Knowledge Bases and Databases

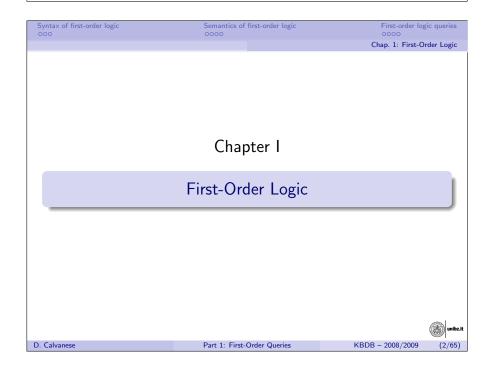
Part 1: First-Order Queries

Diego Calvanese

Faculty of Computer Science
Master of Science in Computer Science

A.Y. 2008/2009

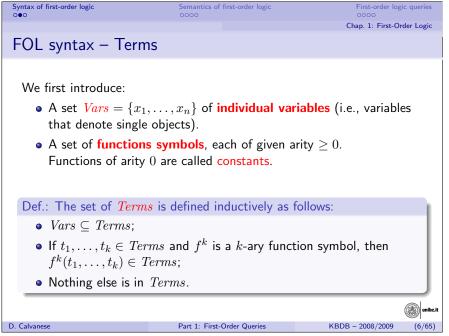


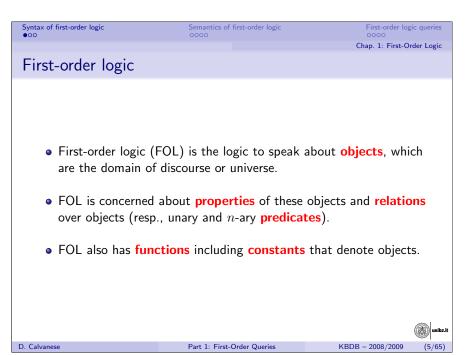


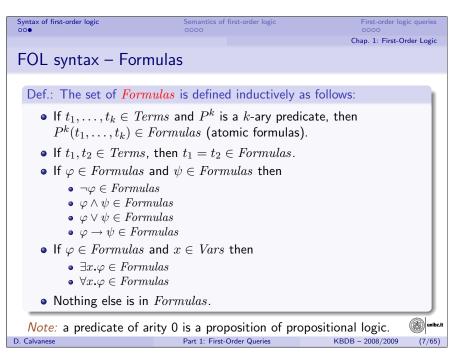
Overview of Part 1: First-order queries 1 First-order logic Syntax of first-order logic Semantics of first-order logic First-order logic queries 2 First-order query evaluation Query evaluation problem Complexity of query evaluation 2 Conjunctive queries Evaluation of conjunctive queries Containment of conjunctive queries Unions of conjunctive queries Unions of conjunctive queries











Syntax of first-order logic OOOO Chap. 1: First-Order Logic Outline

1 Syntax of first-order logic
2 Semantics of first-order logic
3 First-order logic queries

1 Syntax of first-order logic
2 Semantics of first-order logic
3 First-order logic queries

D. Calvanese Part 1: First-Order Queries KBDB – 2008/2009 (8/65)

Syntax of first-order logic

Semantics of first-order logic

Chap. 1: First-Order Logic

Assignment

D. Calvanese

Let *Vars* be a set of (individual) variables.

Def.: Given an interpretation \mathcal{I} , an **assignment** is a function

$$\alpha: Vars \longrightarrow \Delta^{\mathcal{I}}$$

that assigns to each variable $x \in Vars$ an object $\alpha(x) \in \Delta^{\mathcal{I}}$.

It is convenient to extend the notion of assignment to terms. We can do so by defining a function $\hat{\alpha}: Terms \longrightarrow \Delta^{\mathcal{I}}$ inductively as follows:

Part 1: First-Order Queries

- $\hat{\alpha}(x) = \alpha(x)$, if $x \in Vars$
- $\bullet \hat{\alpha}(f(t_1,\ldots,t_k)) = f^{\mathcal{I}}(\hat{\alpha}(t_1),\ldots,\hat{\alpha}(t_k))$

Note: for constants $\hat{\alpha}(c) = c^{\mathcal{I}}$.



KBDB - 2008/2009

Syntax of first-order logic

Semantics of first-order logic

Semantics of first-order logic

Sooo

Chap. 1: First-Order Logic

Interpretations

Given an **alphabet** of predicates P_1, P_2, \ldots and functions f_1, f_2, \ldots , each with an associated arity, a FOL **interpretation** is:

$$\mathcal{I} = (\Delta^{\mathcal{I}}, P_1^{\mathcal{I}}, P_2^{\mathcal{I}}, \dots, f_1^{\mathcal{I}}, f_2^{\mathcal{I}}, \dots)$$

where:

- $\Delta^{\mathcal{I}}$ is the domain (a set of objects)
- if P_i is a k-ary predicate, then $P_i^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}$ (k times)
- if f_i is a k-ary function, then $f_i^{\mathcal{I}}: \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}} \longrightarrow \Delta^{\mathcal{I}}$ (k times)
- if f_i is a constant (i.e., a 0-ary function), then $f_i^{\mathcal{I}}:()\longrightarrow \Delta^{\mathcal{I}}$ (i.e., f_i denotes exactly one object of the domain)



Calvanese

Part 1: First-Order Queries

KBDB - 2008/

(9/05)

Syntax of first-order logic

Semantics of first-order logic

0000

Chap. 1: First-Order Logic

Truth in an interpretation wrt an assignment

We define when a FOL formula φ is **true** in an interpretation \mathcal{I} wrt an assignment α , written $\mathcal{I}, \alpha \models \varphi$:

- $\mathcal{I}, \alpha \models P(t_1, \dots, t_k)$ if $(\hat{\alpha}(t_1), \dots, \hat{\alpha}(t_k)) \in P^{\mathcal{I}}$
- $\mathcal{I}, \alpha \models t_1 = t_2$ if $\hat{\alpha}(t_1) = \hat{\alpha}(t_2)$
- $\bullet \ \mathcal{I}, \alpha \models \neg \varphi \quad \text{ if } \mathcal{I}, \alpha \not\models \varphi$
- $\bullet \ \mathcal{I}, \alpha \models \varphi \wedge \psi \quad \text{ if } \mathcal{I}, \alpha \models \varphi \text{ and } \mathcal{I}, \alpha \models \psi$
- $\mathcal{I}, \alpha \models \varphi \lor \psi$ if $\mathcal{I}, \alpha \models \varphi$ or $\mathcal{I}, \alpha \models \psi$
- $\mathcal{I}, \alpha \models \varphi \rightarrow \psi$ if $\mathcal{I}, \alpha \models \varphi$ implies $\mathcal{I}, \alpha \models \psi$
- $\bullet \ \mathcal{I}, \alpha \models \exists x. \varphi \quad \text{if for some } a \in \Delta^{\mathcal{I}} \text{ we have } \mathcal{I}, \alpha[x \mapsto a] \models \varphi$
- $\bullet \ \mathcal{I}, \alpha \models \forall x. \varphi \quad \text{ if for every } a \in \Delta^{\mathcal{I}} \text{ we have } \mathcal{I}, \alpha[x \mapsto a] \models \varphi$

Here, $\alpha[x\mapsto a]$ stands for the new assignment obtained from α as follows:

$$\alpha[x \mapsto a](x) = a$$

 $\alpha[x \mapsto a](y) = \alpha(y)$ for $y \neq x$



Open vs. closed formulas

Definitions

- A variable x in a formula φ is **free** if x does not occur in the scope of any quantifier, otherwise it is **bound**.
- An open formula is a formula that has some free variable.
- A closed formula, also called sentence, is a formula that has no free variables.

For closed formulas (but not for open formulas) we can define what it means to be **true in an interpretation**, written $\mathcal{I} \models \varphi$, without mentioning the assignment, since the assignment α does not play any role in verifying $\mathcal{I}, \alpha \models \varphi$.

Instead, open formulas are strongly related to queries — cf. relational databases.

D. Calvanese

Part 1: First-Order Queries

KBDB - 2008/2009

First-order logic queries

Chap. 1: First-Order Logic

FOL queries

Def.: A **FOL query** is an (open) FOL formula.

When φ is a FOL query with free variables (x_1, \ldots, x_k) , then we sometimes write it as $\varphi(x_1,\ldots,x_k)$, and say that φ has **arity** k.

Given an interpretation \mathcal{I} , we are interested in those assignments that map the variables x_1, \ldots, x_k (and only those). We write an assignment α s.t. $\alpha(x_i) = a_i$, for $i = 1, \dots, k$, as $\langle a_1, \dots, a_k \rangle$.

Def.: Given an interpretation \mathcal{I} , the **answer to a query** $\varphi(x_1,\ldots,x_k)$ is

$$\varphi(x_1,\ldots,x_k)^{\mathcal{I}} = \{(a_1,\ldots,a_k) \mid \mathcal{I}, \langle a_1,\ldots,a_k \rangle \models \varphi(x_1,\ldots,x_k)\}$$

Note: We will also use the notation $\varphi^{\mathcal{I}}$, which keeps the free variables implicit, and $\varphi(\mathcal{I})$ making apparent that φ becomes a functions from interpretations to set of tuples.

Syntax of first-order logic First-order logic queries Chap. 1: First-Order Logic

Outline

- Syntax of first-order logic
- Semantics of first-order logic
- 3 First-order logic queries



Part 1: First-Order Queries

First-order logic queries

Chap. 1: First-Order Logic

FOL boolean gueries

Def.: A FOL boolean query is a FOL query without free variables.

Hence, the answer to a boolean query $\varphi()$ is defined as follows:

$$\varphi()^{\mathcal{I}} = \{() \mid \mathcal{I}, \langle \rangle \models \varphi()\}$$

Such an answer is

- (), if $\mathcal{I} \models \varphi$
- \emptyset , if $\mathcal{I} \not\models \varphi$.

As an obvious convention we read () as "true" and \emptyset as "false".



Syntax of first-order logic Semantics of first-order logic OOO Syntax of first-order logic OOO Syntax of first-order logic OOOO Syntax of first-order logic OOOO Chap. 1: First-Order Logic

FOL formulas: logical tasks

Definitions

- Validity: φ is valid iff for all \mathcal{I} and α we have that $\mathcal{I}, \alpha \models \varphi$.
- Satisfiability: φ is satisfiable iff there exists an \mathcal{I} and α such that $\mathcal{I}, \alpha \models \varphi$, and unsatisfiable otherwise.
- Logical implication: φ logically implies ψ , written $\varphi \models \psi$ iff for all \mathcal{I} and α , if $\mathcal{I}, \alpha \models \varphi$ then $\mathcal{I}, \alpha \models \psi$.
- Logical equivalence: φ is **logically equivalent** to ψ , iff for all \mathcal{I} and α , we have that $\mathcal{I}, \alpha \models \varphi$ iff $\mathcal{I}, \alpha \models \psi$ (i.e., $\varphi \models \psi$ and $\psi \models \varphi$).



D. Calvanese

D. Calvanese

Part 1: First-Order Queries

KBDB - 2008/2009

(16/65

Query evaluation problem

Complexity of query evaluation

00000

Chap. 2: First-Order Query Evaluation

Chapter II

First-Order Query Evaluation

Part 1: First-Order Queries



Syntax of first-order logic Semantics of first-order logic ooo First-order logic queries ooo Chap. 1: First-Order Logic

FOL queries - Logical tasks

- Validity: if φ is valid, then $\varphi^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \cdots \times \Delta^{\mathcal{I}}$ for all \mathcal{I} , i.e., the query always returns all the tuples of \mathcal{I} .
- Satisfiability: if φ is satisfiable, then $\varphi^{\mathcal{I}} \neq \emptyset$ for some \mathcal{I} , i.e., the query returns at least one tuple.
- Logical implication: if φ logically implies ψ , then $\varphi^{\mathcal{I}} \subseteq \psi^{\mathcal{I}}$ for all \mathcal{I} , written $\varphi \subseteq \psi$, i.e., the answer to φ is contained in that of ψ in every interpretation. This is called **query containment**.
- Logical equivalence: if φ is logically equivalent to ψ , then $\varphi^{\mathcal{I}} = \psi^{\mathcal{I}}$ for all \mathcal{I} , written $\varphi \equiv \psi$, i.e., the answer to the two queries is the same in every interpretation. This is called **query equivalence** and corresponds to query containment in both directions.

Note: These definitions can be extended to the case where we have axioms, i.e., constraints on the admissible interpretations.



D. Calvanese

Part 1: First-Order Queries

KBDB - 2008/2009

(17/05)

Query evaluation problem
Complexity of query evaluation
Ococo
Chap. 2: First-Order Query Evaluation

Outline

4 Query evaluation problem

5 Complexity of query evaluation



Query evaluation problem Complexity of query evaluation Chap. 2: First-Order Query Evaluation Outline

Query evaluation problem

Complexity of query evaluation



Part 1: First-Order Queries

KBDB - 2008/2009

Query evaluation problem

Chap. 2: First-Order Query Evaluation

Query evaluation problem

Definitions

• Query answering problem: given a finite interpretation \mathcal{I} and a FOL query $\varphi(x_1,\ldots,x_k)$, compute

$$\varphi^{\mathcal{I}} = \{(a_1, \dots, a_k) \mid \mathcal{I}, \langle a_1, \dots, a_k \rangle \models \varphi(x_1, \dots, x_k)\}$$

• Recognition problem (for query answering): given a finite interpretation \mathcal{I} , a FOL query $\varphi(x_1,\ldots,x_k)$, and a tuple (a_1,\ldots,a_k) , with $a_i\in\Delta^{\mathcal{I}}$, check whether $(a_1,\ldots,a_k)\in\varphi^{\mathcal{I}}$, i.e., whether

$$\mathcal{I}, \langle a_1, \dots, a_k \rangle \models \varphi(x_1, \dots, x_k)$$

Note: The recognition problem for query answering is the decision problem corresponding to the query answering problem.



Query evaluation problem

Complexity of query evaluation

Chap. 2: First-Order Query Evaluation

Query evaluation

Let us consider:

- a finite alphabet, i.e., we have a finite number of predicates and functions, and
- \bullet a finite interpretation \mathcal{I} , i.e., an interpretation (over the finite alphabet) for which $\Delta^{\mathcal{I}}$ is finite.

Then we can consider query evaluation as an algorithmic problem, and study its computational properties.

Note: To study the **computational complexity** of the problem, we need to define a corresponding decision problem.



Part 1: First-Order Queries

KBDB - 2008/2009

Query evaluation problem

Chap. 2: First-Order Query Evaluation

Query evaluation algorithm

We define now an algorithm that computes the function $Truth(\mathcal{I}, \alpha, \varphi)$ in such a way that $Truth(\mathcal{I}, \alpha, \varphi) = true iff \mathcal{I}, \alpha \models \varphi$.

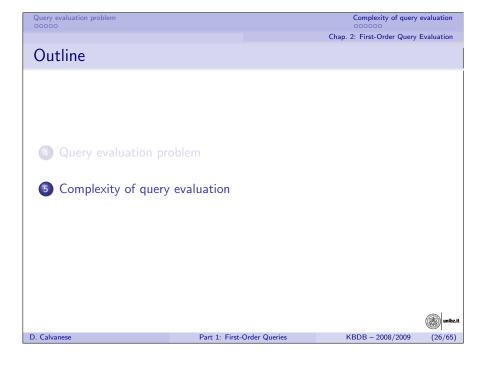
We make use of an auxiliary function $\mathtt{TermEval}(\mathcal{I}, \alpha, t)$ that, given an interpretation \mathcal{I} and an assignment α , evaluates a term t returning an object $o \in \Delta^{\mathcal{I}}$:

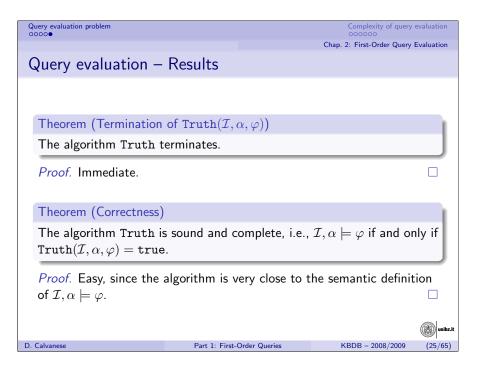
```
\Delta^{\mathcal{I}} TermEval(\mathcal{I}, \alpha, t) {
      if (t is x \in Vars)
            return \alpha(x);
      if (t \text{ is } f(t_{-1},\ldots,t_{-k}))
            return f^{\mathcal{I}}(\text{TermEval}(\mathcal{I}, \alpha, t_{-1}), \dots, \text{TermEval}(\mathcal{I}, \alpha, t_{-k}));
}
```

Then, Truth $(\mathcal{I}, \alpha, \varphi)$ can be defined by structural recursion on φ .



```
Query evaluation problem
                                                                                                            Complexity of query evaluation
                                                                                                    Chap. 2: First-Order Query Evaluation
Query evaluation algorithm (cont'd)
   boolean Truth(\mathcal{I}, \alpha, \varphi) {
      if (\varphi is t_1 = t_2)
         return TermEval(\mathcal{I}, \alpha, t_{-1}) = TermEval(\mathcal{I}, \alpha, t_{-2});
      if (\varphi is P(t_{-1},\ldots,t_{-k}))
         return P^{\mathcal{I}}(\text{TermEval}(\mathcal{I}, \alpha, t-1), \dots, \text{TermEval}(\mathcal{I}, \alpha, t-k));
      if (\varphi \text{ is } \neg \psi)
         return \neg Truth(\mathcal{I}, \alpha, \psi);
      if (\varphi \text{ is } \psi \circ \psi')
         return Truth(\mathcal{I}, \alpha, \psi) \circ Truth(\mathcal{I}, \alpha, \psi');
      if (\varphi \text{ is } \exists x.\psi) {
        boolean b = false;
         for all (a \in \Delta^{\mathcal{I}})
              b = b \vee Truth(\mathcal{I}, \alpha[x \mapsto a], \psi);
         return b;
      if (\varphi \text{ is } \forall x.\psi) {
         boolean b = true;
         for all (a \in \Delta^{\mathcal{I}})
               b = b \wedge Truth(\mathcal{I}, \alpha[x \mapsto a], \psi);
         return b;
  }
                                                        Part 1: First-Order Queries
                                                                                                          KBDB - 2008/2009
```





Chap. 2: First-Order Query Evaluation Query evaluation – Time complexity I

Theorem (Time complexity of $Truth(\mathcal{I}, \alpha, \varphi)$)

The time complexity of Truth $(\mathcal{I}, \alpha, \varphi)$ is $(|\mathcal{I}| + |\alpha| + |\varphi|)^{|\varphi|}$, i.e., polynomial in the size of \mathcal{I} and exponential in the size of φ .

Proof.

- $f^{\mathcal{I}}$ (of arity k) can be represented as k-dimensional array, hence accessing the required element can be done in time linear in $|\mathcal{I}|$.
- TermEval(...) visits the term, so it generates a polynomial number of recursive calls, hence is time polynomial in $(|\mathcal{I}| + |\alpha| + |\varphi|)$.



Complexity of query evaluation

••0000

Chap. 2: First-Order Query Evaluation

Query evaluation – Time complexity II

- $P^{\mathcal{I}}$ (of arity k) can be represented as k-dimensional boolean array, hence accessing the required element can be done in time linear in
- Truth(...) for the boolean cases simply visits the formula, so generates either one or two recursive calls.
- Truth(...) for the quantified cases $\exists x.\varphi$ and $\forall x.\psi$ involves looping for all elements in $\Delta^{\mathcal{I}}$ and testing the resulting assignments.
- The total number of such testings is $O(|\mathcal{I}|^{\sharp Vars})$.

Hence the claim holds.

Part 1: First-Order Queries

KBDB - 2008/2009

Part 1: First-Order Queries

• $f^{\mathcal{I}}(\ldots)$ can be represented as k-dimensional array, hence accessing

• TermEval(...) simply visits the term, so it generates a polynomial number of recursive calls. Each activation record has a constant

• $P^{\mathcal{I}}(...)$ can be represented as k-dimensional boolean array, hence

Complexity of query evaluation

Proof.

Chap. 2: First-Order Query Evaluation

Complexity of query evaluation

Chap. 2: First-Order Query Evaluation

Query evaluation - Complexity measures [Var82]

Query evaluation – Space complexity I

Theorem (Space complexity of $Truth(\mathcal{I}, \alpha, \varphi)$)

the required element requires $O(\log |\mathcal{I}|)$;

size, and we need $O(|\varphi|)$ activation records;

accessing the required element requires $O(\log |\mathcal{I}|)$;

The space complexity of Truth $(\mathcal{I}, \alpha, \varphi)$ is $|\varphi| \cdot (|\varphi| \cdot \log |\mathcal{I}|)$, i.e., logarithmic in the size of \mathcal{I} and polynomial in the size of φ .

Definition (Combined complexity)

The **combined complexity** is the complexity of $\{\langle \mathcal{I}, \alpha, \varphi \rangle \mid \mathcal{I}, \alpha \models \varphi \}$, i.e., interpretation, tuple, and guery are all considered part of the input.

Definition (Data complexity)

The **data complexity** is the complexity of $\{\langle \mathcal{I}, \alpha \rangle \mid \mathcal{I}, \alpha \models \varphi \}$, i.e., the query φ is fixed (and hence not considered part of the input).

Definition (Query complexity)

The **query complexity** is the complexity of $\{\langle \alpha, \varphi \rangle \mid \mathcal{I}, \alpha \models \varphi \}$, i.e., the interpretation \mathcal{I} is fixed (and hence not considered part of the input).



KBDB - 2008/2009

Chap. 2: First-Order Query Evaluation Query evaluation – Space complexity II

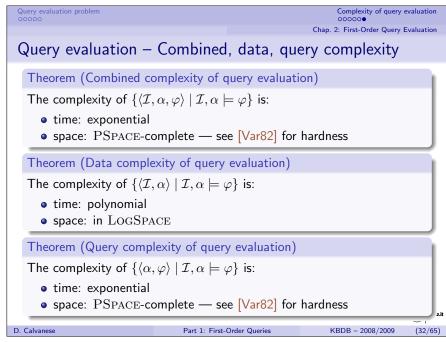
- Truth(...) for the boolean cases simply visits the formula, so generates either one or two recursive calls, each requiring constant size:
- Truth(...) for the quantified cases $\exists x.\varphi$ and $\forall x.\psi$ involves looping for all elements in $\Delta^{\mathcal{I}}$ and testing the resulting assignments:
- The total number of activation records that need to be at the same time on the stack is $O(\sharp Vars) \leq O(|\varphi|)$.

Hence the claim holds.

Note: the worst case form for the formula is

$$\forall x_1.\exists x_2.\cdots \forall x_{n-1}.\exists x_n.P(x_1,x_2,\ldots,x_{n-1},x_n).$$





Part 1: First-Order Queries

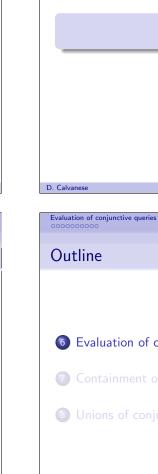
Outline

D. Calvanese

6 Evaluation of conjunctive queries

Containment of conjunctive queries

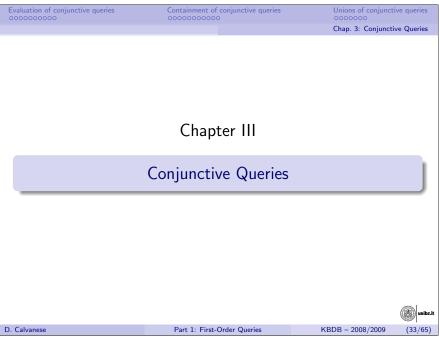
8 Unions of conjunctive queries



Chap. 3: Conjunctive Queries

KBDB - 2008/2009

(34/65)





Conjunctive queries (CQs)

Def.: A conjunctive query (CQ) is a FOL query of the form

$$\exists \vec{y}.conj(\vec{x},\vec{y})$$

where $conj(\vec{x}, \vec{y})$ is a conjunction (i.e., an "and") of atoms and equalities, over the free variables \vec{x} , the existentially quantified variables \vec{y} , and possibly constants.

Note:

- CQs contain no disjunction, no negation, no universal quantification, and no function symbols besides constants.
- Hence, they correspond to relational algebra select-project-join (SPJ) queries.
- CQs are the most frequently asked queries.



D. Calvanese

Part 1: First-Order Queries

KBDB - 2008/2009

(36/65)

Evaluation of conjunctive queries 00•0000000

Containment of conjunctive querie

Unions of conjunctive queries

Chap. 3: Conjunctive Queries

Datalog notation for CQs

A CQ $q = \exists \vec{y}.conj(\vec{x}, \vec{y})$ can also be written using datalog notation as

$$q(\vec{x}_1) \leftarrow conj'(\vec{x}_1, \vec{y}_1)$$

where $conj'(\vec{x}_1, \vec{y}_1)$ is the list of atoms in $conj(\vec{x}, \vec{y})$ obtained by equating the variables \vec{x} , \vec{y} according to the equalities in $conj(\vec{x}, \vec{y})$.

As a result of such an equality elimination, we have that \vec{x}_1 and \vec{y}_1 can contain constants and multiple occurrences of the same variable.

Def.: In the above query q, we call:

- $q(\vec{x}_1)$ the **head**;
- $conj'(\vec{x}_1, \vec{y}_1)$ the **body**;
- the variables in \vec{x}_1 the **distinguished variables**;
- the variables in \vec{y}_1 the non-distinguished variables.

D. Calvanese Part 1: First-Order Queries

KBDB - 2008/2009 (

-

Evaluation of conjunctive queries

Containment of conjunctive queries

Unions of conjunctive queries

Chap. 3: Conjunctive Queries

Conjunctive queries and SQL – Example

Relational alphabet:

Person(name, age), Lives(person, city), Manages(boss, employee)

Query: return name and age of all persons that live in the same city as their boss.

Expressed in SQL:

```
SELECT P.name, P.age
FROM Person P, Manages M, Lives L1, Lives L2
WHERE P.name = L1.person AND P.name = M.employee AND
M.boss = L2.person AND L1.city = L2.city
```

Expressed as a CQ: (the distinguished variables are the blue ones)

$$\exists b, e, p_1, c_1, p_2, c_2. \mathsf{Person}(n, a) \land \mathsf{Manages}(b, e) \land \mathsf{Lives}(p1, c1) \land \mathsf{Lives}(p2, c2) \land \\ n = p1 \land n = e \land b = p2 \land c1 = c2$$

Or simpler: $\exists b, c. \mathsf{Person}(n, a) \land \mathsf{Manages}(b, n) \land \mathsf{Lives}(n, c) \land \mathsf{Lives}(b, c)$



D. Calvanese

Part 1: First-Order Queries

KBDB - 2008/20

(37/65)

Evaluation of conjunctive queries

Containment of conjunctive queries

0000000

Chap. 3: Conjunctive Queries

Conjunctive queries – Example

- Consider an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, E^{\mathcal{I}})$, where $E^{\mathcal{I}}$ is a binary relation note that such interpretation is a (directed) graph.
- The following \mathbf{CQ} q returns all nodes that participate to a triangle in the graph:

$$\exists y, z. E(x, y) \land E(y, z) \land E(z, x)$$

ullet The query q in **datalog notation** becomes:

$$q(x) \leftarrow E(x, y), E(y, z), E(z, x)$$

• The query q in **SQL** is (we use Edge(f,s) for E(x,y):

SELECT E1.f

FROM Edge E1, Edge E2, Edge E3
WHERE E1.s = E2.f AND E2.s = E3.f AND E3.s = E1.f



Nondeterministic evaluation of CQs

Since a CQ contains only existential quantifications, we can evaluate it by:

- guessing a truth assignment for the non-distinguished variables;
- **evaluating** the resulting formula (that has no quantifications).

```
boolean ConjTruth(\mathcal{I}, \alpha, \exists \vec{y}.conj(\vec{x}, \vec{y})) { GUESS assignment \alpha[\vec{y} \mapsto \vec{a}] { return Truth(\mathcal{I}, \alpha[\vec{y} \mapsto \vec{a}], conj(\vec{x}, \vec{y})); }
```

where $\mathtt{Truth}(\mathcal{I},\alpha,\varphi)$ is defined as for FOL queries, considering only the required cases.



D. Calvanese

0000000000

Evaluation of conjunctive queries

Part 1: First-Order Queries

KBDB - 2008/2009

Chap. 3: Conjunctive Queries

(40/65)

Evaluation of conjunctive queries

Containment of conjunctive querie

Part 1: First-Order Queries

Unions of conjunctive queries

KBDB - 2008/2009

Unions of conjunctive queries

Chap. 3: Conjunctive Queries

Chap. 3: Conjunctive Queries

CQ evaluation - Combined, data, and guery complexity

Theorem (Combined complexity of CQ evaluation)

 $\{\langle \mathcal{I}, \alpha, q \rangle \mid \mathcal{I}, \alpha \models q \}$ is **NP-complete** — see below for hardness.

- time: exponential
- space: polynomial

Theorem (Data complexity of CQ evaluation)

 $\{\langle \mathcal{I}, \alpha \rangle \mid \mathcal{I}, \alpha \models q\}$ is in LogSpace

- time: polynomial
- space: logarithmic

Theorem (Query complexity of CQ evaluation)

 $\{\langle \alpha, q \rangle \mid \mathcal{I}, \alpha \models q \}$ is **NP-complete** — see below for hardness.

- time: exponential
- space: polynomial

D. Calvanese Part 1: First-Order Queries

KBDB - 2008/2009

(42/65)

Nondeterministic CQ evaluation algorithm

```
boolean Truth(\mathcal{I}, \alpha, \varphi) {
    if (\varphi is t.1 = t.2)
      return TermEval(\mathcal{I}, \alpha, t.1) = TermEval(\mathcal{I}, \alpha, t.2);
    if (\varphi is P(t.1, \ldots, t.k))
      return P^{\mathcal{I}}(TermEval(\mathcal{I}, \alpha, t.1),...,TermEval(\mathcal{I}, \alpha, t.k));
    if (\varphi is \psi \land \psi')
      return Truth(\mathcal{I}, \alpha, \psi) \land Truth(\mathcal{I}, \alpha, \psi');
}

\Delta^{\mathcal{I}} TermEval(\mathcal{I}, \alpha, t) {
    if (t is a variable x) return \alpha(x);
    if (t is a constant t) return t
```

3-colorability

Evaluation of conjunctive queries

An undirected graph is k-colorable if it is possible to assign to each node one of k colors in such a way that every two nodes connected by an edge have different colors.

Def.: **3-colorability** is the following decision problem

Given an undirected graph G = (V, E), is it 3-colorable?

Theorem

3-colorability is NP-complete.

We exploit 3-colorability to show NP -hardness of conjunctive query evaluation.



Ivanese Part 1: First-Order Queries

KBDB - 2008/2009

Reduction from 3-colorability to CQ evaluation

Let G = (V, E) be an undirected graph. We consider a relational alphabet consisting of a single binary relation Edge and define:

- An Interpretation: $\mathcal{I} = (\Delta^{\mathcal{I}}, \mathsf{Edge}^{\mathcal{I}})$ where:

 - Edge^{\mathcal{I}} = {(r,g), (g,r), (r,b), (b,r), (g,b), (b,g)}
- A conjunctive query: Let $V = \{x_1, \dots, x_n\}$, then consider the boolean conjunctive query defined as:

$$q_G = \exists x_1, \dots, x_n. \bigwedge_{(x_i, x_j) \in E} \mathsf{Edge}(x_i, x_j) \land \mathsf{Edge}(x_j, x_i)$$

Theorem

G is 3-colorable iff $\mathcal{I} \models q_G$.

unibz,i

Part 1: First-Order Queries

KBDB - 2008/2009

(44/65)

Containment of conjunctive gueries

Chap. 3: Conjunctive Queries

Outline

- Containment of conjunctive queries
- Unions of conjunctive queries



KBDB - 2008/2009

NP-hardness of CQ evaluation

The previous reduction immediately gives us the hardness for combined complexity.

Theorem

Evaluation of conjunctive queries

CQ evaluation is NP-hard in combined complexity.

Note: in the previous reduction, the interpretation does not depend on the actual graph. Hence, the reduction provides also the lower-bound for query complexity.

Theorem

CQ evaluation is NP-hard in query (and combined) complexity.



Part 1: First-Order Queries

Containment of conjunctive queries •0000000000

Unions of conjunctive queries

Chap. 3: Conjunctive Queries

Homomorphism

Let $\mathcal{I} = (\Delta^{\mathcal{I}}, P^{\mathcal{I}}, \dots, c^{\mathcal{I}}, \dots)$ and $\mathcal{J} = (\Delta^{\mathcal{J}}, P^{\mathcal{J}}, \dots, c^{\mathcal{J}}, \dots)$ be two interpretations over the same alphabet (for simplicity, we consider only constants as functions).

Def.: A **homomorphism** from \mathcal{I} to \mathcal{J}

is a mapping $h: \Delta^{\mathcal{I}} \to \Delta^{\mathcal{J}}$ such that:

- $\bullet h(c^{\mathcal{I}}) = c^{\mathcal{J}}$
- $h(P^{\mathcal{I}}(a_1, \dots, a_k)) = P^{\mathcal{J}}(h(a_1), \dots, h(a_k))$

Note: An isomorphism is a homomorphism that is one-to-one and onto.

Theorem

FOL is unable to distinguish between interpretations that are isomorphic.

Proof. See any standard book on logic.



Part 1: First-Order Queries

KBDB - 2008/2009

Consider the recognition problem associated to the evaluation of a query q of arity k. Then

$$\mathcal{I}, \alpha \models q(x_1, \dots, x_k)$$
 iff $\mathcal{I}_{\alpha, \vec{c}} \models q(c_1, \dots, c_k)$

where $\mathcal{I}_{\alpha,\vec{c}}$ is identical to \mathcal{I} but includes new constants c_1,\ldots,c_k that are interpreted as $c_i^{\mathcal{I}_{\alpha},\vec{c}}=\alpha(x_i)$.

That is, we can reduce the recognition problem to the evaluation of a boolean query.



D. Calvanese

Part 1: First-Order Queries

KBDB - 2008/2009

(48/65)

2/65)

valuation of conjunctive queries

Containment of conjunctive queries

Unions of conjunctive queries

Chap. 3: Conjunctive Queries

Canonical interpretation of a (boolean) CQ – Example

Consider the boolean query \boldsymbol{q}

$$q(c) \leftarrow E(c, y), E(y, z), E(z, c)$$

Then, the canonical interpretation \mathcal{I}_q is defined as

$$\mathcal{I}_q = (\Delta^{\mathcal{I}_q}, E^{\mathcal{I}_q}, c^{\mathcal{I}_q})$$

where

D. Calvanese

- $\bullet \ \Delta^{\mathcal{I}_q} = \{y, z, c\}$
- $E^{\mathcal{I}_q} = \{(c, y), (y, z), (z, c)\}$
- $c^{\mathcal{I}_q} = c$



Evaluation of conjunctive queries

Containment of conjunctive queries

Unions of conjunctive queries

Chap. 3: Conjunctive Queries

Canonical interpretation of a (boolean) CQ

Let q be a conjunctive query $\exists x_1, \ldots, x_n \cdot conj$

Def.: The **canonical interpretation** \mathcal{I}_q associated with q

is the interpretation $\mathcal{I}_q = (\Delta^{\mathcal{I}_q}, P^{\mathcal{I}_q}, \dots, c^{\mathcal{I}_q}, \dots)$, where

- $\Delta^{\mathcal{I}_q} = \{x_1, \dots, x_n\} \cup \{c \mid c \text{ constant occurring in } q\}$, i.e., all the variables and constants in q;
- $c^{\mathcal{I}_q} = c$, for each constant c in q;
- $(t_1, \ldots, t_k) \in P^{\mathcal{I}_q}$ iff the atom $P(t_1, \ldots, t_k)$ occurs in q.

Sometimes the procedure for obtaining the canonical interpretation is called **freezing** of q.



D. Calvanes

Part 1: First-Order Queries

KBDB - 2008/200

(49/03)

Evaluation of conjunctive querie

Containment of conjunctive queries

0000000

Chap. 3: Conjunctive Queries

Canonical interpretation and (boolean) CQ evaluation

Theorem ([CM77])

For boolean CQs, $\mathcal{I} \models q$ iff there exists a homomorphism from \mathcal{I}_q to \mathcal{I} .

Proof.

" \Rightarrow " Let $\mathcal{I} \models q$, let α be an assignment to the existential variables that makes q true in \mathcal{I} , and let $\hat{\alpha}$ be its extension to constants. Then $\hat{\alpha}$ is a homomorphism from \mathcal{I}_q to \mathcal{I} .

" \Leftarrow " Let h be a homomorphism from \mathcal{I}_q to \mathcal{I} . Then restricting h to the variables only we obtain an assignment to the existential variables that makes q true in \mathcal{I} .



The previous result can be rephrased as follows:

(The recognition problem associated to) query evaluation can be reduced to finding a homomorphism.

Finding a homomorphism between two interpretations (aka relational structures) is also known as solving a Constraint Satisfaction **Problem** (CSP), a problem well-studied in AI – see also [KV98].



D. Calvanese

Part 1: First-Order Queries

KBDB - 2008/2009

Containment of conjunctive gueries

Chap. 3: Conjunctive Queries

Query containment for CQs

For CQs, query containment $q_1(\vec{x}) \subseteq q_2(\vec{x})$ can be reduced to query evaluation.

- Freeze the free variables, i.e., consider them as constants. This is possible, since $q_1(\vec{x}) \subseteq q_2(\vec{x})$ iff
 - $\mathcal{I}, \alpha \models q_1(\vec{x})$ implies $\mathcal{I}, \alpha \models q_2(\vec{x})$, for all \mathcal{I} and α ; or equivalently
 - $\mathcal{I}_{\alpha,\vec{c}} \models q_1(\vec{c})$ implies $\mathcal{I}_{\alpha,\vec{c}} \models q_2(\vec{c})$, for all $\mathcal{I}_{\alpha,\vec{c}}$, where \vec{c} are new constants, and $\mathcal{I}_{\alpha,\vec{c}}$ extends \mathcal{I} to the new constants with $c^{\mathcal{I}_{\alpha,\vec{c}}} = \alpha(x)$.
- **2** Construct the canonical interpretation $\mathcal{I}_{q_1(\vec{c})}$ of the CQ $q_1(\vec{c})$ on the left hand side
- **3** ... and evaluate on $\mathcal{I}_{q_1(\vec{c})}$ the CQ $q_2(\vec{c})$ on the right hand side, i.e., check whether $\mathcal{I}_{q_1(\vec{c})} \models q_2(\vec{c})$.



Containment of conjunctive queries Evaluation of conjunctive queries

Unions of conjunctive queries Chap. 3: Conjunctive Queries

Query containment

Def.: Query containment

Given two FOL queries φ and ψ of the same arity, φ is contained in ψ , denoted $\varphi \subseteq \psi$, if for all interpretations \mathcal{I} and all assignments α we have that

$$\mathcal{I}, \alpha \models \varphi \quad \text{implies} \quad \mathcal{I}, \alpha \models \psi$$

(In logical terms: $\varphi \models \psi$.)

Note: Query containment is of special interest in query optimization.

Theorem

For FOL queries, query containment is undecidable.

Proof.: Reduction from FOL logical implication.



Part 1: First-Order Queries

Containment of conjunctive queries

Chap. 3: Conjunctive Queries

Reducing containment of CQs to CQ evaluation

Theorem ([CM77])

For CQs, $q_1(\vec{x}) \subseteq q_2(\vec{x})$ iff $\mathcal{I}_{q_1(\vec{c})} \models q_2(\vec{c})$, where \vec{c} are new constants.

Proof.

" \Rightarrow " Assume that $q_1(\vec{x}) \subseteq q_2(\vec{x})$.

• Since $\mathcal{I}_{q_1(\vec{c})} \models q_1(\vec{c})$ it follows that $\mathcal{I}_{q_1(\vec{c})} \models q_2(\vec{c})$.

"\(\sim \)" Assume that $\mathcal{I}_{q_1(\vec{c})} \models q_2(\vec{c})$.

- By [CM77] on hom., for every \mathcal{I} such that $\mathcal{I} \models q_1(\vec{c})$ there exists a homomorphism h from $\mathcal{I}_{q_1(\vec{c})}$ to \mathcal{I} .
- On the other hand, since $\mathcal{I}_{q_1(\vec{c})}\models q_2(\vec{c})$, again by [CM77] on hom., there exists a homomorphism h' from $\mathcal{I}_{q_2(\vec{c})}$ to $\mathcal{I}_{q_1(\vec{c})}$.
- The mapping $h \circ h'$ (obtained by composing h and h') is a homomorphism from $\mathcal{I}_{q_2(\vec{c})}$ to \mathcal{I} . Hence, once again by [CM77] on hom., $\mathcal{I} \models q_2(\vec{c})$.

So we can conclude that $q_1(\vec{c}) \subseteq q_2(\vec{c})$, and hence $q_1(\vec{x}) \subseteq q_2(\vec{x})$.



Evaluation of conjunctive queries

For CQs, we also have that (boolean) guery evaluation $\mathcal{I} \models q$ can be reduced to guery containment.

Containment of conjunctive queries

Let
$$\mathcal{I} = (\Delta^{\mathcal{I}}, P^{\mathcal{I}}, \dots, c^{\mathcal{I}}, \dots)$$
.

We construct the (boolean) CQ q_T as follows:

- q_T has no existential variables (hence no variables at all);
- the constants in q_T are the elements of Δ^T ;
- ullet for each relation P interpreted in ${\mathcal I}$ and for each fact $(a_1,\ldots,a_k)\in P^{\mathcal{I}},\ q_{\mathcal{I}}$ contains one atom $P(a_1,\ldots,a_k)$ (note that each $a_i \in \Delta^{\mathcal{I}}$ is a constant in $q_{\mathcal{I}}$).

Theorem

For CQs, $\mathcal{I} \models q$ iff $q_{\mathcal{I}} \subseteq q$.



D. Calvanese

Part 1: First-Order Queries

KBDB - 2008/2009

KBDB - 2008/2009

Unions of conjunctive queries

Unions of conjunctive queries

(56/65)

Chap. 3: Conjunctive Queries Outline 6 Evaluation of conjunctive queries Containment of conjunctive queries Unions of conjunctive queries

Part 1: First-Order Queries

Containment of conjunctive queries Unions of conjunctive queries Evaluation of conjunctive queries Chap. 3: Conjunctive Queries Query containment for CQs - Complexity From the previous results and NP-completenss of combined complexity of CQ evaluation, we immediately get: Theorem Containment of CQs is NP-complete. Since CQ evaluation is NP-complete even in query complexity, the

above result can be strengthened:

Theorem

Containment $q_1(\vec{x}) \subseteq q_2(\vec{x})$ of CQs is NP-complete, even when q_1 is considered fixed



Part 1: First-Order Queries

Containment of conjunctive queries

•000000

Unions of conjunctive queries

Chap. 3: Conjunctive Queries

Union of conjunctive queries (UCQs)

Def.: A union of conjunctive queries (UCQ) is a FOL query of the form

$$\bigvee_{i=1}^{n} \exists \vec{y}_i.conj_i(\vec{x}, \vec{y}_i)$$

where each $conj_i(\vec{x}, \vec{y_i})$ is a conjunction of atoms and equalities with free variables \vec{x} and $\vec{y_i}$, and possibly constants.

Note: Obviously, each conjunctive query is also a of union of conjunctive gueries.



Evaluation of conjunctive queries

Containment of conjunctive queries

Unions of conjunctive queries

Chap. 3: Conjunctive Queries

Datalog notation for UCQs

A union of conjunctive queries

$$q = \bigvee_{i=1,\dots,n} \exists \vec{y}_i.conj_i(\vec{x},\vec{y}_i)$$

is written in datalog notation as

$$\left\{ \begin{array}{ll} q(\vec{x}) & \leftarrow & conj_1'(\vec{x}, \vec{y_1}') \\ & \vdots & \\ q(\vec{x}) & \leftarrow & conj_n'(\vec{x}, \vec{y_n}') \end{array} \right\}$$

where each element of the set is the datalog expression corresponding to the conjunctive query $q_i = \exists \vec{y_i}.conj_i(\vec{x},\vec{y_i})$.

Note: in general, we omit the set brackets.



D. Calvanese

Part 1: First-Order Queries

KBDB - 2008/2009

(60/65)

nctive queries

Unions of conjunctive queries 000 • 000 Chap. 3: Conjunctive Queries

UCQ evaluation — Combined, data, and query complexity

Theorem (Combined complexity of UCQ evaluation)

 $\{\langle \mathcal{I}, \alpha, q \rangle \mid \mathcal{I}, \alpha \models q \}$ is **NP-complete**.

- time: exponential
- space: polynomial

Theorem (Data complexity of UCQ evaluation)

 $\{\langle \mathcal{I}, q \rangle \mid \mathcal{I}, \alpha \models q\}$ is **in LogSpace** (query q fixed).

- time: polynomial
- space: logarithmic

Theorem (Query complexity of UCQ evaluation)

 $\{\langle \alpha, q \rangle \mid \mathcal{I}, \alpha \models q\}$ is **NP-complete** (interpretation \mathcal{I} fixed).

- time: exponential
- space: polynomial

D. Calvanese Part 1: First-Order Queries

KBDB - 2008/2009

(62/65)

Containment of conjunctive queries

Unions of conjunctive queries

Chap. 3: Conjunctive Queries

Evaluation of UCQs

Evaluation of conjunctive queries

From the definition of FOL query we have that:

$$\mathcal{I}, \alpha \models \bigvee_{i=1,\dots,n} \exists \vec{y_i}.conj_i(\vec{x}, \vec{y_i})$$

if and only if

$$\mathcal{I}, \alpha \models \exists \vec{y_i}.conj_i(\vec{x}, \vec{y_i})$$
 for some $i \in \{1, \dots, n\}$.

Hence to evaluate a UCQ q, we simply evaluate a number (linear in the size of q) of conjunctive queries in isolation.

Hence, evaluating UCQs has the same complexity as evaluating CQs.

unibz.

D. Calvanese

Part 1: First-Order Queries

KBDB - 2008/200

(01/0

Unions of conjunctive queries

Chap. 3: Conjunctive Queries

Query containment for UCQs

Theorem

For UCQs, $\{q_1,\ldots,q_k\}\subseteq\{q'_1,\ldots,q'_n\}$ iff for each q_i there is a q'_j such that $q_i\subseteq q'_j$.

Proof.

"

—" Obvious.

" \Rightarrow " If the containment holds, then we have $\{q_1(\vec{c}), \dots, q_k(\vec{c})\} \subseteq \{q_1'(\vec{c}), \dots, q_n'(\vec{c})\}$, where \vec{c} are new constants:

- Now consider $\mathcal{I}_{q_i(\vec{c})}$. We have $\mathcal{I}_{q_i(\vec{c})} \models q_i(\vec{c})$, and hence $\mathcal{I}_{q_i(\vec{c})} \models \{q_1(\vec{c}), \dots, q_k(\vec{c})\}.$
- By the containment, we have that $\mathcal{I}_{q_i(\vec{c})} \models \{q'_1(\vec{c}), \dots, q'_n(\vec{c})\}$. I.e., there exists a $q'_j(\vec{c})$ such that $\mathcal{I}_{q_i(\vec{c})} \models q'_j(\vec{c})$.
- ullet Hence, by [CM77] on containment of CQs, we have that $q_i\subseteq q_j'$.



nnese Part 1: First-Order Queries

KBDB - 2008/2009

(63/65)

Evaluation of conjunctive queries

Containment of conjunctive queries

Unions of conjunctive queries

Chap. 3: Conjunctive Queries

Query containment for UCQs - Complexity

From the previous result, we have that we can check $\{q_1, \ldots, q_k\} \subseteq \{q'_1, \ldots, q'_n\}$ by at most $k \cdot n$ CQ containment checks.

We immediately get:

Theorem

Containment of UCQs is NP-complete.



D. Calvanese Part 1: First-Order Queries

KBDB - 2008/2009

64/65)

Evaluation of conjunctive queries

Containment of conjunctive queries

Unions of conjunctive queries

Chap. 3: Conjunctive Queries

References

[CM77] A. K. Chandra and P. M. Merlin.

Optimal implementation of conjunctive queries in relational data bases.

In Proc. of the 9th ACM Symp. on Theory of Computing (STOC'77), pages 77–90, 1977.

[KV98] P. G. Kolaitis and M. Y. Vardi.

Conjunctive-query containment and constraint satisfaction.

In Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98), pages 205–213, 1998.

[Var82] M. Y. Vardi.

The complexity of relational query languages.

In Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82), pages 137–146, 1982.



D. Calvanese Part 1: First-Order Queries KBDB – 2008/2009