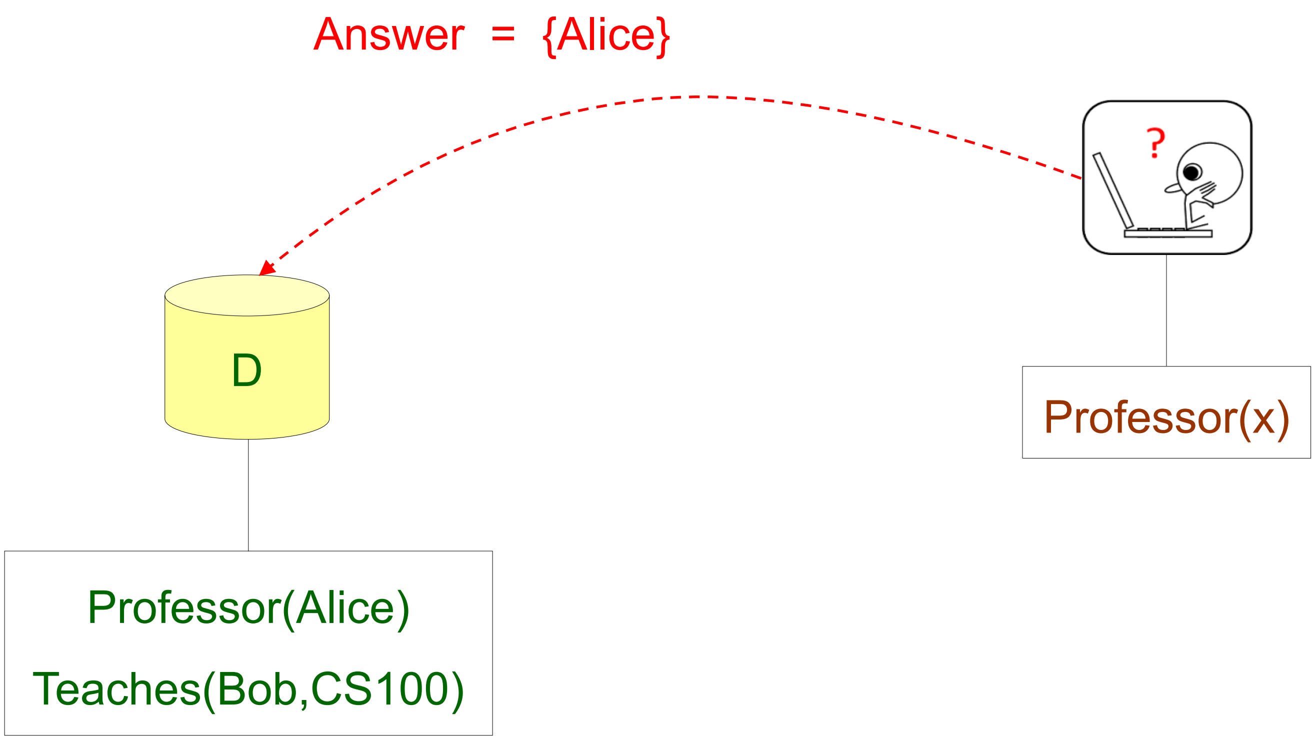
Reasoning with Data

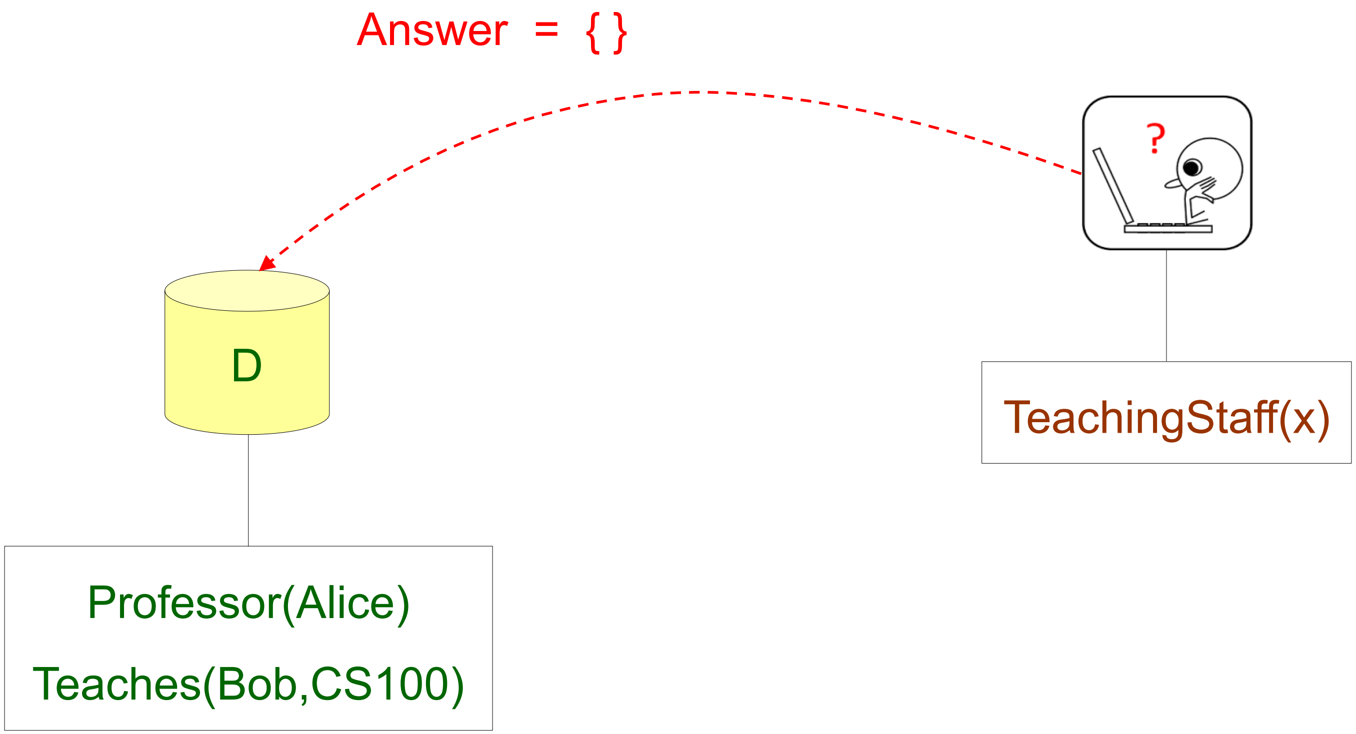
- Names: Ontology Based Query Answering
- Sometimes OBDA (Ontology Based Data Access)

- Scenario:
    - data is incomplete
    - but is supplemented with additional knowledge
    - typically in the form of an ontology
    - query answering takes into account both

Ontology-based Query Answering



Ontology-based Query Answering



Ontology-based Query Answering

Professors are teaching staff

Someone who teaches is teaching staff

O

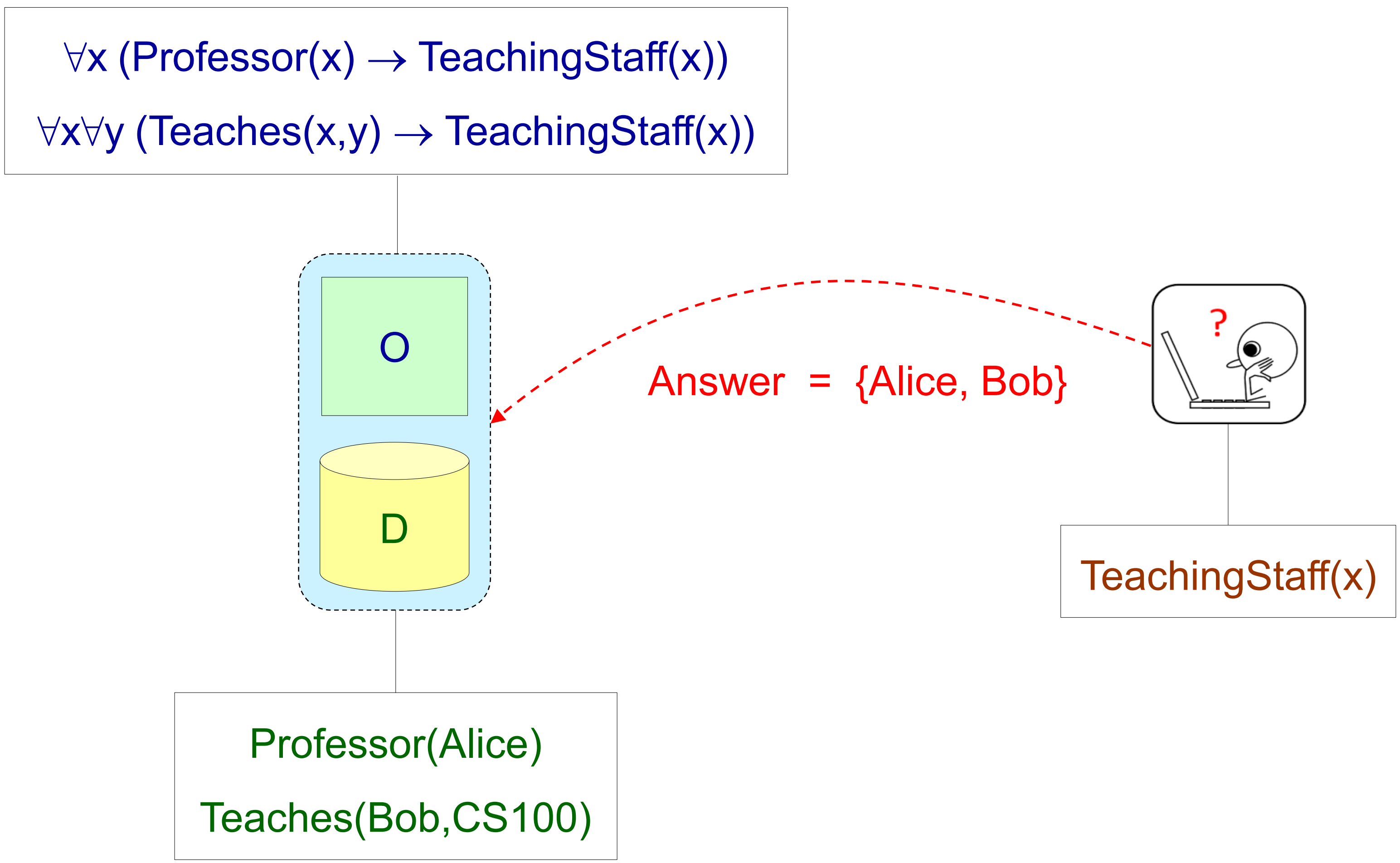
D

Professor(Alice)

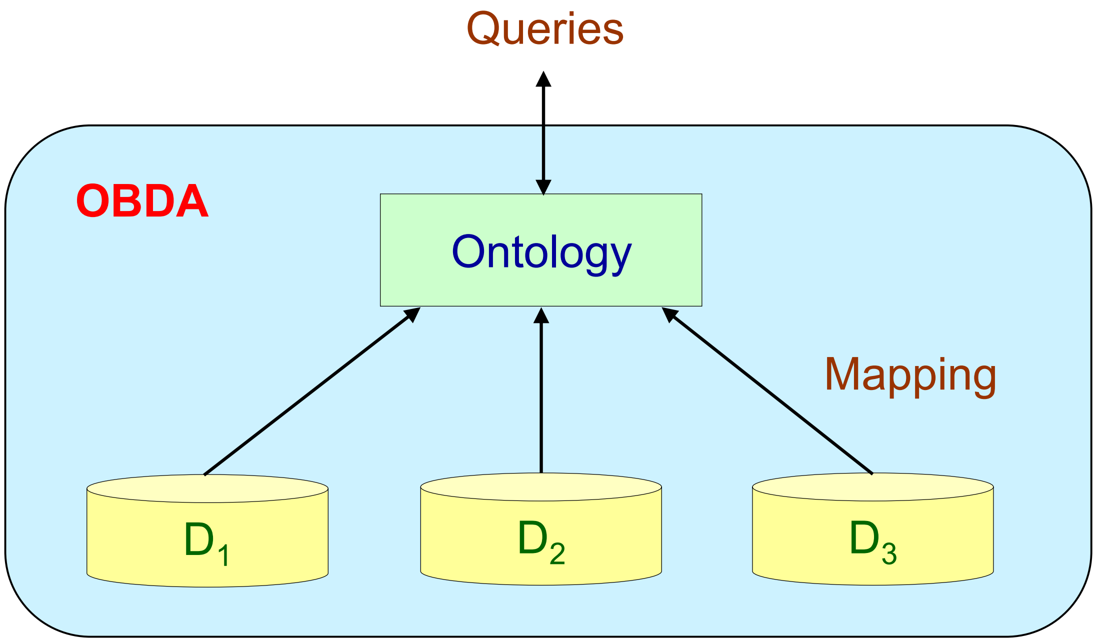
Teaches(Bob,CS100)

TeachingStaff(x)

Ontology-based Query Answering

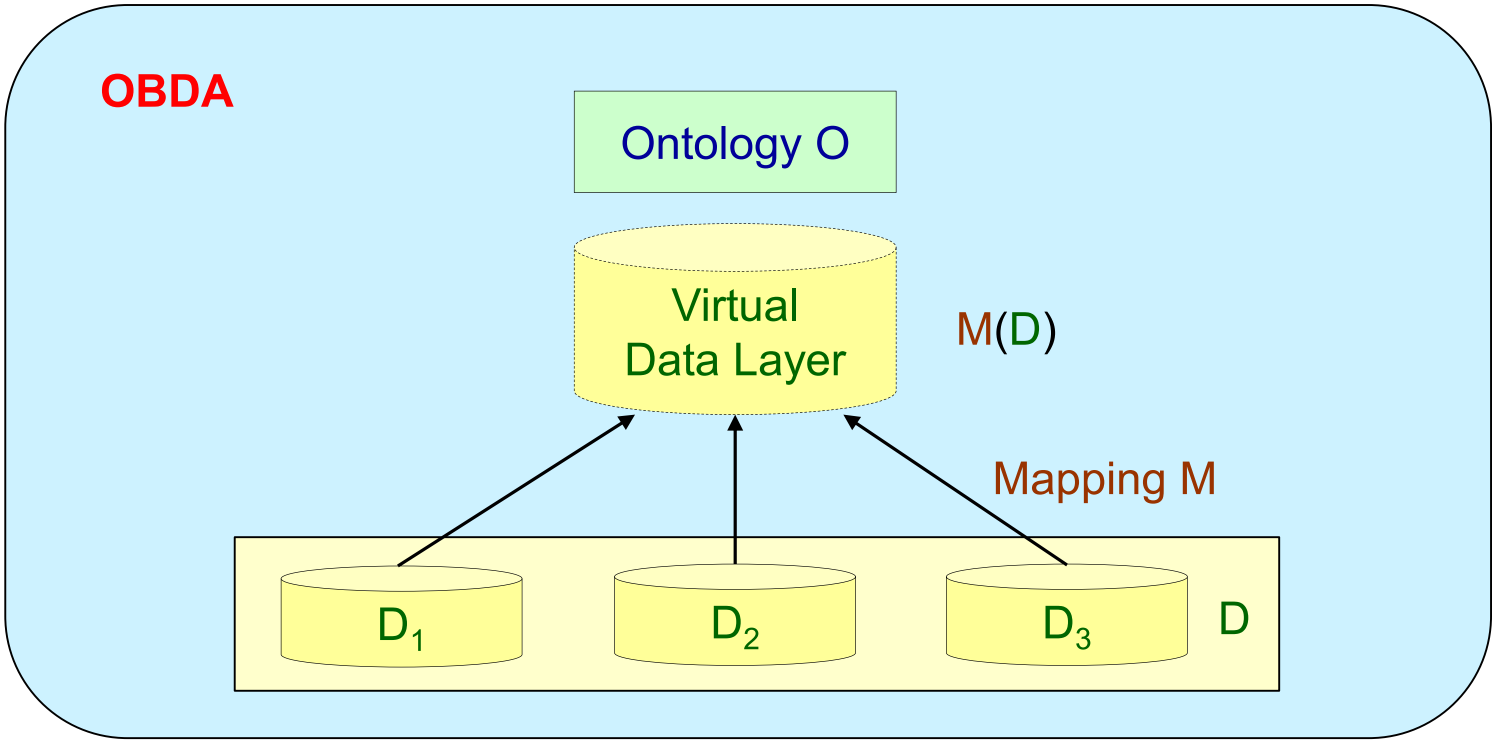


Ontology-based Data Access: Architecture



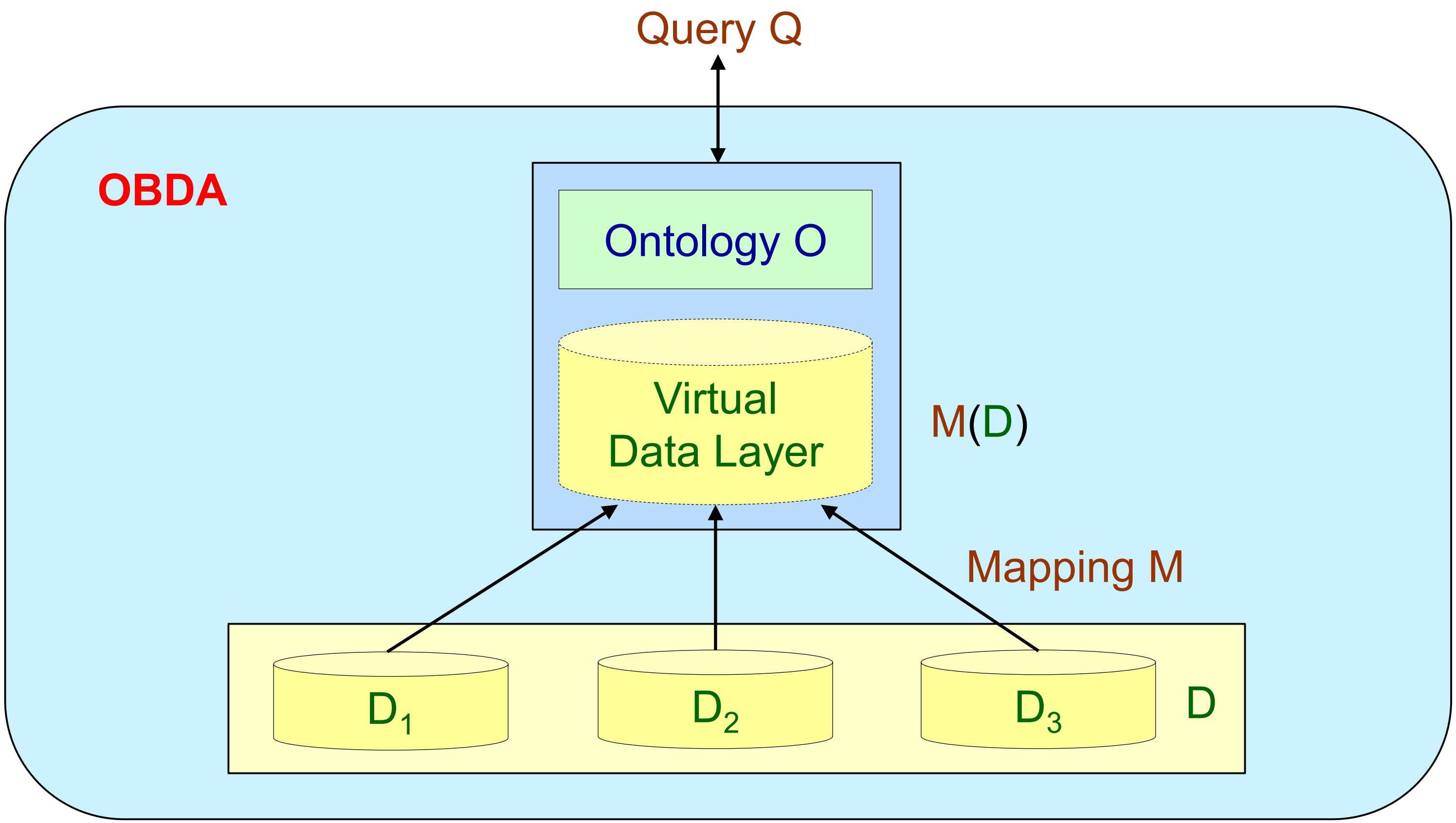
- Ontology: provides a unified conceptual "global view" of the data
- Data Sources: external and independent (possibly multiple and heterogeneous)
- Mapping: semantically link data at the sources with the ontology

Query Answering in OBDA



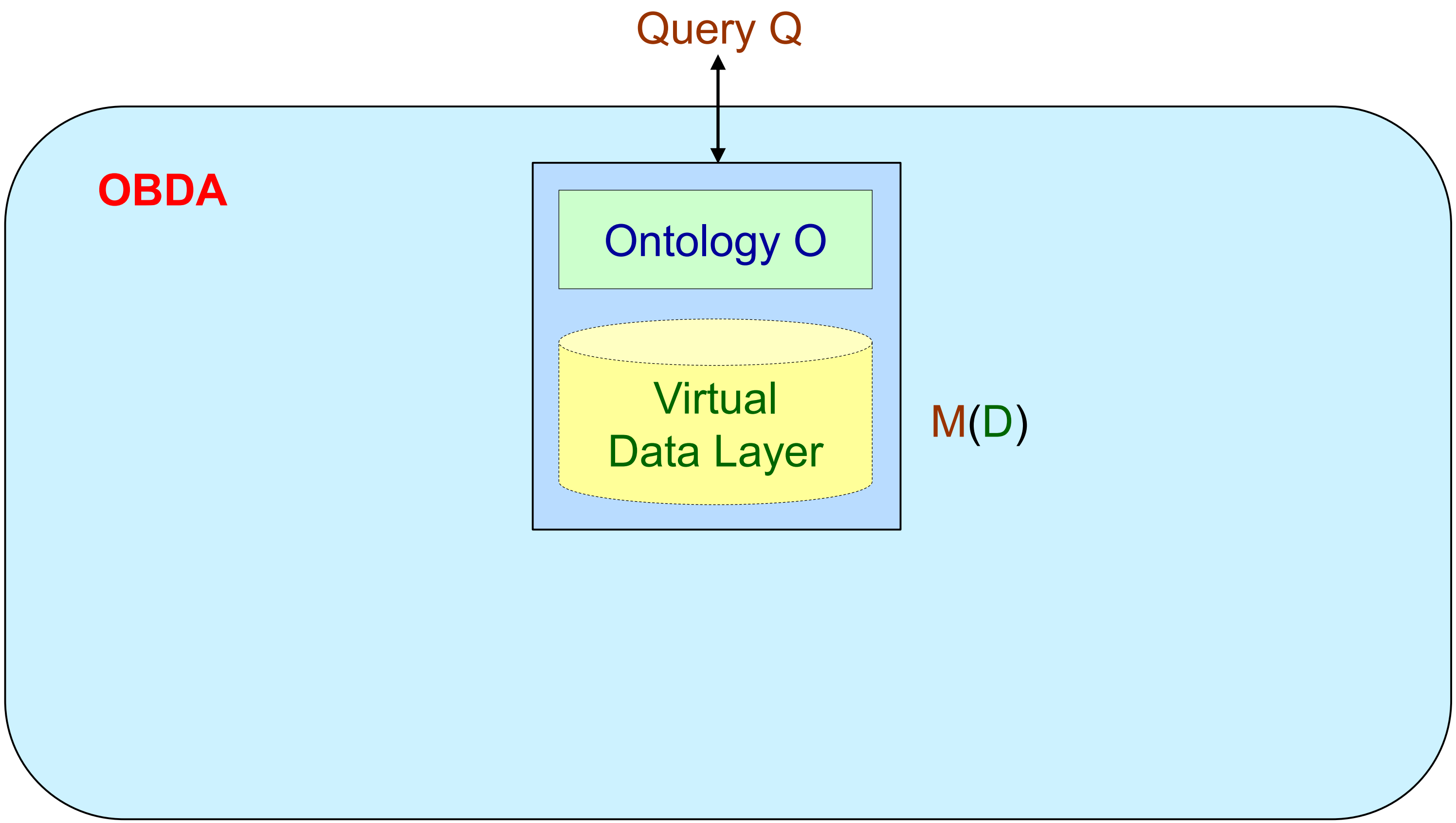
- The sources and the mapping define a virtual data layer M(D)

Query Answering in OBDA



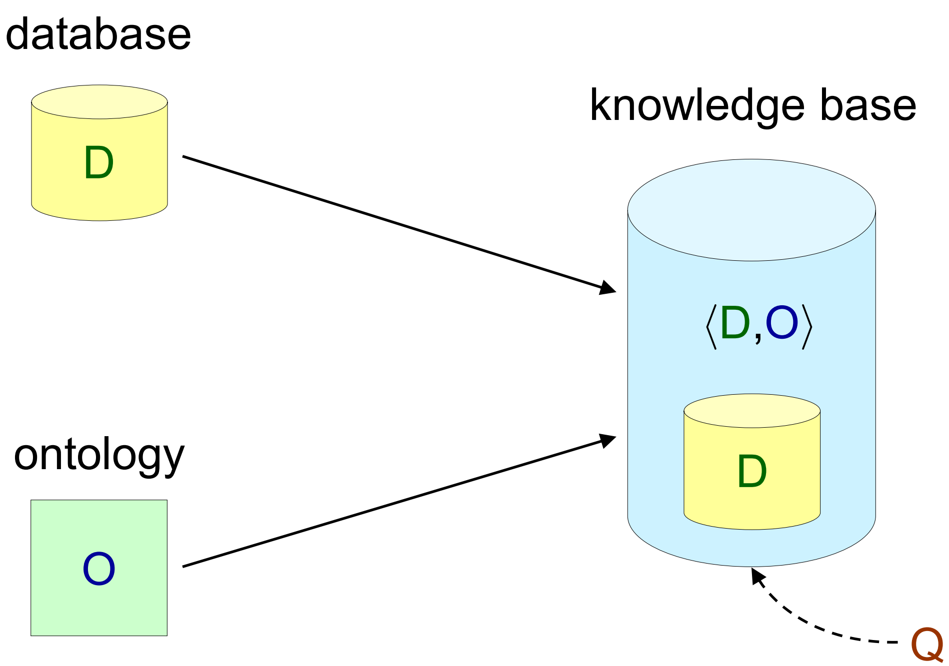
- The sources and the mapping define a virtual data layer M(D)
- Queries are answered against the knowledge base ⟨M(D), O⟩

Query Answering in OBDA



Ontology-Based Query Answering

Ontology-based Query Answering (OBQA)



$$\text{Certain-Answers}(Q, \langle D, O \rangle) \quad = \quad \bigcap_{M \in \text{models}(D \wedge O)} Q(M)$$

(formal definitions later - once we fix the languages)

Issues in Ontology-based Query Answering

What is the right ontology language?

- A wide spectrum of languages that differ in expressive power and computational complexity (e.g., description logics, existential rules)

- Data tractability is a key property to be useful in practice

What is the right query language?

- Well-known database query languages (e.g., conjunctive queries)

Few Words on Description Logics (DLs)

- DLs are well-behaved fragments of first-order logic
- Several DL-based languages exist (from lightweight to very expressive logics)
- Strongly influenced the W3C standard Web Ontology Language OWL

- **Syntax:** We start from a vocabulary with
  - Concept names: atomic classes or unary predicates - Parent, Person
  - Role names: atomic relations or binary predicates - HasParent

  and we build axioms
  - $\text{Person} \sqsubseteq \exists\text{HasParent.Parent}$ - each person has a parent
  - $\text{Parent} \sqsubseteq \text{Person}$ - each parent is a person

- **Semantics:** Standard first-order semantics

DL-Lite Family

DL-Lite: Popular family of DLs - at the basis of the OWL 2 QL profile of OWL 2

DL-Lite Axioms	First-order Representation
$A \sqsubseteq B$	$\forall x\ (A(x) \rightarrow B(x))$
$A \sqsubseteq \exists R$	$\forall x\ (A(x) \rightarrow \exists y\ R(x,y))$
$\exists R \sqsubseteq A$	$\forall x \forall y\ (R(x,y) \rightarrow A(x))$
$\exists R \sqsubseteq \exists P$	$\forall x \forall y\ (R(x,y) \rightarrow \exists z\ P(x,z))$
$A \sqsubseteq \exists R.B$	$\forall x\ (A(x) \rightarrow \exists y\ (R(x,y) \land B(y)))$
$R \sqsubseteq P$	$\forall x \forall y\ (R(x,y) \rightarrow P(x,y))$
$A \sqsubseteq \neg B$	$\forall x\ (A(x) \land B(x) \rightarrow \bot)$

The Description Logic EL

EL: Popular DL for biological applications - at the basis of the OWL 2 EL profile

EL Axioms	First-order Representation
$A \sqsubseteq B$	$\forall x\ (A(x) \rightarrow B(x))$
$A \sqcap B \sqsubseteq C$	$\forall x\ (A(x) \wedge B(x) \rightarrow C(x))$
$A \sqsubseteq \exists R.B$	$\forall x\ (A(x) \rightarrow \exists y\ (R(x,y) \wedge B(y)))$
$\exists R.B \sqsubseteq A$	$\forall x \forall y\ (R(x,y) \wedge B(y) \rightarrow A(x))$

…several other, more expressive, description logics exist

A Simple Example

$$\forall x\ (Researcher(x) \rightarrow \exists y\ (WorksFor(x,y) \wedge Project(y)))$$

$$\forall x\ (Project(x) \rightarrow \exists y\ (WorksFor(y,x) \wedge Researcher(y)))$$

$$\forall x \forall y\ (WorksFor(x,y) \rightarrow Researcher(x) \wedge Project(y))$$

$$\forall x\ (Project(x) \rightarrow \exists y\ (ProjectName(x,y)))$$

Some Terminology

- Our basic vocabulary:
  - A countable set **C** of constants - domain of a database
  - A countable set **N** of (labeled) nulls - globally ∃-quantified variables
  - A countable set **V** of (regular) variables - used in rules and queries
- A term is a constant, null or variable
- An atom has the form $P(t_1,\ldots,t_n)$ - P is an n-ary predicate and t_i's are terms
- An instance is a (possibly infinite) set of atoms with constants and nulls
- A database is a finite instance with only constants

Syntax of Existential Rules

An existential rule is a first-order sentence

$$\forall \mathbf{x} \forall \mathbf{y}\ (\underbrace{\varphi(\mathbf{x},\mathbf{y})}_{\text{body}} \rightarrow \exists \mathbf{z}\ \underbrace{\psi(\mathbf{x},\mathbf{z})}_{\text{head}})$$

- $\mathbf{x}$,$\mathbf{y}$ and $\mathbf{z}$ are tuples of variables of **V**
- $\varphi(\mathbf{x},\mathbf{y})$ and $\psi(\mathbf{x},\mathbf{z})$ are conjunctions of atoms (possibly with constants)

…a.k.a. tuple-generating dependencies and Datalog$^{\pm}$ rules

Homomorphism

- Semantics of existential rules via the key notion of homomorphism

- A substitution from a set of symbols $\mathbf{S}$ to a set of symbols $\mathbf{T}$ is a function $h : \mathbf{S} \rightarrow \mathbf{T}$ - h is a set of mappings of the form $s \mapsto t$, where $s \in \mathbf{S}$ and $t \in \mathbf{T}$

- A homomorphism from a set of atoms $\mathbf{A}$ to a set of atoms $\mathbf{B}$ is a substitution $h : \mathbf{C} \cup \mathbf{N} \cup \mathbf{V} \rightarrow \mathbf{C} \cup \mathbf{N} \cup \mathbf{V}$ such that:

  (i) $t \in \mathbf{C} \;\Rightarrow\; h(t) = t$

  (ii) $P(t_1,\ldots, t_n) \in \mathbf{A} \;\Rightarrow\; h(P(t_1,\ldots, t_n)) = P(h(t_1),\ldots, h(t_n)) \in \mathbf{B}$

- Can be naturally extended to conjunctions of atoms

Semantics of Existential Rules

- An instance J is a model of the existential rule

$$\rho \;=\; \forall \mathbf{x} \forall \mathbf{y}\, (\varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{x},\mathbf{z}))$$

  written as $J \vDash \rho$, if the following holds:

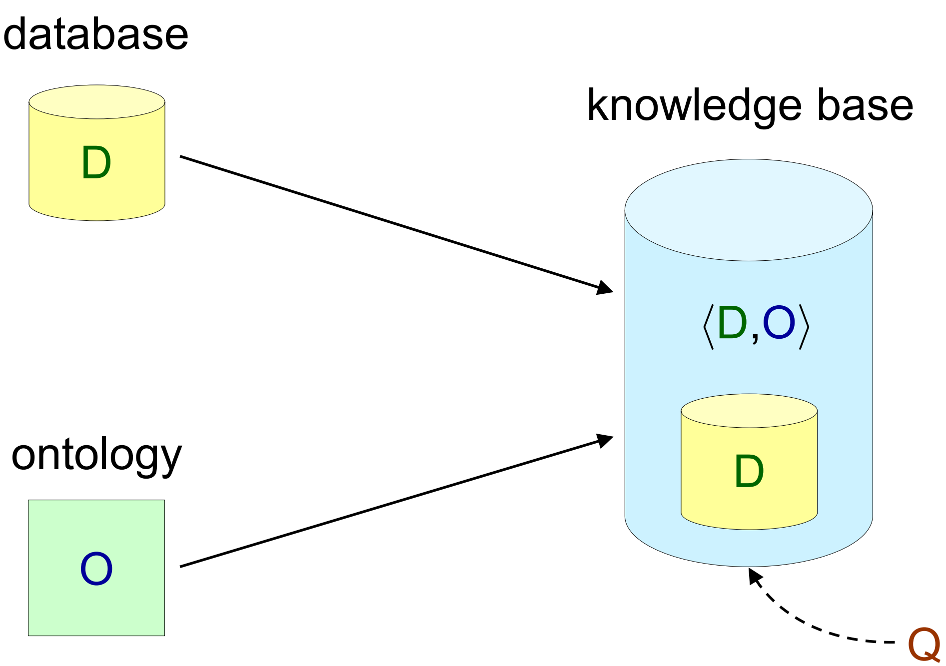
  whenever there exists a homomorphism h such that $h(\varphi(\mathbf{x},\mathbf{y})) \subseteq J$,

  then there exists $g \supseteq h_{|\mathbf{x}}$ such that $g(\psi(\mathbf{x},\mathbf{z})) \subseteq J$

  $\{t \mapsto h(t) \mid t \in \mathbf{x}\}$ - the restriction of h to **x**

- Given a set O of existential rules, J is a model of O, written as $J \vDash O$, if the following holds: for each $\rho \in O$, $J \vDash \rho$

Ontology-Based Query Answering (OBQA)

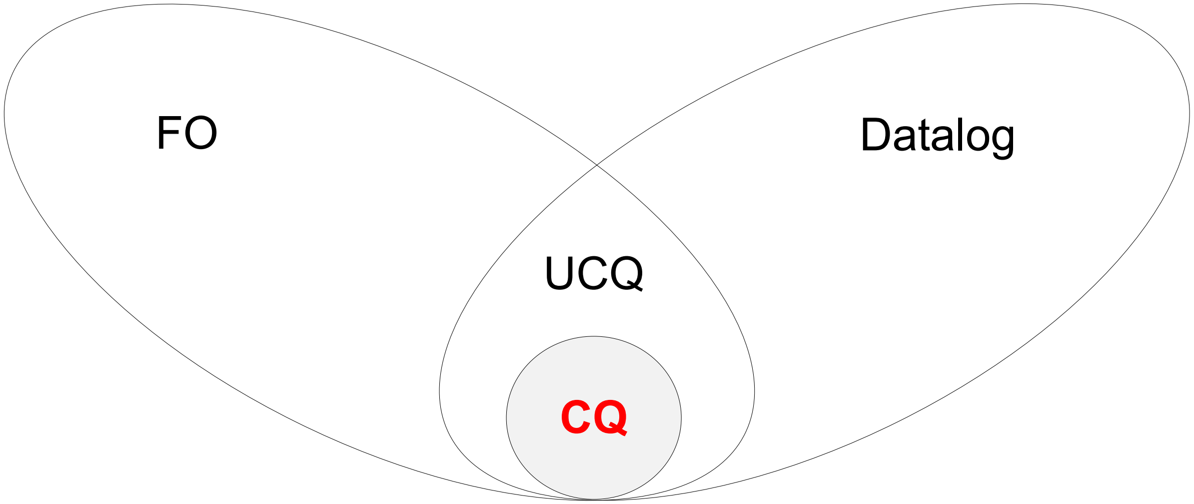


existential / Datalog$^{\pm}$ rules

$\forall \mathbf{x} \forall \mathbf{y}\ (\varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z}))$

Query Languages

- The four most important query languages
  - **Conjunctive Queries (CQ)**
  - Unions of Conjunctive Queries (UCQ)
  - First-order Queries (FO)
  - Datalog



Syntax of Conjunctive Queries

A conjunctive query (CQ) is an expression

$$\exists \mathbf{y}\ (\varphi(\mathbf{x},\mathbf{y})) \quad \text{or} \quad \mathrm{Ans}(\mathbf{x}) \leftarrow \varphi(\mathbf{x},\mathbf{y})$$

- $\mathbf{x}$ and $\mathbf{y}$ are tuples of variables of **V**
- $\varphi(\mathbf{x},\mathbf{y})$ is a conjunction of atoms (possibly with constants)

The most important query language used in practice

Forms the SELECT-FROM-WHERE fragment of SQL

Semantics of Conjunctive Queries

- A match of a CQ $\exists \mathbf{y}\ (\varphi(\mathbf{x},\mathbf{y}))$ in an instance J is a homomorphism h such that $h(\varphi(\mathbf{x},\mathbf{y})) \subseteq J$ - all the atoms of the query are satisfied

- The answer to $Q(\mathbf{x}) = \exists \mathbf{y}\ (\varphi(\mathbf{x},\mathbf{y}))$ over J is the set of tuples

$$Q(J) = \{h(\mathbf{x}) \in \mathbf{C} \mid h \text{ is a match of } Q \text{ in } J\}$$

- The answer consists of the witnesses for the free variables of the query

Conjunctive Queries: Example

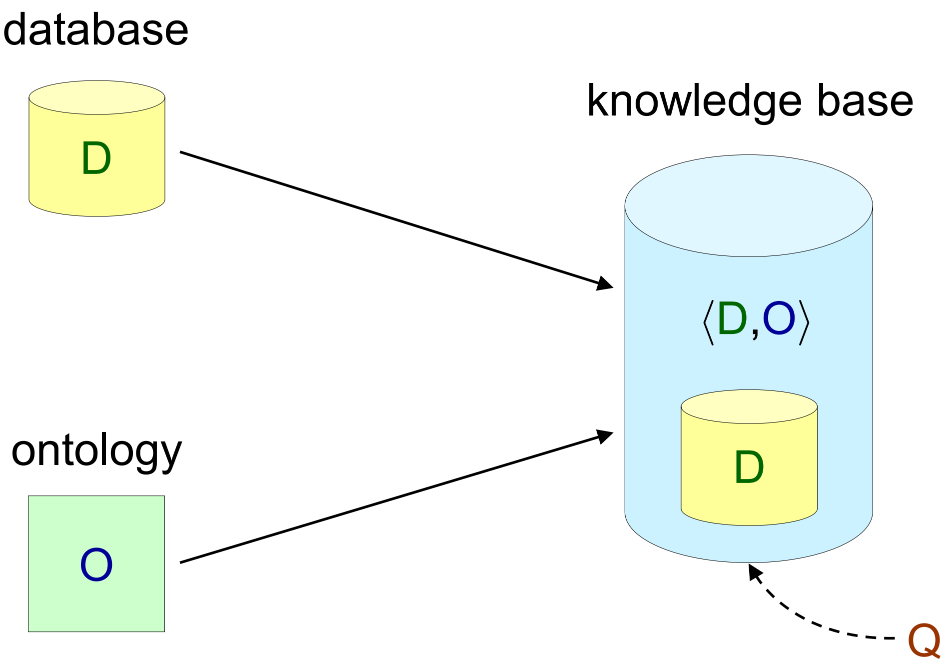
Find the researchers who work for the “VADA” project

Researcher(id), Project(id), WorksFor(rid, pid), ProjectName(pid, name)

$\exists y\,(\text{Researcher}(x) \wedge \text{WorksFor}(x,y) \wedge \text{Project}(y) \wedge \text{ProjectName}(y,\text{“VADA”}))$

```
SELECT  R.id
FROM  Researcher R,  WorksFor W,  Project P,  ProjectName N
WHERE   R.id = W.rid  AND
        W.pid = P.id  AND
        P.id = N.pid  AND
        N.name = “VADA”
```

Ontology-based Query Answering (OBQA)



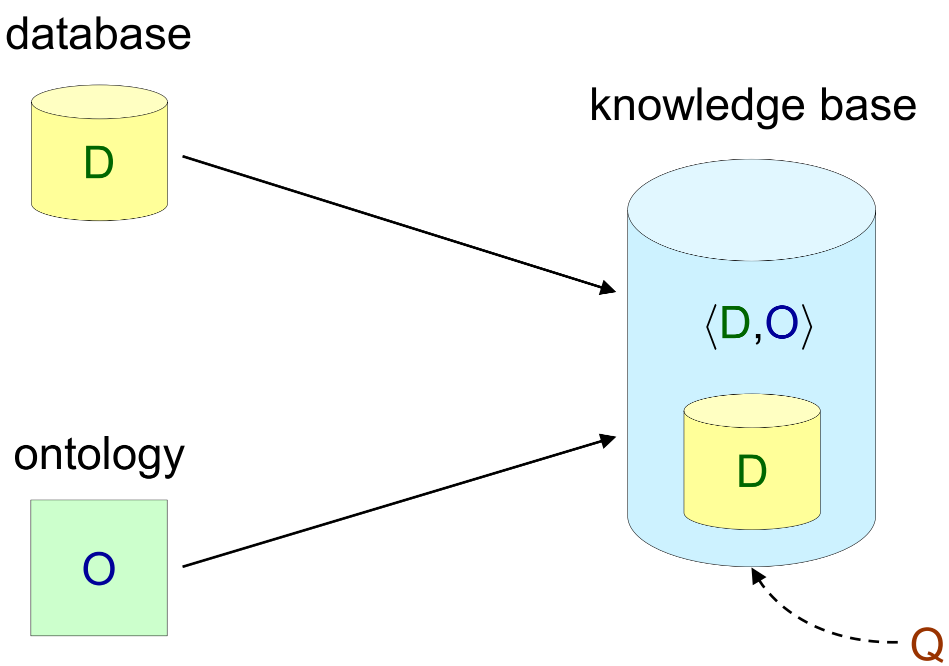
existential / Datalog$^{\pm}$ rules

$$\forall \mathbf{x} \forall \mathbf{y}\ (\varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z}))$$

conjunctive queries

$$\exists \mathbf{y}\ (\varphi(\mathbf{x},\mathbf{y}))$$

Ontology-based Query Answering (OBQA)



$$\text{Certain-Answers}(Q, \langle D, O \rangle) \quad = \quad \bigcap_{M \in \text{models}(D \wedge O)} Q(M)$$

$\{J \mid J \supseteq D \text{ and } J \vDash O\}$

OBQA: Formal Definition

an ontology language based on existential rules

OBQA(**L**)

Input: database D, ontology O $\in$ **L**, CQ $Q(\mathbf{x}) = \exists \mathbf{y}\,(\varphi(\mathbf{x},\mathbf{y}))$, tuple **t** $\in$ adom(D)$^{|\mathbf{x}|}$

Question: **t** $\in$ Certain-Answers(Q,$\langle$D,O$\rangle$) $= \bigcap_{M \in \text{models}(D \wedge O)} Q(M)$?

active domain - constants occurring in D

OBQA: Complexity Metrics

- Combined complexity - everything is part of the input

- Data complexity - only D and **t** are part of the input

OBQA[O,Q]

Input: database D, tuple $\mathbf{t} \in \text{adom}(D)^{|\mathbf{x}|}$

Question: $\mathbf{t} \in$ Certain-Answers(Q,$\langle$D,O$\rangle$)?

OBQA(**L**) is $\mathcal{C}$-complete in data complexity if:

1. For every O $\in$ **L** and CQ Q, OBQA[O,Q] is in $\mathcal{C}$

2. There exists O $\in$ **L** and CQ Q such that OBQA[O,Q] is $\mathcal{C}$-hard

OBQA: The Boolean Case

OBQA($\mathbf{L}$)

Input: database D, ontology $O \in \mathbf{L}$, CQ $Q(\mathbf{x}) = \exists \mathbf{y}\ (\varphi(\mathbf{x},\mathbf{y}))$, tuple $\mathbf{t} \in adom(D)^{|\mathbf{x}|}$

Question: $\mathbf{t} \in$ Certain-Answers$(Q,\langle D,O\rangle) = \bigcap_{M \in models(D \wedge O)} Q(M)$?

$$\mathbf{t} \in \text{Certain-Answers}(Q,\langle D\ O\rangle) \Leftrightarrow \forall M \in models(D \wedge O),\ M \models \exists \mathbf{y}\ (\varphi(\mathbf{t},\mathbf{y}))$$

$$\Leftrightarrow D \wedge O \models \exists \mathbf{y}\ (\varphi(\mathbf{t},\mathbf{y}))$$

Boolean CQ - no free variables

OBQA: The Boolean Case

OBQA(**L**)

Input: database D, ontology O $\in$ **L**, CQ $Q(\mathbf{x}) = \exists \mathbf{y}\, (\varphi(\mathbf{x},\mathbf{y}))$, tuple $\mathbf{t} \in \mathrm{adom}(D)^{|\mathbf{X}|}$

Question: $\mathbf{t} \in$ Certain-Answers$(Q,\langle D,O\rangle) = \bigcap_{M \in \mathrm{models}(D \wedge O)} Q(M)$?

For understanding the complexity of OBQA(**L**), it suffices to focus on Boolean CQs

OBQA(**L**)

Input: database D, ontology O $\in$ **L**, Boolean CQ Q

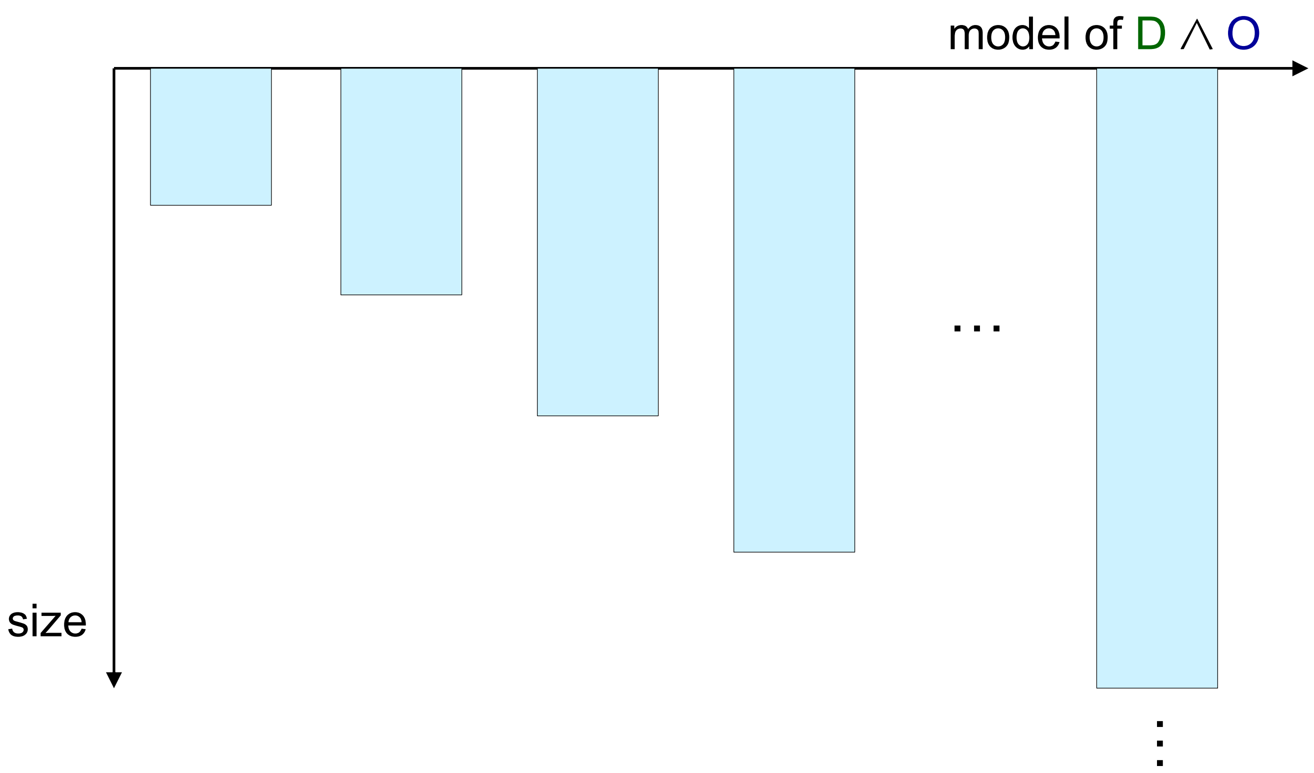
Question: $D \wedge O \models Q$?

Why is OBQA technically challenging?

What is the right tool for tackling this problem?

The Two Dimensions of Infinity

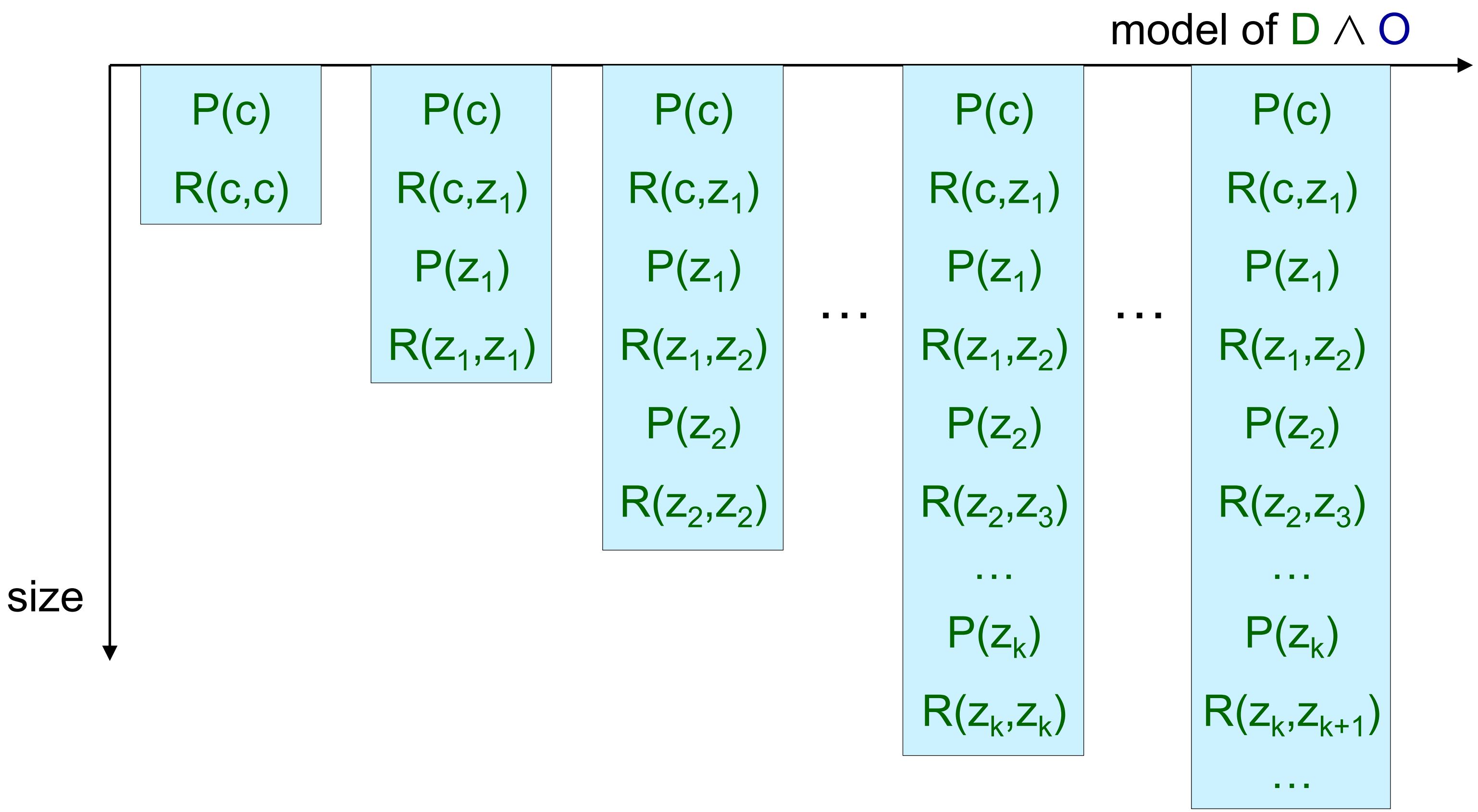
Consider the database D, and the ontology O



D ∧ O admits infinitely many models, of possibly infinite size

The Two Dimensions of Infinity

D = {P(c)} O = $\{\forall x\, (P(x) \rightarrow \exists y\, (R(x,y) \wedge P(y)))\}$



$z_1, z_2, z_3, \ldots$ are nulls of **N**

Taming the First Dimension of Infinity

$D = \{P(c)\}$ $\quad\quad$ $O = \{\forall x\, (P(x) \rightarrow \exists y\, (R(x,y) \wedge P(y)))\}$

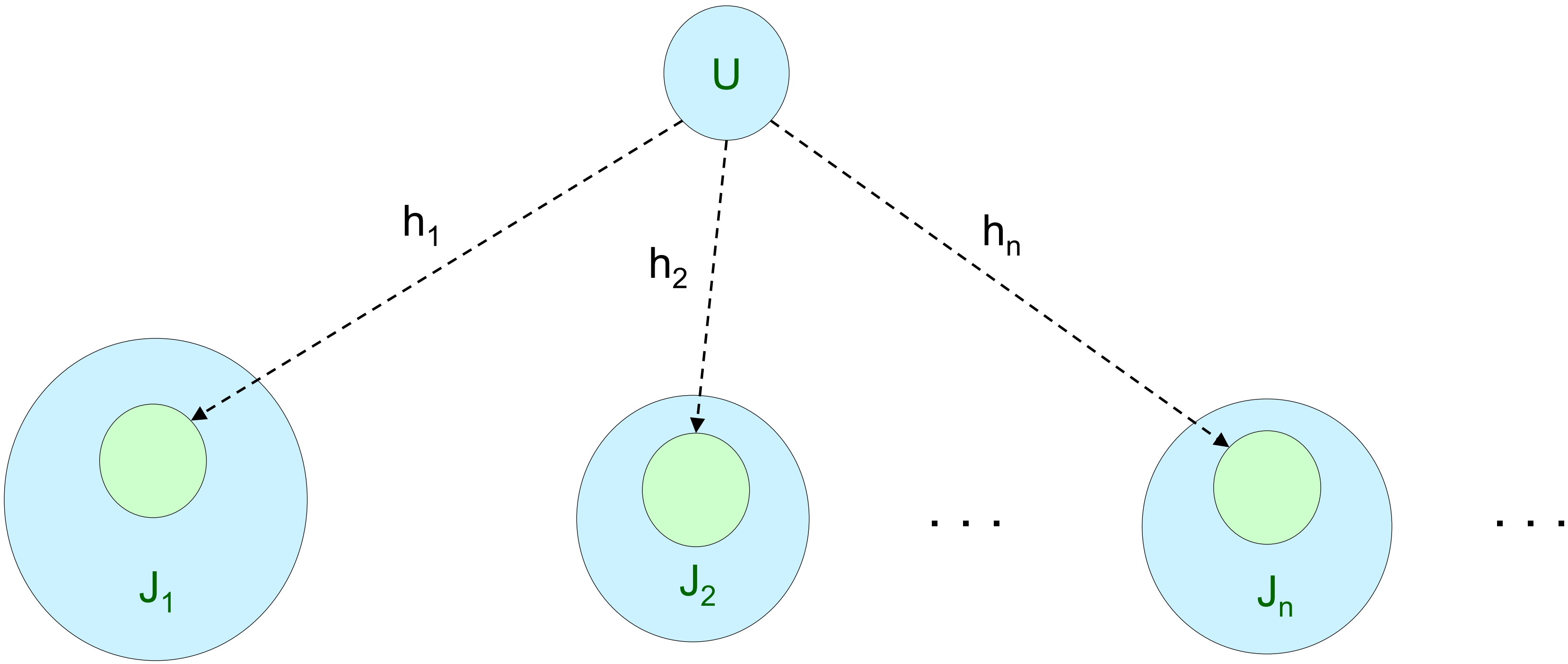
model of $D \wedge O$

size

$P(c)$	$P(c)$	$P(c)$	…	$P(c)$	…	$P(c)$
$R(c,c)$	$R(c,z_1)$	$R(c,z_1)$		$R(c,z_1)$		$R(c,z_1)$
✗	$P(z_1)$	$P(z_1)$		$P(z_1)$		$P(z_1)$
	$R(z_1,z_1)$	$R(z_1,z_2)$		$R(z_1,z_2)$		$R(z_1,z_2)$
	✗	$P(z_2)$		$P(z_2)$		$P(z_2)$
		$R(z_2,z_2)$		$R(z_2,z_3)$		$R(z_2,z_3)$
		✗		…		…
				$P(z_k)$		$P(z_k)$
				$R(z_k,z_k)$		$R(z_k,z_{k+1})$
				✗		…
						✓

Key Idea: Focus on a representative, a model that is as general as possible

Universal Models (a.k.a. Canonical Models)



An instance U is a universal model of $D \wedge O$ if the following holds:

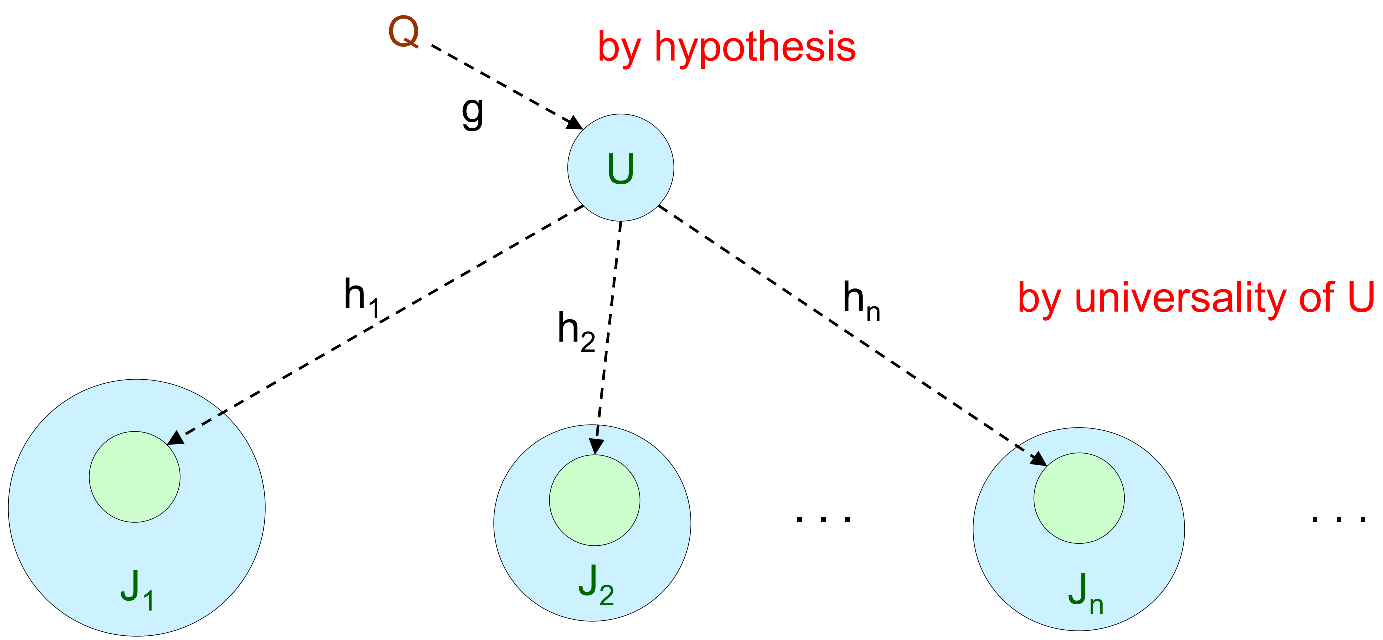
1. U is a model of $D \wedge O$
2. $\forall J \in$ models($D \wedge O$), there exists a homomorphism h_J such that $h_J(U) \subseteq J$

Query Answering via Universal Models

Theorem: $D \land O \models Q$ iff $U \models Q$, where U is a universal model of $D \land O$

Proof: ($\Rightarrow$) Trivial since, for every $J \in \text{models}(D \land O)$, $J \models Q$

($\Leftarrow$) By exploiting the universality of U



$\forall J \in \text{models}(D \land O), \exists h_J$ such that $h_J(g(Q)) \subseteq J \quad \Rightarrow \quad \forall J \in \text{models}(D \land O), J \models Q$

$\Rightarrow \quad D \land O \models Q$

The Chase Procedure

- Fundamental algorithmic tool used in databases

- It has been applied to a wide range of problems:
  - Checking containment of queries under constraints
  - Computing data exchange solutions
  - Computing certain answers in data integration settings
  - …

… what's the reason for the ubiquity of the chase in databases?

it constructs universal models

The Chase Procedure

D

Person(John)

O

$\forall x\ (\mathrm{Person}(x) \rightarrow \exists y\ (\mathrm{HasParent}(x,y) \wedge \mathrm{Person}(y)))$

chase(D,O) = D $\cup$

The Chase Procedure

D

Person(John)

O

$\forall x\,(Person(x) \rightarrow \exists y\,(HasParent(x,y) \wedge Person(y)))$

chase(D,O) = D $\cup$ {HasParent(John, z_1), Person(z_1)

The Chase Procedure

D

Person(John)

O

$\forall x\ (Person(x) \rightarrow \exists y\ (HasParent(x,y) \wedge Person(y)))$

$chase(D,O) = D \cup \{HasParent(John,\ z_1),\ Person(z_1),$

$HasParent(z_1,\ z_2),\ Person(z_2)$

The Chase Procedure

D

Person(John)

O

$\forall x\, (Person(x) \rightarrow \exists y\, (HasParent(x,y) \wedge Person(y)))$

$$chase(D,O) = D \cup \{HasParent(John, z_1), Person(z_1),$$
$$HasParent(z_1, z_2), Person(z_2),$$
$$HasParent(z_2, z_3), Person(z_3)$$

The Chase Procedure

D

Person(John)

O

$\forall x\,(\mathrm{Person}(x) \rightarrow \exists y\,(\mathrm{HasParent}(x,y) \wedge \mathrm{Person}(y)))$

$\mathrm{chase}(D,O) = D \cup \{\mathrm{HasParent}(\mathrm{John}, z_1), \mathrm{Person}(z_1),$

$\mathrm{HasParent}(z_1, z_2), \mathrm{Person}(z_2),$

$\mathrm{HasParent}(z_2, z_3), \mathrm{Person}(z_3), \ldots$

infinite instance

The Chase Procedure: Formal Definition

- Chase rule - the building block of the chase procedure

- A rule $\rho = \forall \mathbf{x} \forall \mathbf{y}\, (\varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{x},\mathbf{z}))$ is applicable to instance J if:

    1. There exists a homomorphism h such that $h(\varphi(\mathbf{x},\mathbf{y})) \subseteq J$
    2. There is no $g \supseteq h_{|\mathbf{x}}$ such that $g(\psi(\mathbf{x},\mathbf{z})) \subseteq J$

$J = \{R(a), P(a,b)\}$

$h = \{x \mapsto a\}$ $\quad$ $g = \{x \mapsto a, y \mapsto b\}$

$\forall x\, (R(x) \rightarrow \exists y\, P(x,y))$

✗

$J = \{R(a), P(b,a)\}$

$h = \{x \mapsto a\}$ $\quad$ no extension

$\forall x\, (R(x) \rightarrow \exists y\, P(x,y))$

✓

The Chase Procedure: Formal Definition

- Chase rule - the building block of the chase procedure

- A rule $\rho = \forall \mathbf{x} \forall \mathbf{y}\ (\varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z}))$ is applicable to instance J if:
  1. There exists a homomorphism h such that $h(\varphi(\mathbf{x},\mathbf{y})) \subseteq J$
  2. There is no $g \supseteq h_{|\mathbf{x}}$ such that $g(\psi(\mathbf{x},\mathbf{z})) \subseteq J$

- Let $J_+ = J \cup \{g(\psi(\mathbf{x},\mathbf{z}))\}$, where $g \supseteq h_{|\mathbf{x}}$ and $g(\mathbf{z})$ are "fresh" nulls not in J

- The result of applying ρ to J is J_+, denoted $J\langle\rho,h\rangle J_+$ - single chase step

The Chase Procedure: Formal Definition

- A finite chase of D w.r.t. O is a finite sequence

$$D\langle\rho_1,h_1\rangle J_1\langle\rho_2,h_2\rangle J_2\langle\rho_3,h_3\rangle J_3 \ \dots\ \langle\rho_n,h_n\rangle J_n$$

and chase(D,O) is defined as the instance J_n

all applicable rules will eventually be applied

- An infinite chase of D w.r.t. O is a fair infinite sequence

$$D\langle\rho_1,h_1\rangle J_1\langle\rho_2,h_2\rangle J_2\langle\rho_3,h_3\rangle J_3 \ \dots\ \langle\rho_n,h_n\rangle J_n \dots$$

and chase(D,O) is defined as the instance $\cup_{k \geq 0} J_k$ (with $J_0 = D$)

least fixpoint of a monotonic operator - the chase step

Chase: A Universal Model

Theorem: chase(D,O) is a universal model of $D \wedge O$

Proof (sketch):

- By construction, chase(D,O) $\in$ models($D \wedge O$)
- It remains to show that chase(D,O) can be homomorphically embedded into every other model of $D \wedge O$
- Fix an arbitrary instance $J \in$ models($D \wedge O$). We need to show that there exists h such that h(chase(D,O)) $\subseteq$ J
- By induction on the number of applications of the chase step, we show that for every $k \geq 0$, there exists h_k such that $h_k(\text{chase}^{[k]}(D,O)) \subseteq J$, and h_k is compatible with h_{k-1}
- Clearly, $\cup_{k \geq 0} h_k$ is a well-defined homomorphism that maps chase(D,O) to J
- The claim follows with $h = \cup_{k \geq 0} h_k$

the result of the chase after k applications of the chase step

Chase: Uniqueness Property

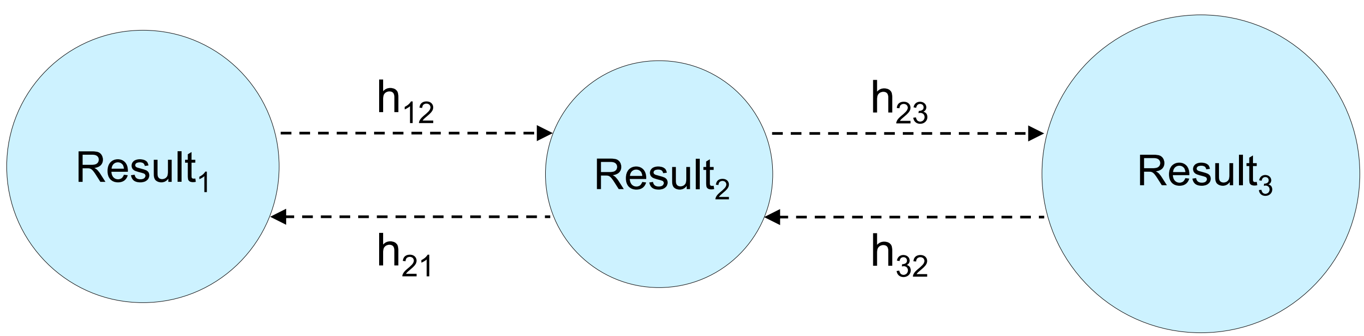
- The result of the chase is not unique - depends on the order of rule application

$D = \{P(a)\}$ $\rho_1 = \forall x\, (P(x) \rightarrow \exists y\, R(y))$ $\rho_2 = \forall x\, (P(x) \rightarrow R(x))$

$Result_1 = \{P(a), R(z), R(a)\}$ ρ_1 then ρ_2

$Result_2 = \{P(a), R(a)\}$ ρ_2 then ρ_1

- But, it is unique up to homomorphic equivalence



$\Rightarrow$ it is unique for query answering purposes

Query Answering via the Chase

Theorem: $D \wedge O \models Q$ iff $U \models Q$, where U is a universal model of $D \wedge O$

&

Theorem: chase(D,O) is a universal model of $D \wedge O$

$\Downarrow$

Corollary: $D \wedge O \models Q$ iff chase(D,O) $\models Q$

We can tame the first dimension of infinity by exploiting the chase procedure

Can we tame the second dimension of infinity?

Undecidability of OBQA

arbitrary existential rules

Theorem: OBQA(**∃RULES**) is undecidable

Proof Idea : By simulating a deterministic Turing machine with an empty tape.

Gaining Decidability

By restricting the database

- $\{\text{Start}(c)\} \wedge O \models Q$ iff the Turing Machine T accepts
- The problem is undecidable already for singleton databases

By restricting the query language

- $D \wedge O \models \exists x \, \text{Accept}(x)$ iff the Turing Machine T accepts
- The problem is undecidable already for atomic queries

By restricting the ontology language

- Achieve a good trade-off between expressive power and complexity
- Field of intense research (Calabria, Dresden, Edinburgh, Montpellier, Oxford, Vienna)

Datalog$^{\pm}$ Nomenclature

- Extend Datalog by allowing in the head:

    - Existential quantification ($\exists$)
    - Equality atoms ($=$)
    - Constant false ($\perp$)

Datalog[$\exists$,$=$,$\perp$]

a highly expressive ontology language

Datalog$^{\pm}$ Nomenclature

- Extend Datalog by allowing in the head:
  - Existential quantification ($\exists$)
  - Equality atoms (=)
  - Constant false ($\perp$)

  Datalog[$\exists$,=,$\perp$]

- But, already Datalog[$\exists$] is undecidable

- Datalog[$\exists$,=,$\perp$] is syntactically restricted $\rightarrow$ **Datalog$^{\pm}$**

Gaining Decidability

By restricting the database

- {Start(c)} $\wedge$ O $\models$ Q iff the DTM M accepts
- The problem is undecidable already for singleton databases

By restricting the query language

- D $\wedge$ O $\models$ $\exists$x Accept(x) iff the DTM M accepts
- The problem is undecidable already for atomic queries

By restricting the ontology language

- Achieve a good trade-off between expressive power and complexity
- Field of intense research (Calabria, Dresden, Edinburgh, Montpellier, Oxford, Vienna)

What is the Source of Non-termination?

D

Person(John)

O

$\forall x\,(Person(x) \rightarrow \exists y\,(HasParent(x,y) \wedge Person(y)))$

$$chase(D,O) = D \cup \{HasParent(John, z_1), Person(z_1), HasParent(z_1, z_2), Person(z_2), HasParent(z_2, z_3), Person(z_3), \ldots$$

1. **Existential quantification**
2. **Recursive definitions**

Termination of the Chase

- Drop existential quantification
  - We obtain the class of **full** existential rules
  - Very close to Datalog

- Drop recursive definitions
  - We obtain the class of **acyclic** existential rules
  - A.k.a. non-recursive existential rules

Recall our Example

D

Person(John)

O

$\forall x\,(\text{Person}(x) \rightarrow \exists y\,(\text{HasParent}(x,y) \wedge \text{Person}(y)))$

$$\text{chase}(D,O) = D \cup \{\text{HasParent}(\text{John}, z_1), \text{Person}(z_1),$$

$$\text{HasParent}(z_1, z_2), \text{Person}(z_2),$$

$$\text{HasParent}(z_2, z_3), \text{Person}(z_3), \ldots$$

The above rule can be written as the DL-Lite axiom

$$\text{Person} \sqsubseteq \exists\text{HasParent.Person}$$

Recall our Example

D

Person(John)

O

$\forall x\, (Person(x) \rightarrow \exists y\, (HasParent(x,y) \wedge Person(y)))$

$chase(D,O) = D \cup \{HasParent(John, z_1), Person(z_1),$

$HasParent(z_1, z_2), Person(z_2),$

$HasParent(z_2, z_3), Person(z_3), \ldots$

Existential quantification & recursive definitions

are key features for modelling ontologies

Research Challenge

We need classes of existential rules such that:

1. Existential quantification and recursive definitions coexist
2. OBQA is decidable, and tractable in the data complexity

$$\Downarrow$$

Tame the infinite chase:

Deal with infinite instances without explicitly building them

Linear Existential Rules

- A linear existential rule is an existential rule of the form

$$\forall \mathbf{x} \forall \mathbf{y}\, (P(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\, \psi(\mathbf{x},\mathbf{z}))$$

single atom

- We denote **LINEAR** the ontology language based on linear existential rules

- A local property - we can inspect one rule at a time

  $\Rightarrow$ given O, we can decide in linear time whether O $\in$ **LINEAR**

  $\Rightarrow$ closed under union

- But, is this a reasonable ontology language?

LINEAR vs. DL-Lite

DL-Lite: Popular family of DLs - at the basis of the OWL 2 QL profile of OWL 2

DL-Lite Axioms	First-order Representation
$A \sqsubseteq B$	$\forall x\ (A(x) \rightarrow B(x))$
$A \sqsubseteq \exists R$	$\forall x\ (A(x) \rightarrow \exists y\ R(x,y))$
$\exists R \sqsubseteq A$	$\forall x \forall y\ (R(x,y) \rightarrow A(x))$
$\exists R \sqsubseteq \exists P$	$\forall x \forall y\ (R(x,y) \rightarrow \exists z\ P(x,z))$
$A \sqsubseteq \exists R.B$	$\forall x\ (A(x) \rightarrow \exists y\ (R(x,y) \wedge B(y)))$
$R \sqsubseteq P$	$\forall x \forall y\ (R(x,y) \rightarrow P(x,y))$
$A \sqsubseteq \neg B$	$\forall x\ (A(x) \wedge B(x) \rightarrow \bot)$

Linear Existential Rules

- A linear existential rule is an existential rule of the form

$$\forall \mathbf{x} \forall \mathbf{y}\ (P(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z}))$$

single atom

- We denote **LINEAR** the ontology language based on linear existential rules

- A local property - we can inspect one rule at a time

    $\Rightarrow$ given O, we can decide in linear time whether O $\in$ **LINEAR**

    $\Rightarrow$ closed under union

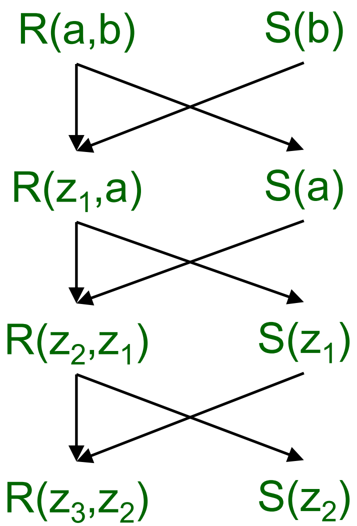
- But, is this a reasonable ontology language? **OWL 2 QL**

Chase Graph

The chase can be naturally seen as a graph - chase graph

$D = \{R(a,b), S(b)\}$

$$O = \begin{cases} \forall x \forall y\, (R(x,y) \wedge S(y) \rightarrow \exists z\, R(z,x)) \\ \forall x \forall y\, (R(x,y) \rightarrow S(x)) \end{cases}$$



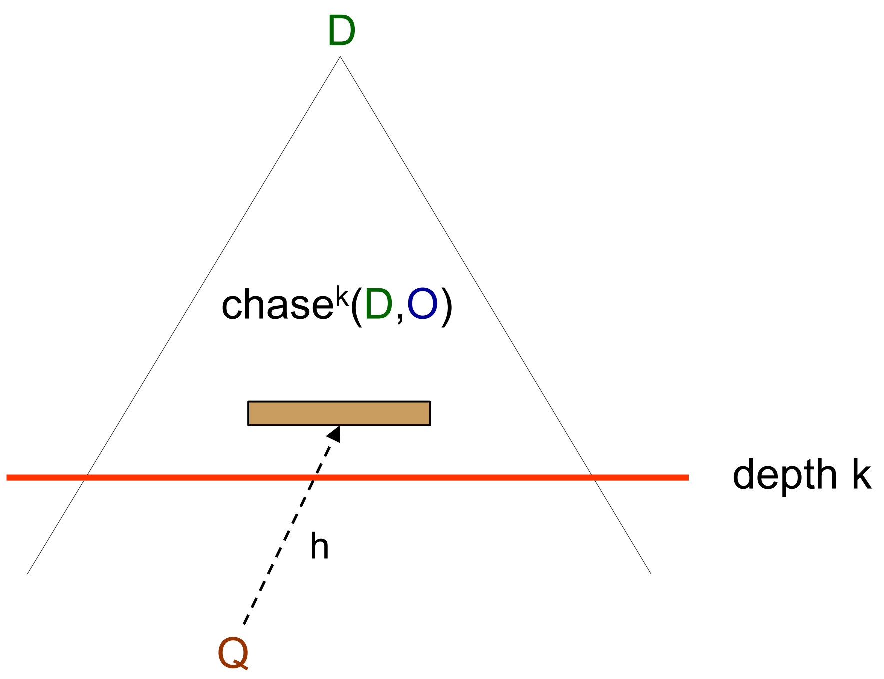
For **LINEAR** the chase graph is a forest

Bounded Derivation-depth Property (BDDP)

Definition: An ontology language **L** enjoys the BDDP if:

for every ontology O $\in$ **L** and CQ Q, there exists k ≥ 0 such that,

for every database D, chase(D,O) $\models$ Q $\Rightarrow$ chase^k(D,O) $\models$ Q



Bounded Derivation-depth Property (BDDP)

Definition: An ontology language **L** enjoys the BDDP if:

for every ontology $O \in$ **L** and CQ Q, there exists $k \geq 0$ such that,

for every database D, $\text{chase}(D,O) \models Q \;\Rightarrow\; \text{chase}^k(D,O) \models Q$

For **LINEAR**, $k = |Q| \cdot m$

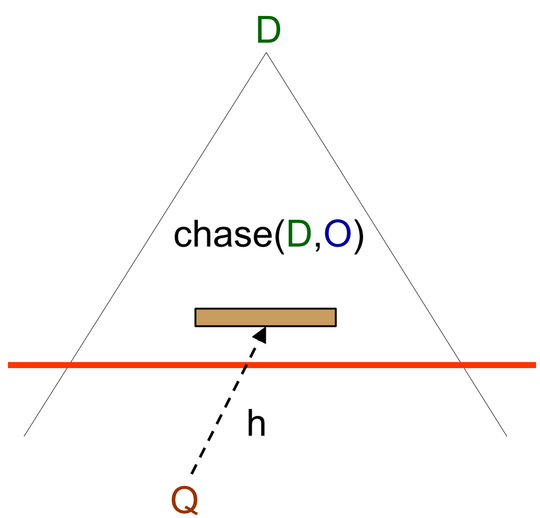
with $m = |\text{sch}(O)| \cdot (2 \cdot \text{maxarity})^{\text{maxarity}}$

predicates occurring in O

The Blocking Algorithm for **LINEAR**

The blocking algorithm shows that OBQA(**LINEAR**) is

- in 2EXPTIME in combined complexity
- in PTIME in data complexity



$k = |Q| \cdot |sch(O)| \cdot (2 \cdot maxarity)^{maxarity}$

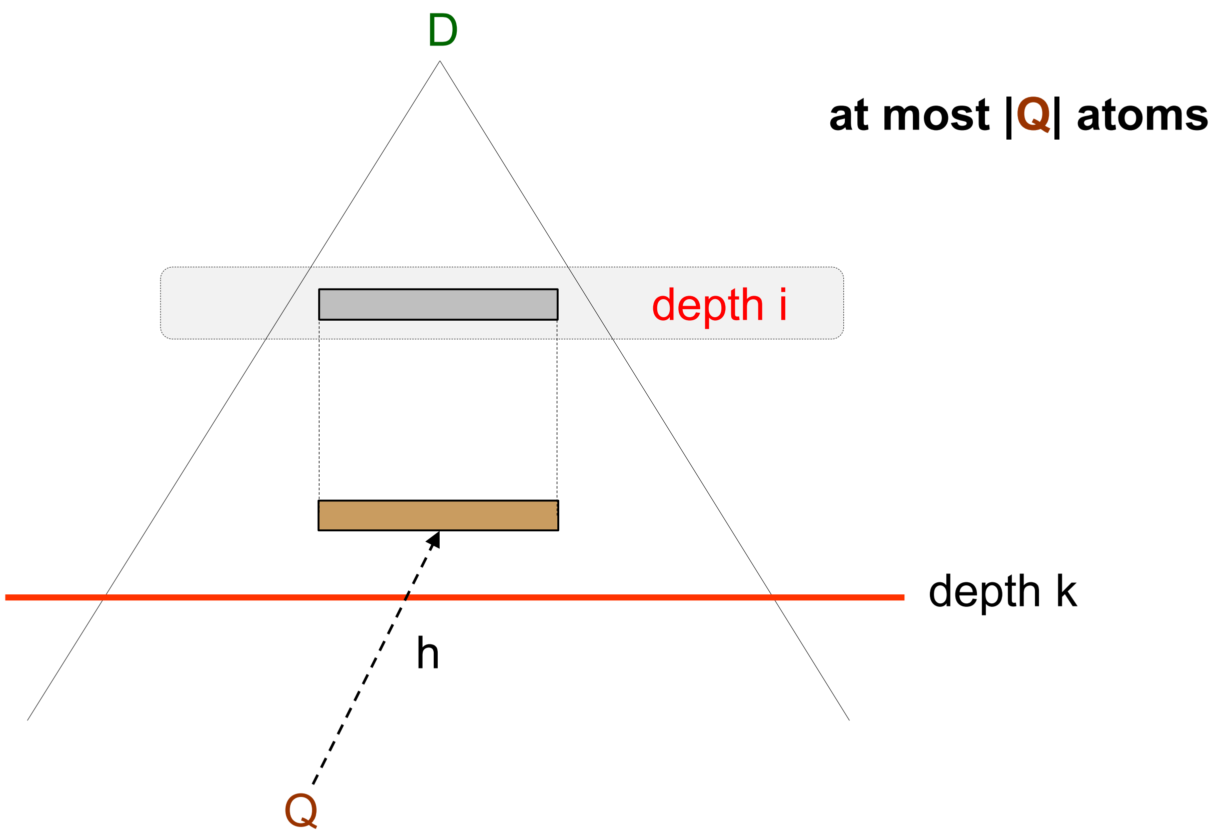
Complexity of OBQA(**LINEAR**)

…but, we can do better than the blocking algorithm

Theorem: OBQA(**LINEAR**) is

- PSPACE-complete in combined complexity
- in LOGSPACE in data complexity

Key Observation



non-deterministic, level-by-level construction

Combined Complexity of **LINEAR**

Theorem: OBQA(**LINEAR**) is in PSPACE

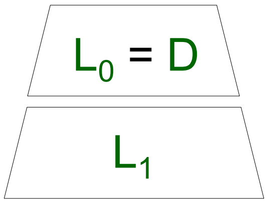
Proof (high-level idea):

$L_0 = D$

Combined Complexity of **LINEAR**

Theorem: OBQA(**LINEAR**) is in PSPACE

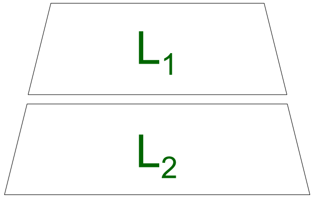
Proof (high-level idea):



Combined Complexity of **LINEAR**

Theorem: OBQA(**LINEAR**) is in PSPACE

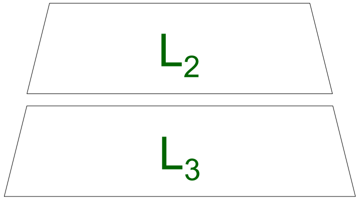
Proof (high-level idea):



Combined Complexity of **LINEAR**

Theorem: OBQA(**LINEAR**) is in PSPACE

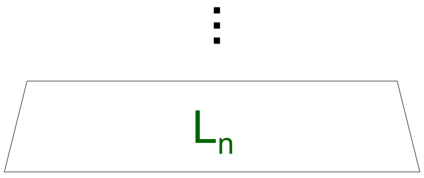
Proof (high-level idea):



Combined Complexity of **LINEAR**

Theorem: OBQA(**LINEAR**) is in PSPACE

Proof (high-level idea):



Combined Complexity of **LINEAR**

Theorem: OBQA(**LINEAR**) is in PSPACE

Proof (high-level idea):

- At each step we need to maintain
    - $\mathcal{O}(|Q|)$ atoms
    - A counter ctr $\leq |Q|^2 \cdot |\text{sch}(O)| \cdot (2 \cdot \text{maxarity})^{\text{maxarity}}$
- Thus, we need polynomial space
- The claim follows since NPSPACE = PSPACE

Combined Complexity of **LINEAR**

We cannot do better than the previous algorithm

Theorem: OBQA(**LINEAR**) is PSPACE-hard

Proof Idea : By simulating a deterministic polynomial space Turing machine

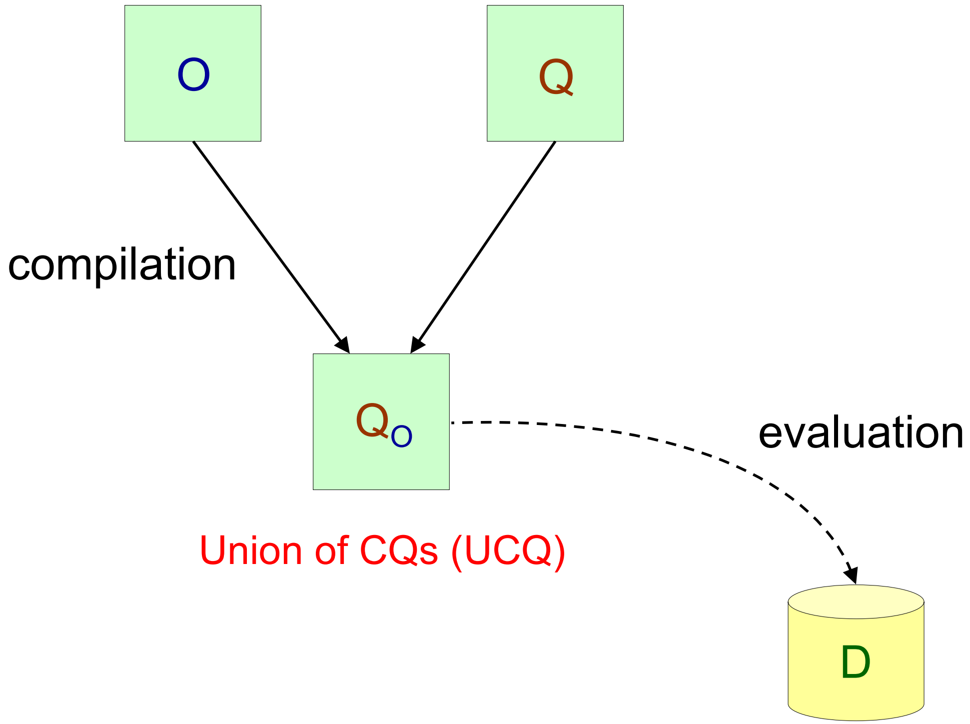
Complexity of OBQA(**LINEAR**)

Theorem: OBQA(**LINEAR**) is

✓ • PSPACE-complete in combined complexity

• in LOGSPACE in data complexity

Query Rewriting



$$\forall D \ : \ D \land O \models Q \quad \Leftrightarrow \quad D \models Q_O$$

Query Rewriting

Theorem: OBQA(**L**) is UCQ-rewritable

$\Rightarrow$ OBQA(**L**) is in LOGSPACE in data complexity

Proof: Fix O $\in$ **L** and CQ Q. We need to show that OBQA[O,Q] is in LOGSPACE:

1. Construct Q_O in $\mathcal{O}(1)$ time (due to UCQ rewritability)
2. Check whether D $\models$ Q_O in LOGSPACE (classical result)

Complexity of OBQA(**LINEAR**)

Theorem: OBQA(**LINEAR**) is

✓ • PSPACE-complete in combined complexity

? • in LOGSPACE in data complexity

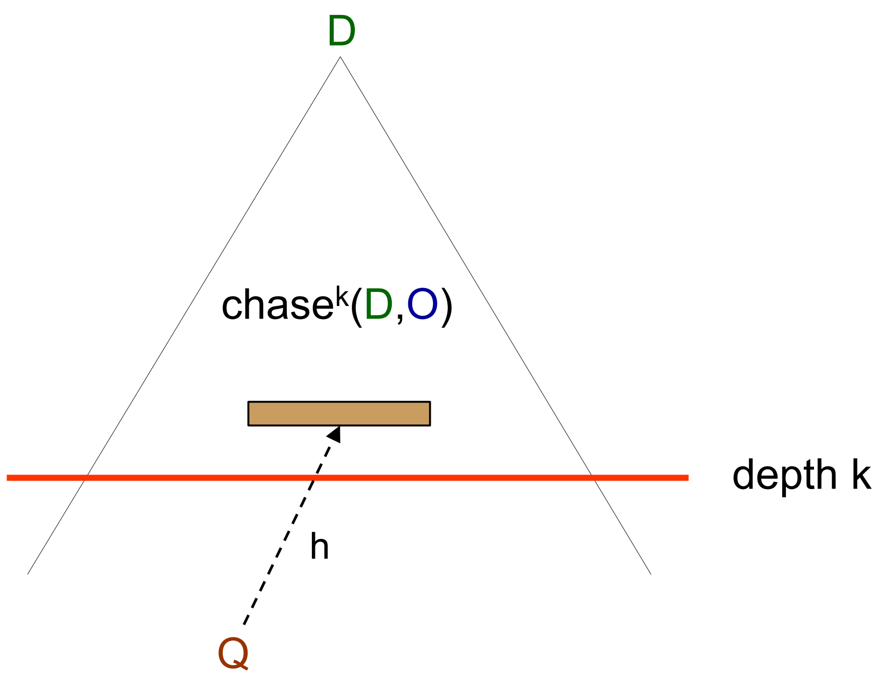
…it suffices to show that OBQA(**LINEAR**) is UCQ-rewritable

Bounded Derivation-depth Property (BDDP)

Definition: An ontology language **L** enjoys the BDDP if:

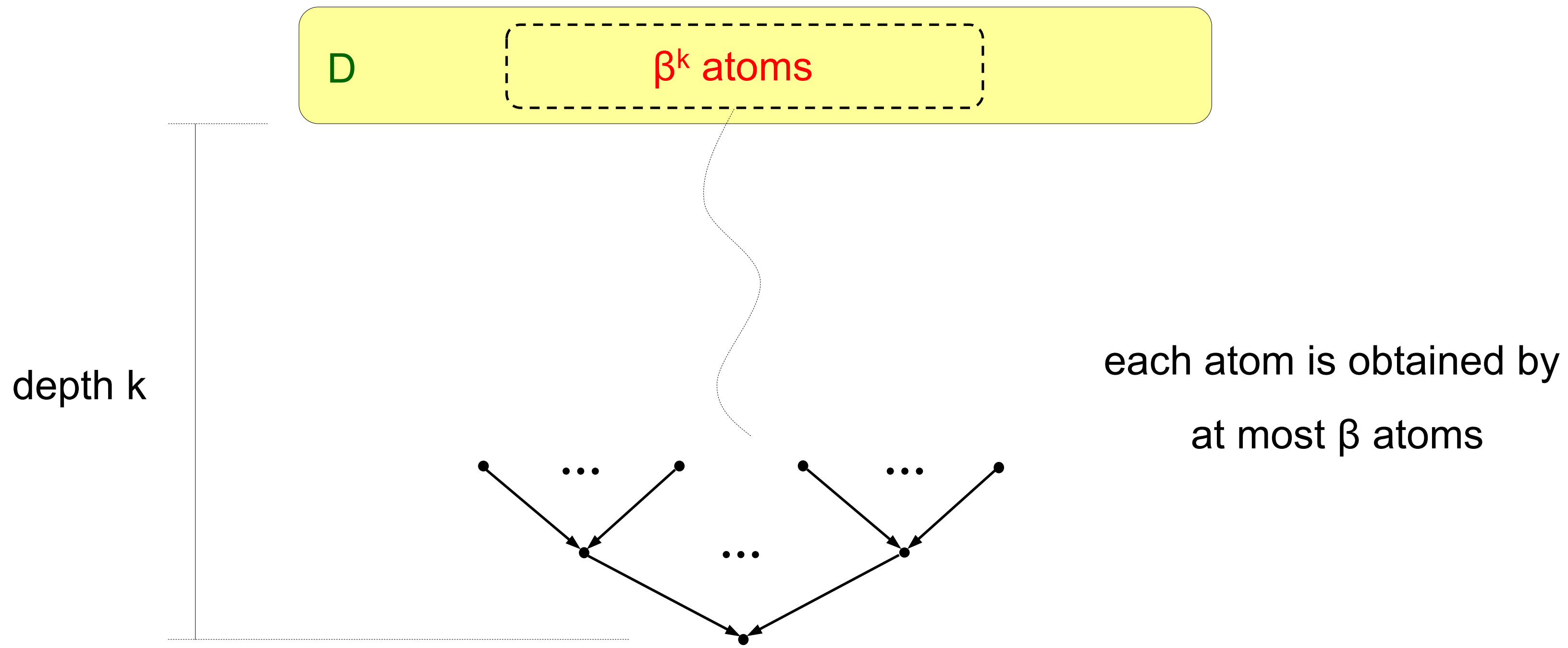
for every ontology O $\in$ **L** and CQ Q, there exists k ≥ 0 such that,

for every database D, chase(D,O) $\models$ Q $\Rightarrow$ chase^k(D,O) $\models$ Q



Bounded Derivation-depth Property (BDDP)

Proposition: **L** enjoys the BDDP $\Rightarrow$ OBQA(**L**) is UCQ-rewritable



$\Rightarrow$ to entail a CQ Q we need at most $|Q| \cdot \beta^k$ database atoms

Bounded Derivation-depth Property (BDDP)

Proposition: **L** enjoys the BDDP $\Rightarrow$ OBQA(**L**) is UCQ-rewritable

Given an ontology O $\in$ **L** and a CQ Q:

- $\mathbf{D}_{\beta,\delta,q}$ be the set of all possible databases of size at most $|Q| \cdot \beta^{\delta}$
- $\mathbf{C} = \{ D \in \mathbf{D}_{\beta,\delta,q} \mid \text{chase}(D,O) \models Q \}$
- Convert **C** into a UCQ

Complexity of OBQA(**LINEAR**)

Theorem: OBQA(**LINEAR**) is

✓ • PSPACE-complete in combined complexity

✓ • in LOGSPACE in data complexity

Recap

- Ontology-based query answering under existential rules

- Technical challenges and the right technical tool (the chase)

- Tame the infinite chase: linear existential rules - key properties and complexity

…but, is **LINEAR** the ultimate ontology language?

Research Challenge

We need classes of existential rules such that:

1. Existential quantification and recursive definitions coexist
2. OBQA is decidable, and tractable in the data complexity

$$\Downarrow$$

Tame the infinite chase:

Deal with infinite structures without explicitly building them

Transitive Closure

$$\forall x \forall y\ (\text{ParentOf}(x,y) \rightarrow \text{AncestorOf}(x,y))$$

$$\forall x \forall y \forall z\ (\text{ParentOf}(x,y) \land \text{AncestorOf}(y,z) \rightarrow \text{AncestorOf}(x,z))$$

IDB-Linear Existential Rules

- A predicate that does not occur in the head of a rule is extensional (EDB); otherwise, is intensional (IDB)

- A set of existential rules is IDB-linear if every rule is of the form

$$\forall \mathbf{x} \forall \mathbf{y}\ (\varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z}))$$

single occurrence of an IDB predicate

- We denote **IDB-LINEAR** the obtained ontology language

Transitive Closure

$$\forall x \forall y \, (\mathrm{ParentOf}(x,y) \rightarrow \mathrm{AncestorOf}(x,y))$$

$$\forall x \forall y \forall z \, (\mathrm{ParentOf}(x,y) \wedge \mathrm{AncestorOf}(y,z) \rightarrow \mathrm{AncestorOf}(x,z))$$

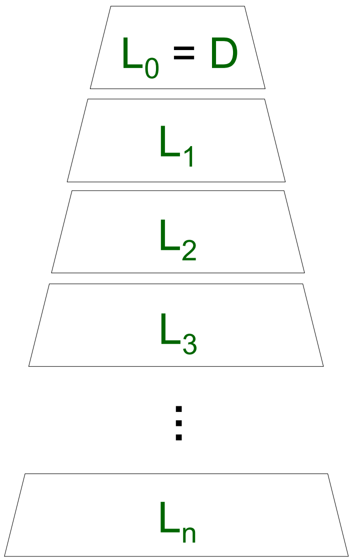
Complexity of OBQA(**IDB-LINEAR**)

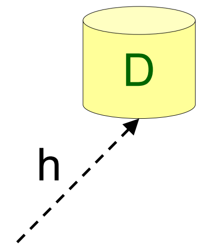
Theorem: OBQA(**IDB-LINEAR**) is

- PSPACE-complete in combined complexity
- NLOGSPACE-complete in data complexity

Complexity of **IDB-LINEAR**

Proof (high-level idea):





$\forall \mathbf{x} \forall \mathbf{y}\ (R(\mathbf{x},\mathbf{y}) \wedge \varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z}))$

and then apply the linear rule

$\forall \mathbf{x} \forall \mathbf{y}\ (R(h(\mathbf{x}),h(\mathbf{y})) \rightarrow \exists \mathbf{z}\ \psi(h(\mathbf{x}),\mathbf{z}))$

non-deterministic

level-by-level construction

Complexity of OBQA(**IDB-LINEAR**)

Theorem: OBQA(**IDB-LINEAR**) is

- PSPACE-complete in combined complexity
- NLOGSPACE-complete in data complexity

But

$$\forall \mathbf{x} \forall \mathbf{y}\ (\varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \psi(\mathbf{x},\mathbf{z}))$$

single occurrence of an IDB predicate

- We cannot have joins over null values

- We cannot express "complex" recursive definitions

…we need more sophisticated restrictions at the level of variables

Restrict the Use of Body-variables

Classification of body-variables

- Harmless: one that can be satisfied only by constants
- Harmful: one that is not harmless
- Dangerous: one that is harmful, and also appears in the rule-head

$$\forall x \forall y \forall z\ (P(x,y), S(y,z) \rightarrow \exists w\ T(y,x,w))$$

$$\forall x \forall y \forall z\ (T(x,y,z) \rightarrow \exists w\ P(w,z))$$

$$\forall x \forall y\ (P(x,y) \rightarrow \exists z\ Q(x,z))$$

Restrict the Use of Body-variables

Classification of body-variables

- Harmless: one that can be satisfied only by constants
- Harmful: one that is not harmless
- Dangerous: one that is harmful, and also appears in the rule-head

$\forall x \forall y \forall z\ (P(x,y), S(y,z) \rightarrow \exists w\ T(y,x,\underline{w}))$

$\forall x \forall y \forall z\ (T(x,y,z) \rightarrow \exists w\ P(\underline{w},z))$

$\forall x \forall y\ (P(x,y) \rightarrow \exists z\ Q(x,\underline{z}))$

Existential Positions

T[3], P[1], Q[2]

Restrict the Use of Body-variables

Classification of body-variables

- Harmless: one that can be satisfied only by constants
- Harmful: one that is not harmless
- Dangerous: one that is harmful, and also appears in the rule-head

$\forall x \forall y \forall z\ (P(\underline{x},y),\ S(y,z)\ \rightarrow\ \exists w\ T(y,\underline{x},\underline{w}))$	Existential Positions
$\forall x \forall y \forall z\ (T(x,y,z)\ \rightarrow\ \exists w\ P(\underline{w},z))$	T[3], P[1], Q[2]
$\forall x \forall y\ (P(x,y)\ \rightarrow\ \exists z\ Q(x,\underline{z}))$	T[2]

Restrict the Use of Body-variables

Classification of body-variables

- Harmless: one that can be satisfied only by constants
- Harmful: one that is not harmless
- Dangerous: one that is harmful, and also appears in the rule-head

$\forall x \forall y \forall z\ (P(\underline{x},y),\ S(y,z)\ \rightarrow\ \exists w\ T(y,\underline{x},\underline{w}))$

$\forall x \forall y \forall z\ (T(x,y,\underline{z})\ \rightarrow\ \exists w\ P(\underline{w},\underline{z}))$

$\forall x \forall y\ (P(x,y)\ \rightarrow\ \exists z\ Q(x,\underline{z}))$

Existential Positions

T[3], P[1], Q[2]

T[2], P[2]

Restrict the Use of Body-variables

Classification of body-variables

- Harmless: one that can be satisfied only by constants
- Harmful: one that is not harmless
- Dangerous: one that is harmful, and also appears in the rule-head

$\forall x \forall y \forall z\, (P(\underline{x},y), S(y,z) \rightarrow \exists w\, T(y,\underline{x},\underline{w}))$	Existential Positions
$\forall x \forall y \forall z\, (T(x,y,\underline{z}) \rightarrow \exists w\, P(\underline{w},\underline{z}))$	T[3], P[1], Q[2]
$\forall x \forall y\, (P(\underline{x},y) \rightarrow \exists z\, Q(\underline{x},\underline{z}))$	T[2], P[2], Q[1]

Restrict the Use of Body-variables

Classification of body-variables

- Harmless: one that can be satisfied only by constants
- Harmful: one that is not harmless
- Dangerous: one that is harmful, and also appears in the rule-head

$\forall x \forall y \forall z\ (P(x,y), S(y,z) \rightarrow \exists w\ T(y,x,w))$	Existential Positions
$\forall x \forall y \forall z\ (T(x,y,z) \rightarrow \exists w\ P(w,z))$	T[3], P[1], Q[2]
$\forall x \forall y\ (P(x,y) \rightarrow \exists z\ Q(x,z))$	T[2], P[2], Q[1] ✓

Weakly-Frontier-Guarded (WFG)

- A set of existential rules is WFG if every rule is of the form

$$\forall \mathbf{x} \forall \mathbf{y} \, (\varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z} \, \psi(\mathbf{x},\mathbf{z}))$$

there exists a guard atom that contains all the dangerous variables

- We denote **WFG** the obtained ontology language

Complexity of OBQA(**WFG**)

Theorem: OBQA(**WFG**) is

- 2EXPTIME-complete in combined complexity
- EXPTIME-complete in data complexity

Source of complexity: The guard and the rest of the body share harmful variables

Warded

- A set of existential rules is warded if every rule is of the form

$$\forall \mathbf{x} \forall \mathbf{y} \, (G(\mathbf{x},\mathbf{y}) \land \varphi(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z} \, \psi(\mathbf{x},\mathbf{z}))$$

contains all the dangerous variables, and shares with $\varphi(\mathbf{x},\mathbf{y})$ only harmless variables

- We denote **WARDED** the obtained ontology language

Complexity of OBQA(**WARDED**)

Theorem: OBQA(**WARDED**) is

- EXPTIME-complete in combined complexity
- PTIME-complete in data complexity

a "nearly" maximal fragment of WFG

at least one occurrence of a dangerous variable that appears in the guard,

appears outside the guard $\Rightarrow$ EXPTIME-complete in data complexity

Warded + Stratified Negation