Finite Model Theory
Lecture 4: Conjunctive Queries

Spring 2025

Where We Are

- Thanks for submitting HW1.
- HW2 still to be released: it covers this lecture and the next one.
- Demystifying decidability of Σ. Two problems:
  - Membership: given φ, is $\varphi \in \Sigma$?
  - Implication: given φ, does $\Sigma \models \varphi$ hold?
  - If membership is r.e. then implication is r.e. (Gödel's completeness).
  - Example of Σ that is not r.e.: $\mathtt{Th}(\mathbb{N}, +, *)$ (Gödel's incompleteness).

Today: Model Checking, and Conjunctive Queries.

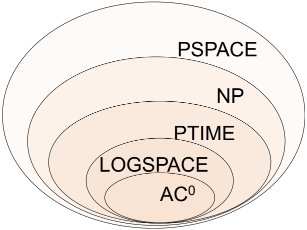
Model Checking

Model Checking

Model checking: given $\boldsymbol{A}$, φ, check whether $\boldsymbol{A} \models \varphi$.

When $\boldsymbol{A}$ is a database and φ a query, then this is called *query evaluation*. This is the bread-and-butter of database engines.

What is the complexity of checking $\boldsymbol{A} \models \varphi$?



Three Ways to Define Complexity

$$\boxed{\boldsymbol{A} \models \varphi}$$

- Data Complexity. Fix the sentence φ, complexity is $f(|\boldsymbol{A}|)$.
- Expression Complexity. Fix the structure $\boldsymbol{A}$, complexity is $f(|\varphi|)$.
- Combined Complexity, $f(|\varphi|, |\boldsymbol{A}|)$.

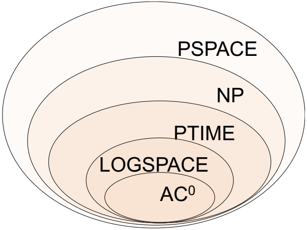
Three Ways to Define Complexity

$$\boxed{\boldsymbol{A} \models \varphi}$$

Theorem

The Data Complexity of FO is in AC^0 (actually in uniform AC^0).
The Expression- and Combined Complexity of FO is PSPACE complete.

AC^0 is at the bottom of the hierarchy
We start by proving something simpler:
The Data Complexity of FO is in PTIME.



Data Complexity of FO is in PTIME: Proof

How do we evaluate this? $\boxed{\varphi = \exists x\,(A(x) \wedge \forall y(B(y) \Rightarrow C(x,y)))}$

Assume the domain is $[N]$.

Data Complexity of FO is in PTIME: Proof

How do we evaluate this? $\boxed{\varphi = \exists x\,(A(x) \wedge \forall y(B(y) \Rightarrow C(x,y)))}$

Assume the domain is $[N]$.

```
fx = false;
for x = 1,N do:
    if A(x) then:
            fy = true
            for y = 1,N do:
                if not (B(y) => C(x,y))
                then: fy = false;
    if fy then: fx = true;
return fx
```

Data Complexity of FO is in PTIME: Proof

How do we evaluate this? $\boxed{\varphi = \exists x\,(A(x) \wedge \forall y(B(y) \Rightarrow C(x,y)))}$

Assume the domain is $[N]$.

```
fx = false;
for x = 1,N do:
    if A(x) then:
            fy = true
            for y = 1,N do:
                if not (B(y) => C(x,y))
                then: fy = false;
    if fy then: fx = true;
return fx
```

- Generalizes to any sentence in prenex normal form φ
- Runtime $O(N^k)$, where: $N = |\text{Dom}|$, $k = |\text{Vars}(\varphi)|$
- In PTIME (and in LOGSPACE).

Data Complexity of FO is in PTIME: Proof

How do we evaluate this? $\boxed{\varphi = \exists x\,(A(x) \wedge \forall y(B(y) \Rightarrow C(x,y)))}$

Assume the domain is $[N]$.

```
fx = false;
for x = 1,N do:
    if A(x) then:
            fy = true
            for y = 1,N do:
                if not (B(y) => C(x,y))
                then: fy = false;
    if fy then: fx = true;
return fx
```

- Generalizes to any sentence in prenex normal form φ
- Runtime $O(N^k)$, where: $N = |\text{Dom}|$, $k = |\text{Vars}(\varphi)|$
- In PTIME (and in LOGSPACE).

Many texts state that the data complexity is in LOGSPACE, or in PTIME. The correct complexity is AC^0. Let's prove it

Definition of AC^0

Definition

A problem is in AC^0 if $\forall N$, there exists a circuit of polynomial size and constant depth, consisting NOT gates and unbounded fan-in AND and OR gates, that computes the problem when the input is encoded using N bits.

Definition of AC^0

Definition

A problem is in AC^0 if $\forall N$, there exists a circuit of polynomial size and constant depth, consisting NOT gates and unbounded fan-in AND and OR gates, that computes the problem when the input is encoded using N bits.

E.g. Given an undirected graph with n nodes, check if it has a triangle.

Definition of AC^0

Definition

A problem is in AC^0 if $\forall N$, there exists a circuit of polynomial size and constant depth, consisting NOT gates and unbounded fan-in AND and OR gates, that computes the problem when the input is encoded using N bits.

E.g. Given an undirected graph with n nodes, check if it has a triangle. When $n = 4$ there are $N = 6$ possible edges, E_{ij} for $1 \leq i < j \leq 4$

Definition of AC^0

Definition

A problem is in AC^0 if $\forall N$, there exists a circuit of polynomial size and constant depth, consisting NOT gates and unbounded fan-in AND and OR gates, that computes the problem when the input is encoded using N bits.

E.g. Given an undirected graph with n nodes, check if it has a triangle. When $n = 4$ there are $N = 6$ possible edges, E_{ij} for $1 \leq i < j \leq 4$

$$(E_{12} \wedge E_{23} \wedge E_{13}) \vee (E_{12} \wedge E_{24} \wedge E_{14}) \vee (E_{23} \wedge E_{34} \wedge E_{24}) \vee (E_{34} \wedge E_{14} \wedge E_{13})$$

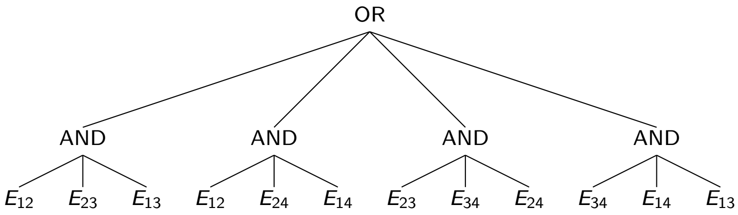
Definition of AC^0

Definition

A problem is in AC^0 if $\forall N$, there exists a circuit of polynomial size and constant depth, consisting NOT gates and unbounded fan-in AND and OR gates, that computes the problem when the input is encoded using N bits.

E.g. Given an undirected graph with n nodes, check if it has a triangle. When $n = 4$ there are $N = 6$ possible edges, E_{ij} for $1 \le i < j \le 4$

$$(E_{12} \wedge E_{23} \wedge E_{13}) \vee (E_{12} \wedge E_{24} \wedge E_{14}) \vee (E_{23} \wedge E_{34} \wedge E_{24}) \vee (E_{34} \wedge E_{14} \wedge E_{13})$$



Data Complexity FO is in AC^0: Proof

Fix φ in FO. Encode the input $\boldsymbol{A}$ using bits:

- Let $N = |\mathrm{Dom}(\boldsymbol{A})|$.

Data Complexity FO is in AC^0: Proof

Fix φ in FO. Encode the input $\boldsymbol{A}$ using bits:

- Let $N = |\text{Dom}(\boldsymbol{A})|$.
- Encode a relation of arity k using N^k bits.
  E.g. $R(X, Y)$ encoded using Boolean matrix R_{ij}, with N^2 bits.

Data Complexity FO is in AC^0: Proof

Fix φ in FO. Encode the input $\boldsymbol{A}$ using bits:

- Let $N = |\mathrm{Dom}(\boldsymbol{A})|$.
- Encode a relation of arity k using N^k bits.
  E.g. $R(X, Y)$ encoded using Boolean matrix R_{ij}, with N^2 bits.
- Expand φ into a Boolean formula $F_{\varphi,\boldsymbol{A}}$, called the lineage of φ on $\boldsymbol{A}$.

Data Complexity FO is in AC^0: Proof

Fix φ in FO. Encode the input $\boldsymbol{A}$ using bits:

- Let $N = |\mathrm{Dom}(\boldsymbol{A})|$.
- Encode a relation of arity k using N^k bits.
  E.g. $R(X, Y)$ encoded using Boolean matrix R_{ij}, with N^2 bits.
- Expand φ into a Boolean formula $F_{\varphi,\boldsymbol{A}}$, called the lineage of φ on $\boldsymbol{A}$.
- Represent $F_{\varphi,\boldsymbol{A}}$ using an AC^0 circuit.

Data Complexity FO is in AC^0: Proof

Fix φ in FO. Encode the input $\boldsymbol{A}$ using bits:

- Let $N = |\mathrm{Dom}(\boldsymbol{A})|$.
- Encode a relation of arity k using N^k bits.
  E.g. $R(X, Y)$ encoded using Boolean matrix R_{ij}, with N^2 bits.
- Expand φ into a Boolean formula $F_{\varphi,\boldsymbol{A}}$, called the lineage of φ on $\boldsymbol{A}$.
- Represent $F_{\varphi,\boldsymbol{A}}$ using an AC^0 circuit.

Example: $\boxed{\varphi = \exists x\,(A(x) \wedge \forall y(B(y) \Rightarrow C(x,y)))}$

Data Complexity FO is in AC^0: Proof

Fix φ in FO. Encode the input $\boldsymbol{A}$ using bits:

- Let $N = |\mathrm{Dom}(\boldsymbol{A})|$.
- Encode a relation of arity k using N^k bits.
  E.g. $R(X, Y)$ encoded using Boolean matrix R_{ij}, with N^2 bits.
- Expand φ into a Boolean formula $F_{\varphi,\boldsymbol{A}}$, called the lineage of φ on $\boldsymbol{A}$.
- Represent $F_{\varphi,\boldsymbol{A}}$ using an AC^0 circuit.

Example: $\boxed{\varphi = \exists x \left(A(x) \wedge \forall y (B(y) \Rightarrow C(x, y))\right)}$

Its lineage is: $\boxed{F_{\varphi,\boldsymbol{A}} = \bigvee_{i=1,N} \left(A_i \wedge \bigwedge_{j=1,N} \left(\neg B_j \vee C_{ij}\right)\right)}$

Data Complexity FO is in AC^0: Proof

Fix φ in FO. Encode the input $\boldsymbol{A}$ using bits:

- Let $N = |\mathrm{Dom}(\boldsymbol{A})|$.
- Encode a relation of arity k using N^k bits.
  E.g. $R(X, Y)$ encoded using Boolean matrix R_{ij}, with N^2 bits.
- Expand φ into a Boolean formula $F_{\varphi,\boldsymbol{A}}$, called the lineage of φ on $\boldsymbol{A}$.
- Represent $F_{\varphi,\boldsymbol{A}}$ using an AC^0 circuit.

Example: $\boxed{\varphi = \exists x\,(A(x) \wedge \forall y(B(y) \Rightarrow C(x,y)))}$

Its lineage is: $\boxed{F_{\varphi,\boldsymbol{A}} = \bigvee_{i=1,N}\left(A_i \wedge \bigwedge_{j=1,N}\left(\neg B_j \vee C_{ij}\right)\right)}$

Represented by a circuit of depth 5 and size $O(N^2)$ in class

Summary

- Data complexity: AC^0. This implies LOGSPACE, PTIME.
  - AC^0 is the class of highly parallelizable problems.
  - "SQL is embarrassingly parallel"
- Expression complexity, combined complexity: PSPACE complete
  - Membership is straightforward WHY????
  - We will prove hardness later.

Conjunctive Queries

Motivation

- FO is too rich for static analysis: Trakhtenbrot's theorem is a fundamental limit.
- For some FO fragments static analysis is possible, and they still capture the most important queries in practice.
- Conjunctive Queries (CQ) and Unions of Conjunctive Queries (UCQ).

Motivation

- FO is too rich for static analysis: Trakhtenbrot's theorem is a fundamental limit.
- For some FO fragments static analysis is possible, and they still capture the most important queries in practice.
- Conjunctive Queries (CQ) and Unions of Conjunctive Queries (UCQ).

Terminology alert	query=formula; Boolean query=sentence;	database instance=structure; query evaluation=model checking

Conjunctive Queries

Definition

A Conjunctive Query (CQ) is a formula of the form:

$$\boxed{Q(\boldsymbol{x}_0) = \exists \boldsymbol{y}\, (R_1(\boldsymbol{x}_1) \wedge R_2(\boldsymbol{x}_2) \wedge \cdots \wedge R_m(\boldsymbol{x}_m))}$$

[1] Relational Algebra: $\sigma, \Pi, \bowtie, \cup, -$.

Conjunctive Queries

Definition

A Conjunctive Query (CQ) is a formula of the form:

$$\boxed{Q(\boldsymbol{x}_0) = \exists \boldsymbol{y}\,(R_1(\boldsymbol{x}_1) \wedge R_2(\boldsymbol{x}_2) \wedge \cdots \wedge R_m(\boldsymbol{x}_m))}$$

- The free variables, $\boldsymbol{x}_0$, are called head variables.

[1] Relational Algebra: $\sigma, \Pi, \bowtie, \cup, -$.

Conjunctive Queries

Definition

A Conjunctive Query (CQ) is a formula of the form:

$$\boxed{Q(\boldsymbol{x}_0) = \exists \boldsymbol{y}\,(R_1(\boldsymbol{x}_1) \wedge R_2(\boldsymbol{x}_2) \wedge \cdots \wedge R_m(\boldsymbol{x}_m))}$$

- The free variables, $\boldsymbol{x}_0$, are called head variables.
- Safety Condition: every head variable must occur in some atom R_j.

[1]Relational Algebra: $\sigma, \Pi, \bowtie, \cup, -$.

Conjunctive Queries

Definition

A Conjunctive Query (CQ) is a formula of the form:

$$\boxed{Q(\boldsymbol{x}_0) = \exists \boldsymbol{y}\,(R_1(\boldsymbol{x}_1) \wedge R_2(\boldsymbol{x}_2) \wedge \cdots \wedge R_m(\boldsymbol{x}_m))}$$

- The free variables, $\boldsymbol{x}_0$, are called head variables.
- Safety Condition: every head variable must occur in some atom R_j.
- E.g. $Q(x, y) = \exists z(E(x, z) \wedge E(z, y))$.

[1] Relational Algebra: $\sigma, \Pi, \bowtie, \cup, -$.

Conjunctive Queries

Definition

A Conjunctive Query (CQ) is a formula of the form:

$$\boxed{Q(\boldsymbol{x}_0) = \exists \boldsymbol{y}\,(R_1(\boldsymbol{x}_1) \wedge R_2(\boldsymbol{x}_2) \wedge \cdots \wedge R_m(\boldsymbol{x}_m))}$$

- The free variables, $\boldsymbol{x}_0$, are called head variables.
- Safety Condition: every head variable must occur in some atom R_j.
- E.g. $Q(x, y) = \exists z(E(x, z) \wedge E(z, y))$.
- Same as FO formulas restricted to $=, \wedge, \exists$
  Same as RA[1] restricted to $\sigma, \Pi, \bowtie$.
  Same as SELECT-DISTINCT-FROM-WHERE

[1]Relational Algebra: $\sigma, \Pi, \bowtie, \cup, -$.

Unions of Conjunctive Queries

Definition

A Union of Conjunctive Queries (UCQ) is a formula of the form:

$$\boxed{Q(\boldsymbol{x}) = Q_1(\boldsymbol{x}) \vee Q_2(\boldsymbol{x}) \vee \cdots \vee Q_k(\boldsymbol{x})}$$

where all Q_i's are CQs, and have the same sets of free variables.

Unions of Conjunctive Queries

Definition

A Union of Conjunctive Queries (UCQ) is a formula of the form:

$$\boxed{Q(\boldsymbol{x}) = Q_1(\boldsymbol{x}) \vee Q_2(\boldsymbol{x}) \vee \cdots \vee Q_k(\boldsymbol{x})}$$

where all Q_i's are CQs, and have the same sets of free variables.

- E.g. $Q(x, y) = E(x, y) \bigvee \exists z(E(x, z) \wedge E(z, y))$.

Unions of Conjunctive Queries

Definition

A Union of Conjunctive Queries (UCQ) is a formula of the form:

$$\boxed{Q(\boldsymbol{x}) = Q_1(\boldsymbol{x}) \vee Q_2(\boldsymbol{x}) \vee \cdots \vee Q_k(\boldsymbol{x})}$$

where all Q_i's are CQs, and have the same sets of free variables.

- E.g. $Q(x, y) = E(x, y) \bigvee \exists z(E(x, z) \wedge E(z, y))$.
- Same as FO formulas restricted to $=, \wedge, \exists, \vee$.
  Same as RA restricted to $\sigma, \Pi, \bowtie, \cup$.

Unions of Conjunctive Queries

Definition

A Union of Conjunctive Queries (UCQ) is a formula of the form:

$$\boxed{Q(\boldsymbol{x}) = Q_1(\boldsymbol{x}) \vee Q_2(\boldsymbol{x}) \vee \cdots \vee Q_k(\boldsymbol{x})}$$

where all Q_i's are CQs, and have the same sets of free variables.

- E.g. $Q(x, y) = E(x, y) \bigvee \exists z(E(x, z) \wedge E(z, y))$.
- Same as FO formulas restricted to $=, \wedge, \exists, \vee$.
  Same as RA restricted to $\sigma, \Pi, \bowtie, \cup$.
- Notice: all queries $Q_i(\boldsymbol{x})$ must have the same head variables $\boldsymbol{x}$.

Monotone Queries

Given two databases $\boldsymbol{D}, \boldsymbol{D}'$ over the same schema, we write $\boldsymbol{D} \subseteq \boldsymbol{D}'$ if $R_i^D \subseteq R_i^{D'}$ for every relation R_i in the schema.

Definition

A query Q is monotone if $\boldsymbol{D} \subseteq \boldsymbol{D}'$ implies $Q(\boldsymbol{D}) \subseteq Q(\boldsymbol{D}')$.

Example: $\exists x, y, z(E(x, y) \wedge E(y, z))$ **Non-example**: $\exists x V(x) \wedge \forall y(V(y) \Rightarrow E(x, y))$

Monotone Queries

Given two databases $\boldsymbol{D}, \boldsymbol{D}'$ over the same schema, we write $\boldsymbol{D} \subseteq \boldsymbol{D}'$ if $R_i^D \subseteq R_i^{D'}$ for every relation R_i in the schema.

Definition

A query Q is monotone if $\boldsymbol{D} \subseteq \boldsymbol{D}'$ implies $Q(\boldsymbol{D}) \subseteq Q(\boldsymbol{D}')$.

Example: $\exists x, y, z(E(x,y) \wedge E(y,z))$ **Non-example**: $\exists x V(x) \wedge \forall y(V(y) \Rightarrow E(x,y))$

All UCQ queries are monotone. Exercise

The only non-monotone operators are:

- negation $\neg$ in FO.
- difference $-$ in RA.

Terminology

- Boolean query: no head vars: $Q() = \exists x \exists y \exists z (E(x,y) \wedge E(y,z))$.
- Full query: no existential vars: $Q(x,y,z) = E(x,y) \wedge E(y,z)$.
- Without selfjoins: every relation name occurs at most once.

$$Q(x) = \exists y \exists z (R(x,y) \wedge S(y,z) \wedge T(z,x)).$$

- We often omit the existential quantifiers, and write for example:

$$Q(x) = R(x,y) \wedge S(y,z) \wedge T(z,x).$$

Discussion

- A Boolean query Q is a sentence.

  "Evaluating Q on a database $\boldsymbol{D}$" means checking $\boldsymbol{D} \vDash Q$.

  We often write $Q(\boldsymbol{D}) = 1$ or $Q(\boldsymbol{D}) = 0$ respectively.

Discussion

- A Boolean query Q is a sentence.

  "Evaluating Q on a database $\boldsymbol{D}$" means checking $\boldsymbol{D} \vDash Q$.

  We often write $Q(\boldsymbol{D}) = 1$ or $Q(\boldsymbol{D}) = 0$ respectively.

- A query $Q(\boldsymbol{x})$ with head variables is a formula.

  The "answer of Q on $\boldsymbol{D}$" is $\{\boldsymbol{a} \mid \boldsymbol{a} \in (\mathrm{Dom}(\boldsymbol{D}))^{|\boldsymbol{x}|}, \boldsymbol{D} \vDash Q[\boldsymbol{a}/\boldsymbol{x}]\}$.

  We write $Q(\boldsymbol{D})$ for the set of answers, $Q(\boldsymbol{D}) \subseteq |\mathrm{Dom}(\boldsymbol{D})|^{|x|}$.

Model Checking for CQ

Motivation

We already know that the data complexity is in AC^0.

What is the expression complexity? The combined complexity?

Will answer both, and also discuss the expression/combined complexity for FO (which we left out).

Importantly: we will define query evaluation for CQ in terms of Homomorphisms

Equivalent Concepts

- A Conjunctive Query:
  $R(x, y, z) \wedge S(x, u) \wedge S(y, v) \wedge S(z, w) \wedge R(u, v, w)$

Equivalent Concepts

- A Conjunctive Query:
  $R(x,y,z) \wedge S(x,u) \wedge S(y,v) \wedge S(z,w) \wedge R(u,v,w)$
- A structure $\boldsymbol{A}$, a.k.a. a database instance:

$\mathrm{Dom}(\boldsymbol{A}) = \{x,y,z,u,v,w\}$

$R^A =$

x	y	z
u	v	w

$S^A =$

x	u
y	v
z	w

Equivalent Concepts

- A Conjunctive Query:
  $R(x,y,z) \wedge S(x,u) \wedge S(y,v) \wedge S(z,w) \wedge R(u,v,w)$
- A structure $\boldsymbol{A}$, a.k.a. a database instance:

$\mathrm{Dom}(\boldsymbol{A}) = \{x,y,z,u,v,w\}$

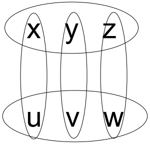
$R^A =$

x	y	z
u	v	w

$S^A =$

x	u
y	v
z	w

- A labeled hypergraph, $G = (V,E)$, where
  $V = \{x,y,z,u,v,w\}$, $E = \{\{x,y,z\},\{u,v,w\},\{x,u\},\{y,v\},\{z,w\}\}$
  (hyperedges are labeled with R, S respectively).



Equivalent Concepts

- A Conjunctive Query:
  $R(x,y,z) \wedge S(x,u) \wedge S(y,v) \wedge S(z,w) \wedge R(u,v,w)$
- A structure $\boldsymbol{A}$, a.k.a. a database instance:

$\mathrm{Dom}(\boldsymbol{A}) = \{x,y,z,u,v,w\}$

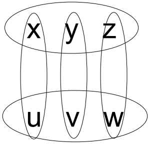
$R^A =$

x	y	z
u	v	w

$S^A =$

x	u
y	v
z	w

- A labeled hypergraph, $G = (V, E)$, where
  $V = \{x,y,z,u,v,w\}$, $E = \{\{x,y,z\},\{u,v,w\},\{x,u\},\{y,v\},\{z,w\}\}$
  (hyperedges are labeled with R, S respectively).



We will often switch back-and-forth between these equivalent notions

Homomorphisms

$\boldsymbol{A} = (\mathrm{Dom}(A), R_1^A, \ldots, R_m^A)$; $\boldsymbol{B} = (\mathrm{Dom}(B), R_1^B, \ldots, R_m^B)$.

Definition

A homomorphism $h : \boldsymbol{A} \to \boldsymbol{B}$ is a function $h : \mathrm{Dom}(A) \to \mathrm{Dom}(B)$ such that, for all $j = 1, m$, $h(R_j^A) \subseteq R_j^B$.

Homomorphisms

$\boldsymbol{A} = (\mathrm{Dom}(A), R_1^A, \ldots, R_m^A)$; $\boldsymbol{B} = (\mathrm{Dom}(B), R_1^B, \ldots, R_m^B)$.

Definition

A homomorphism $h : \boldsymbol{A} \to \boldsymbol{B}$ is a function $h : \mathrm{Dom}(A) \to \mathrm{Dom}(B)$ such that, for all $j = 1, m$, $h(R_j^A) \subseteq R_j^B$.

Graph homomorphism $h : G \to G'$ is $h : V \to V'$ s.t. $\forall e \in E$, $h(e) \in E'$.

Homomorphisms

$\boldsymbol{A} = (\mathrm{Dom}(A), R_1^A, \ldots, R_m^A)$; $\boldsymbol{B} = (\mathrm{Dom}(B), R_1^B, \ldots, R_m^B)$.

Definition

A homomorphism $h : \boldsymbol{A} \to \boldsymbol{B}$ is a function $h : \mathrm{Dom}(A) \to \mathrm{Dom}(B)$ such that, for all $j = 1, m$, $h(R_j^A) \subseteq R_j^B$.

Graph homomorphism $h : G \to G'$ is $h : V \to V'$ s.t. $\forall e \in E$, $h(e) \in E'$.

For query homomorphisms $h : Q(\boldsymbol{x}) \to Q'(\boldsymbol{x}')$ we also require h to map head variables to head variables and constants to constants.

Query Evaluation for CQ and Homomorphisms

If Q is Boolean, then $\boldsymbol{D} \models Q$ iff $\exists h : Q \to \boldsymbol{D}$.
If $Q(\boldsymbol{x})$ has head variables, then $Q(\boldsymbol{D}) = \{h(\boldsymbol{x}) \mid h : Q \to \boldsymbol{D}\}$.

Query Evaluation for CQ and Homomorphisms

If Q is Boolean, then $\boldsymbol{D} \vDash Q$ iff $\exists h : Q \to \boldsymbol{D}$.
If $Q(\boldsymbol{x})$ has head variables, then $Q(\boldsymbol{D}) = \{h(\boldsymbol{x}) \mid h : Q \to \boldsymbol{D}\}$.

$Q(x) = R(x) \wedge S(x, y) \wedge T(y, 'a')$

Query Evaluation for CQ and Homomorphisms

If Q is Boolean, then $\boldsymbol{D} \vDash Q$ iff $\exists h : Q \to \boldsymbol{D}$.
If $Q(\boldsymbol{x})$ has head variables, then $Q(\boldsymbol{D}) = \{h(\boldsymbol{x}) \mid h : Q \to \boldsymbol{D}\}$.

$Q(x) = R(x) \wedge S(x, y) \wedge T(y, 'a')$

$R =$

x
1
2
3

$S =$

x	y
1	10
1	20
2	20

$T =$

y	z
10	a
10	b
20	a

Query Evaluation for CQ and Homomorphisms

If Q is Boolean, then $\boldsymbol{D} \models Q$ iff $\exists h : Q \to \boldsymbol{D}$.
If $Q(\boldsymbol{x})$ has head variables, then $Q(\boldsymbol{D}) = \{h(\boldsymbol{x}) \mid h : Q \to \boldsymbol{D}\}$.

$Q(x) = R(x) \wedge S(x, y) \wedge T(y, 'a')$

$R =$

x
1
2
3

$S =$

x	y
1	10
1	20
2	20

$T =$

y	z
10	a
10	b
20	a

We list all homomorphisms $h : \{x, y\} \to \{1, 2, 10, 20, 'a'\}$:

Query Evaluation for CQ and Homomorphisms

If Q is Boolean, then $\boldsymbol{D} \vDash Q$ iff $\exists h : Q \to \boldsymbol{D}$.
If $Q(\boldsymbol{x})$ has head variables, then $Q(\boldsymbol{D}) = \{h(\boldsymbol{x}) \mid h : Q \to \boldsymbol{D}\}$.

$Q(x) = R(x) \wedge S(x,y) \wedge T(y,'a')$

$R =$

x
1
2
3

$S =$

x	y
1	10
1	20
2	20

$T =$

y	z
10	a
10	b
20	a

We list all homomorphisms $h : \{x, y\} \to \{1, 2, 10, 20, 'a'\}$:

$h =$

$x(=\mathtt{Head}(Q))$	y	a
1	10	a

Query Evaluation for CQ and Homomorphisms

If Q is Boolean, then $\boldsymbol{D} \vDash Q$ iff $\exists h : Q \to \boldsymbol{D}$.
If $Q(\boldsymbol{x})$ has head variables, then $Q(\boldsymbol{D}) = \{h(\boldsymbol{x}) \mid h : Q \to \boldsymbol{D}\}$.

$$Q(x) = R(x) \wedge S(x, y) \wedge T(y, 'a')$$

$R =$

x
1
2
3

$S =$

x	y
1	10
1	20
2	20

$T =$

y	z
10	a
10	b
20	a

We list all homomorphisms $h : \{x, y\} \to \{1, 2, 10, 20, 'a'\}$:

$h =$

$x(=\texttt{Head}(Q))$	y	a
1	10	a
1	20	a

Query Evaluation for CQ and Homomorphisms

If Q is Boolean, then $\boldsymbol{D} \vDash Q$ iff $\exists h : Q \to \boldsymbol{D}$.
If $Q(\boldsymbol{x})$ has head variables, then $Q(\boldsymbol{D}) = \{h(\boldsymbol{x}) \mid h : Q \to \boldsymbol{D}\}$.

$Q(x) = R(x) \wedge S(x, y) \wedge T(y, 'a')$

$R =$

x
1
2
3

$S =$

x	y
1	10
1	20
2	20

$T =$

y	z
10	a
10	b
20	a

We list all homomorphisms $h : \{x, y\} \to \{1, 2, 10, 20, 'a'\}$:

$h =$

$x(=\texttt{Head}(Q))$	y	a
1	10	a
1	20	a
2	20	a

Query Evaluation for CQ and Homomorphisms

If Q is Boolean, then $\boldsymbol{D} \vDash Q$ iff $\exists h : Q \to \boldsymbol{D}$.
If $Q(\boldsymbol{x})$ has head variables, then $Q(\boldsymbol{D}) = \{h(\boldsymbol{x}) \mid h : Q \to \boldsymbol{D}\}$.

$Q(x) = R(x) \wedge S(x, y) \wedge T(y, 'a')$

$R =$

x
1
2
3

$S =$

x	y
1	10
1	20
2	20

$T =$

y	z
10	a
10	b
20	a

We list all homomorphisms $h : \{x, y\} \to \{1, 2, 10, 20, 'a'\}$:

$h =$

$x(= \texttt{Head}(Q))$	y	a
1	10	a
1	20	a
2	20	a

Final answer after duplicate elimination: $Q(\boldsymbol{D}) = \{1, 2\}$.

The Combined Complexity for UCQ is in NP

Theorem

The combined complexity of Boolean UCQ is in NP.

Proof: Fix a UCQ $Q = Q_1 \vee Q_2 \vee \cdots$ and a database $\boldsymbol{D}$.

To check $\boldsymbol{D} \models Q$:

- "guess" a CQ Q_i, and
- "guess" a homomorphism $h : Q_i \to \boldsymbol{D}$

The Expression Complexity for CQ is NP-hard

Theorem

*There exists a database **D** for which the expression complexity of CQ queries is NP complete.*

Thus, both expression and combined complexities are NP-complete.

Proof Many proofs are possible (will explain shortly why). We will use reduction from 3SAT, because we will reuse it a few times.

The Expression Complexity for CQ is NP-hard

Theorem

There exists a database $\boldsymbol{D}$ for which the expression complexity of CQ queries is NP complete.

Thus, both expression and combined complexities are NP-complete.

Proof Many proofs are possible (will explain shortly why). We will use reduction from 3SAT, because we will reuse it a few times.

Given a 3CNF formula Φ we construct $Q_\Phi, \boldsymbol{D}$ such that:

The Expression Complexity for CQ is NP-hard

Theorem

There exists a database $\boldsymbol{D}$ for which the expression complexity of CQ queries is NP complete.

Thus, both expression and combined complexities are NP-complete.

Proof Many proofs are possible (will explain shortly why). We will use reduction from 3SAT, because we will reuse it a few times.

Given a 3CNF formula Φ we construct $Q_\Phi, \boldsymbol{D}$ such that:

$$\boxed{\Phi \text{ is satisfiable iff } \exists h : Q_\Phi \to \boldsymbol{D}.}$$

Notice that $\boldsymbol{D}$ is independent of Φ.

Details next.

Reduction from 3SAT to CQ Evaluation

Given a 3CNF formula Φ we construct $Q_\Phi, \boldsymbol{D}$ such that:

$$\boxed{\Phi \text{ is satisfiable iff } \exists h : Q_\Phi \to \boldsymbol{D}.}$$

Q_Φ has one atom for each clause C in Φ:

- If $C = (X_i \vee X_j \vee X_k)$ then Q_Φ contains $A(x_i, x_j, x_k)$.

Reduction from 3SAT to CQ Evaluation

Given a 3CNF formula Φ we construct $Q_\Phi, \boldsymbol{D}$ such that:

$$\boxed{\Phi \text{ is satisfiable iff } \exists h : Q_\Phi \to \boldsymbol{D}.}$$

Q_Φ has one atom for each clause C in Φ:

- If $C = (X_i \vee X_j \vee X_k)$ then Q_Φ contains $A(x_i, x_j, x_k)$.
- If $C = (X_i \vee X_j \vee \neg X_k)$ then Q_Φ contains $B(x_i, x_j, x_k)$.

Reduction from 3SAT to CQ Evaluation

Given a 3CNF formula Φ we construct $Q_\Phi, \boldsymbol{D}$ such that:

$$\boxed{\Phi \text{ is satisfiable iff } \exists h : Q_\Phi \to \boldsymbol{D}.}$$

Q_Φ has one atom for each clause C in Φ:

- If $C = (X_i \vee X_j \vee X_k)$ then Q_Φ contains $A(x_i, x_j, x_k)$.
- If $C = (X_i \vee X_j \vee \neg X_k)$ then Q_Φ contains $B(x_i, x_j, x_k)$.
- If $C = (X_i \vee \neg X_j \vee \neg X_k)$ then Q_Φ contains $C(x_i, x_j, x_k)$.
- If $C = (\neg X_i \vee \neg X_j \vee \neg X_k)$ then Q_Φ contains $D(x_i, x_j, x_k)$.

Reduction from 3SAT to CQ Evaluation

Given a 3CNF formula Φ we construct $Q_\Phi, \boldsymbol{D}$ such that:

$$\boxed{\Phi \text{ is satisfiable iff } \exists h : Q_\Phi \to \boldsymbol{D}.}$$

Q_Φ has one atom for each clause C in Φ:

- If $C = (X_i \vee X_j \vee X_k)$ then Q_Φ contains $A(x_i, x_j, x_k)$.
- If $C = (X_i \vee X_j \vee \neg X_k)$ then Q_Φ contains $B(x_i, x_j, x_k)$.
- If $C = (X_i \vee \neg X_j \vee \neg X_k)$ then Q_Φ contains $C(x_i, x_j, x_k)$.
- If $C = (\neg X_i \vee \neg X_j \vee \neg X_k)$ then Q_Φ contains $D(x_i, x_j, x_k)$.

$\boldsymbol{D}$ has 4 tables with 7 tuples each which tuple is missing?

$A = \begin{array}{|c|c|c|} \hline 0 & 0 & 1 \\ & \vdots & \\ 1 & 1 & 1 \\ \hline \end{array}$ $B = \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ 0 & 1 & 0 \\ & \vdots & \\ 1 & 1 & 1 \\ \hline \end{array}$ $C = \dots$ $D = \dots$

Reduction from 3SAT to CQ Evaluation

Given a 3CNF formula Φ we construct $Q_\Phi, \boldsymbol{D}$ such that:

$$\boxed{\Phi \text{ is satisfiable iff } \exists h : Q_\Phi \to \boldsymbol{D}.}$$

Q_Φ has one atom for each clause C in Φ:

- If $C = (X_i \vee X_j \vee X_k)$ then Q_Φ contains $A(x_i, x_j, x_k)$.
- If $C = (X_i \vee X_j \vee \neg X_k)$ then Q_Φ contains $B(x_i, x_j, x_k)$.
- If $C = (X_i \vee \neg X_j \vee \neg X_k)$ then Q_Φ contains $C(x_i, x_j, x_k)$.
- If $C = (\neg X_i \vee \neg X_j \vee \neg X_k)$ then Q_Φ contains $D(x_i, x_j, x_k)$.

$\boldsymbol{D}$ has 4 tables with 7 tuples each which tuple is missing?

$A =$

0	0	1
	⋮	
1	1	1

$B =$

0	0	0
0	1	0
	⋮	
1	1	1

$C = \ldots$ $D = \ldots$

In class: Φ is satisfiable iff $\exists h : Q \to \boldsymbol{D}$.

Example of 3SAT Reduction

Check if it is satisfiable:
$\Phi = (X \vee \neg Y \vee Z) \wedge (\neg X \vee Y \vee \neg Z) \wedge (X \vee Y \vee Z) \wedge (\neg X \vee \neg Y \vee Z)$.

Example of 3SAT Reduction

Check if it is satisfiable:
$\Phi = (X \vee \neg Y \vee Z) \wedge (\neg X \vee Y \vee \neg Z) \wedge (X \vee Y \vee Z) \wedge (\neg X \vee \neg Y \vee Z)$.

$Q = B(x, z, y) \wedge C(y, x, z) \wedge A(x, y, z) \wedge C(z, x, y)$

Example of 3SAT Reduction

Check if it is satisfiable:

$\Phi = (X \vee \neg Y \vee Z) \wedge (\neg X \vee Y \vee \neg Z) \wedge (X \vee Y \vee Z) \wedge (\neg X \vee \neg Y \vee Z)$.

$Q = B(x,z,y) \wedge C(y,x,z) \wedge A(x,y,z) \wedge C(z,x,y)$

$$A = \begin{array}{|c|c|c|} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array} \qquad B = \begin{array}{|c|c|c|} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array} \qquad C = \begin{array}{|c|c|c|} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array}$$

(no need for D)

Combined Complexity for FO

Recall that the combined complexity of FO is in PSPACE.

Theorem

*There exists a database **D** for which the expression complexity of FO queries is PSPACE complete.*

Thus, both expression and combined complexities are PSPACE-complete.

Combined Complexity for FO

Recall that the combined complexity of FO is in PSPACE.

Theorem

There exists a database $\mathbf{D}$ for which the expression complexity of FO queries is PSPACE complete.

Thus, both expression and combined complexities are PSPACE-complete.

Proof: Reduction from the Quantified Boolean Formula Satfiability:

$$\boxed{Q_1 X_1 \; Q_2 X_2 \cdots Q_n X_n \; \Phi}$$

where Φ is 3CNF.

Combined Complexity for FO

Recall that the combined complexity of FO is in PSPACE.

Theorem

There exists a database $\boldsymbol{D}$ for which the expression complexity of FO queries is PSPACE complete.

Thus, both expression and combined complexities are PSPACE-complete.

Proof: Reduction from the Quantified Boolean Formula Satfiability:

$$\boxed{Q_1X_1\ Q_2X_2 \cdots Q_nX_n\ \Phi}$$

where Φ is 3CNF.

Use the same $Q_\Phi, \boldsymbol{D}$ before, but add appropriate quantifiers to Q_Φ:

$$\boxed{Qx_1\ Qx_2 \cdots Q_nx_n\ Q_\Phi(x_1, \ldots, x_n)}$$

Example of QBE Reduction

Check if it is true:

$\Phi =$
$\forall X \exists Y \forall Z \, ((X \vee \neg Y \vee Z) \wedge (\neg X \vee Y \vee \neg Z) \wedge (X \vee Y \vee Z) \wedge (\neg X \vee \neg Y \vee Z))$

Example of QBE Reduction

Check if it is true:

$\Phi =$
$\forall X \exists Y \forall Z \left((X \vee \neg Y \vee Z) \wedge (\neg X \vee Y \vee \neg Z) \wedge (X \vee Y \vee Z) \wedge (\neg X \vee \neg Y \vee Z) \right)$

$Q = \forall x \exists y \forall z \left(B(x,z,y) \wedge C(y,x,z) \wedge A(x,y,z) \wedge C(z,x,y) \right)$

Example of QBE Reduction

Check if it is true:

$\Phi =$

$$\forall X \exists Y \forall Z \, ((X \vee \neg Y \vee Z) \wedge (\neg X \vee Y \vee \neg Z) \wedge (X \vee Y \vee Z) \wedge (\neg X \vee \neg Y \vee Z))$$

$$Q = \forall x \exists y \forall z \, (B(x, z, y) \wedge C(y, x, z) \wedge A(x, y, z) \wedge C(z, x, y))$$

$A =$

0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

$B =$

0	0	0
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

$C =$

0	0	0
0	0	1
0	1	0
1	0	0
1	0	1
1	1	0
1	1	1

(no need for D)

Discussion: CQ and CSP

The generalized Constraint Satisfaction Problem is:

Definition (Kolaitis&Vardi)

Given two classes of finite structures $\mathcal{A}, \mathcal{B}$, the $CSP(\mathcal{A}, \mathcal{B})$ problem is:
Given $A \in \mathcal{A}, B \in \mathcal{B}$, is there a homomorphism $h : A \to B$?

Discussion: CQ and CSP

The generalized Constraint Satisfaction Problem is:

Definition (Kolaitis&Vardi)

Given two classes of finite structures $\mathcal{A}, \mathcal{B}$, the $CSP(\mathcal{A}, \mathcal{B})$ problem is:
Given $A \in \mathcal{A}, B \in \mathcal{B}$, is there a homomorphism $h : A \to B$?

Standard CSP restricts the right-hand side, $CSP(-, B)$.
What is B for 3SAT? For 3-colorability? For Hamiltonean cycle?

Discussion: CQ and CSP

The generalized Constraint Satisfaction Problem is:

Definition (Kolaitis&Vardi)

Given two classes of finite structures $\mathcal{A}, \mathcal{B}$, the $CSP(\mathcal{A}, \mathcal{B})$ problem is:
Given $A \in \mathcal{A}, B \in \mathcal{B}$, is there a homomorphism $h : A \to B$?

Standard CSP restricts the right-hand side, $CSP(-, B)$.
What is B for 3SAT? For 3-colorability? For Hamiltonean cycle?

Query evaluation restricts the left-hand side, $CSP(Q, -)$

"Query evaluation is CSP from the other side."

Summary

- Evaluating $Q(\boldsymbol{D})$ consists of finding homomorphisms $h : Q \to \boldsymbol{D}$.
- This problem is in NP, in fact it is the very definition of NP.
- If Q is fixed, then the problem is in PTIME in $|\boldsymbol{D}|$. Data complexity
- If Q is part of the input (i.e. can be huge) then NP-complete. Expression complexity