$ and $Q_2 \subseteq Q_1$

$Q_1 \subseteq Q_2$ iff $Q_1 \equiv (Q_1 \wedge Q_2)$

...thus, we can safely focus on **CQ**-Containment

Homomorphism Theorem

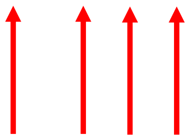
A query homomorphism from $Q_1(x_1,\dots,x_k)$:- $body_1$ to $Q_2(y_1,\dots,y_k)$:- $body_2$ is a substitution h : terms($body_1$) → terms($body_2$) such that:

1. h is a homomorphism from $body_1$ to $body_2$
2. $(h(x_1),\dots,h(x_k)) = (y_1,\dots,y_k)$

Homomorphism Theorem: Let Q_1 and Q_2 be conjunctive queries. It holds that:
$Q_1 \subseteq Q_2$ iff there exists a query homomorphism from Q_2 to Q_1

Homomorphism Theorem: Example

$Q_1(x,y)$:- $R(x,z), S(z,z), R(z,y)$



$h = \{a \mapsto x, b \mapsto y, c \mapsto z, d \mapsto z\}$

$Q_2(a,b)$:- $R(a,c), S(c,d), R(d,b)$

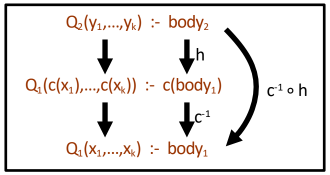
- h is a query homomorphism from Q_2 to Q_1 $\Rightarrow$ $Q_1 \subseteq Q_2$
- But, there is no homomorphism from Q_1 to Q_2 $\Rightarrow$ $Q_1 \subset Q_2$

Homomorphism Theorem: Proof

Assume that $Q_1(x_1,\dots,x_k)$:- $body_1$ and $Q_2(y_1,\dots,y_k)$:- $body_2$

(⇒) $Q_1 \subseteq Q_2$ $\Rightarrow$ there exists a query homomorphism from Q_2 to Q_1

- Clearly, $(c(x_1),\dots,c(x_k)) \in Q_1(D[Q_1])$ - recall that $D[Q_1] = c(body_1)$
- Since $Q_1 \subseteq Q_2$, we conclude that $(c(x_1),\dots,c(x_k)) \in Q_2(D[Q_1])$
- Therefore, there exists a homomorphism h such that $h(body_2) \subseteq D[Q_1] = c(body_1)$ and $h((y_1,\dots,y_k)) = (c(x_1),\dots,c(x_k))$
- By construction, $c^{-1}(c(body_1)) = body_1$ and $c^{-1}((c(x_1),\dots,c(x_k))) = (x_1,\dots,x_k)$
- Therefore, $c^{-1} \circ h$ is a query homomorphism from Q_2 to Q_1

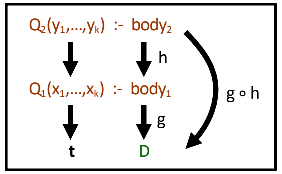


Homomorphism Theorem: Proof

Assume that $Q_1(x_1,\ldots,x_k)$:- $body_1$ and $Q_2(y_1,\ldots,y_k)$:- $body_2$

($\Leftarrow$) $Q_1 \subseteq Q_2$ $\Leftarrow$ there exists a query homomorphism from Q_2 to Q_1

- Consider a database D, and a tuple **t** such that **t** $\in Q_1(D)$
- We need to show that **t** $\in Q_2(D)$
- Clearly, there exists a homomorphism g such that $g(body_1) \subseteq D$ and $g((x_1,\ldots,x_k)) =$ **t**
- By hypothesis, there exists a query homomorphism h from Q_2 to Q_1
- Therefore, $g(h(body_2)) \subseteq D$ and $g(h((y_1,\ldots,y_k))) =$ **t**, which implies that **t** $\in Q_2(D)$



Existence of a Query Homomorphism

Theorem: Let Q_1 and Q_2 be conjunctive queries. The problem of deciding whether there exists a query homomorphism from Q_2 to Q_1 is NP-complete

Proof:

(NP-membership) Guess a substitution and verify that is a query homomorphism

(NP-hardness) Easy reduction from **CQ**-Evaluation

By applying the homomorphism theorem we get that:

Corollary: **CQ**-Equivalence and **CQ**-Containment are NP-complete