Tree-Shaped Aggregate Questions over Ontologies

Camilo Thorne, Diego Calvanese

KRDB Research Centre

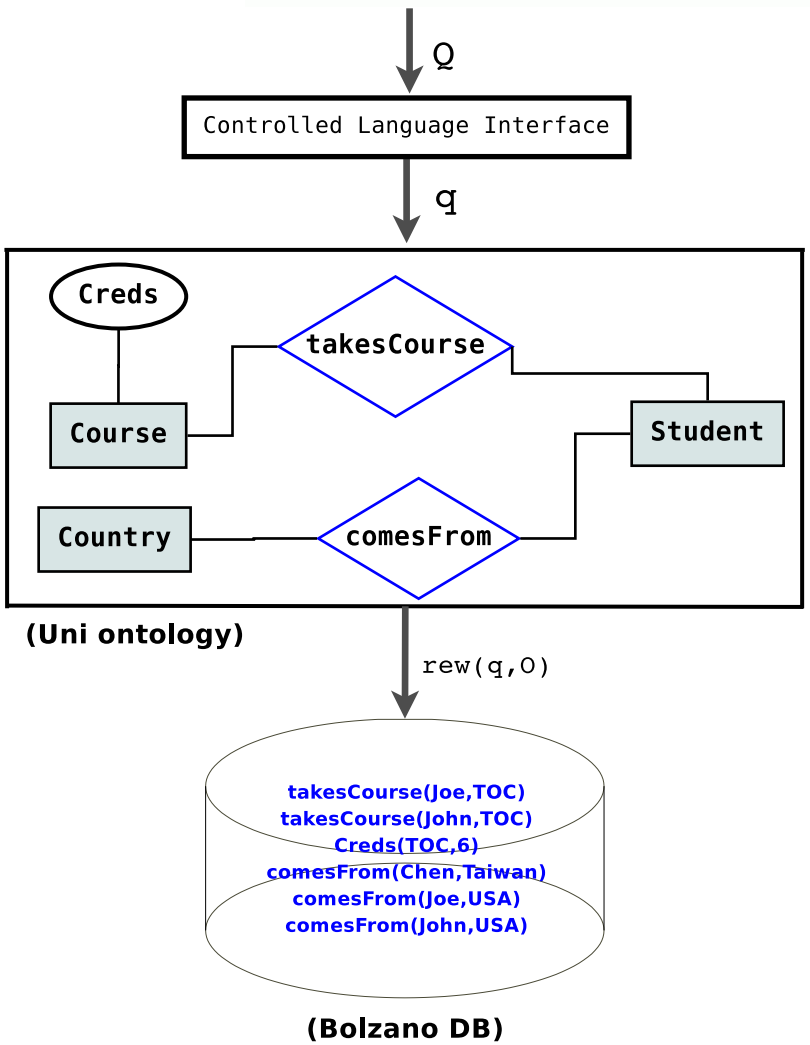
Free University of Bozen-Bolzano

Flexible Query Answering Systems - FQAS 2009

CL Interfaces

- Controlled language interfaces (CLIs) to information systems have been proposed to address the usability problem [Sowa 2004, Fuchs et al. 2006]
- Controlled languages (CLs) are ambiguity-free fragments of natural language which
  - symbolically translate into formal ontology and/or query languages
  - the translation is compositional and polynomial in the size of the input
- More recently, CLIs have been proposed for ontology-based data access systems (OBDASs) [Bernstein et al. 2005, 2007]

OBDASs



In an ontology based data access system $(\mathcal{O}, \mathcal{D})$ data stored in databases $\mathcal{D}$ is queried through an ontology $\mathcal{O}$

Questions are translated into queries, expanded w.r.t. $\mathcal{O}$ and shipped to $\mathcal{D}$

Measuring CLI Performance

- An important measure of performance of OBDASs is the data complexity of query evaluation [Vardi 1982]
- CL constructs in formal ontology/query languages have an impact on data complexity:
  - which combinations of CL constructs give way to tractable (**PTIME** or less) data complexity?
  - which combinations of CL constructs give rise to intractable (**coNP**-hard) data complexity?

Measuring CLI Performance

- An important measure of performance of OBDASs is the data complexity of query evaluation [Vardi 1982]
- CL constructs in formal ontology/query languages have an impact on data complexity:
  - which combinations of CL constructs give way to tractable (**PTIME** or less) data complexity?
  - which combinations of CL constructs give rise to intractable (**coNP**-hard) data complexity?
- We can study this impact by looking at what happens with the formal query/ontology languages they translate into!
- Target: SQL aggregate queries

Tree-Shaped Conditions and Tree Shaped Queries

- We consider logic-based declarative counterparts of SQL
- A tree-shaped condition (TSC) $\varphi(x)$ is a FOL formula

$$\begin{array}{rcl} \varphi(x) & ::= & A(x) \mid \exists y R(x,y) \\ & & \mid \exists y (R(x,y) \wedge \varphi(y)) \\ & & \mid \varphi(x) \wedge \varphi'(x) \end{array}$$

- A tree-shaped query (TSQ) q is a query

$$q(x) \leftarrow \varphi(x)$$

with $\varphi(x)$ a TSC

Tree-Shaped Conditions and Tree Shaped Queries

- We consider logic-based declarative counterparts of SQL
- A tree-shaped condition (TSC) $\varphi(x)$ is a FOL formula

$$\begin{array}{rcl} \varphi(x) & ::= & A(x) \mid \exists y R(x,y) \\ & & \mid \exists y (R(x,y) \wedge \varphi(y)) \\ & & \mid \varphi(x) \wedge \varphi'(x) \end{array}$$

- A tree-shaped query (TSQ) q is a query

$$q(x) \leftarrow \varphi(x)$$

  with $\varphi(x)$ a TSC

- In domains with numerical data we want also to aggregate!

Aggregate Tree-Shaped Queries

- We consider SQL aggregation functions: max, min, count, cntd, sum and avg, denoted α
- An aggregate tree-shaped query (ATQ) q is a query

$$q(x, \alpha(y)) \leftarrow \varphi_1(x) \wedge R(x, y) \wedge \varphi_2(y)$$

  - x is the grouping variable
  - y is the aggregation variable
  - $\alpha(y)$ is the aggregation term
  - $\varphi_1(x)$ and $\varphi_2(y)$ are TSCs
  - $R(x, y)$ is a FOL atom
- Note: $\varphi_1(x)$ expresses GROUP-BY clauses

Ontology Languages

- We are interested in the OWL ontology language and its fragments
- OWL is formally underpinned by description logics (DLs)
- *DL-Lite*: captures the fundamental features of conceptual modelling languages (e.g., ER-diagrams)
  - IS-A, disjointness, keys
  - role typing, mandatory participation
- $\mathcal{ALCI}$: the least subset of OWL containing *DL-Lite* and closed under boolean operations

Certain Answers

- Aggregate functions are interpreted as in SQL
- Given $q(x, \alpha(y)) \leftarrow \psi$ and $(\mathcal{O}, \mathcal{D})$:
  - a certain assignments γ is

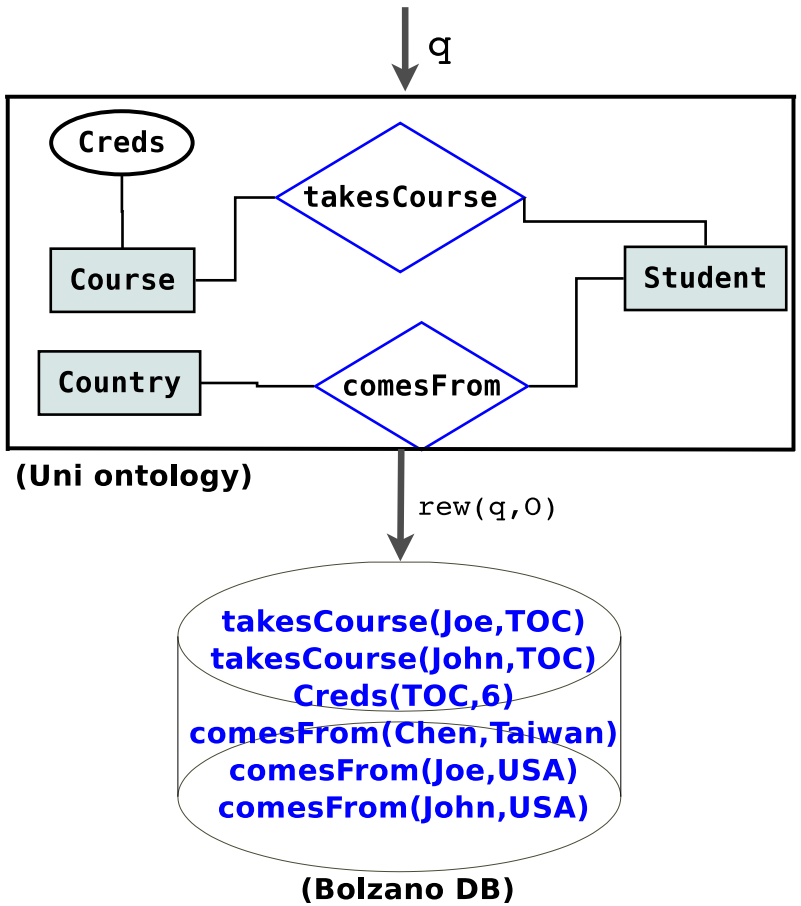
    a FOL satisfying assignment for ψ
    over each $\mathcal{D}' \supseteq \mathcal{D}$ s.t. $\mathcal{D}' \models \mathcal{O}$
  - the certain group $H_{\gamma(x)}$ collects

    the values $\gamma'(y)$ of y
    for all γ s.t. $\gamma'(x) = \gamma(x)$
  - the set of certain answers $cert(q, \mathcal{O}, \mathcal{D})$ collects

    the tuples $(\gamma(x), \alpha(H_{\gamma(x)}))$

Example

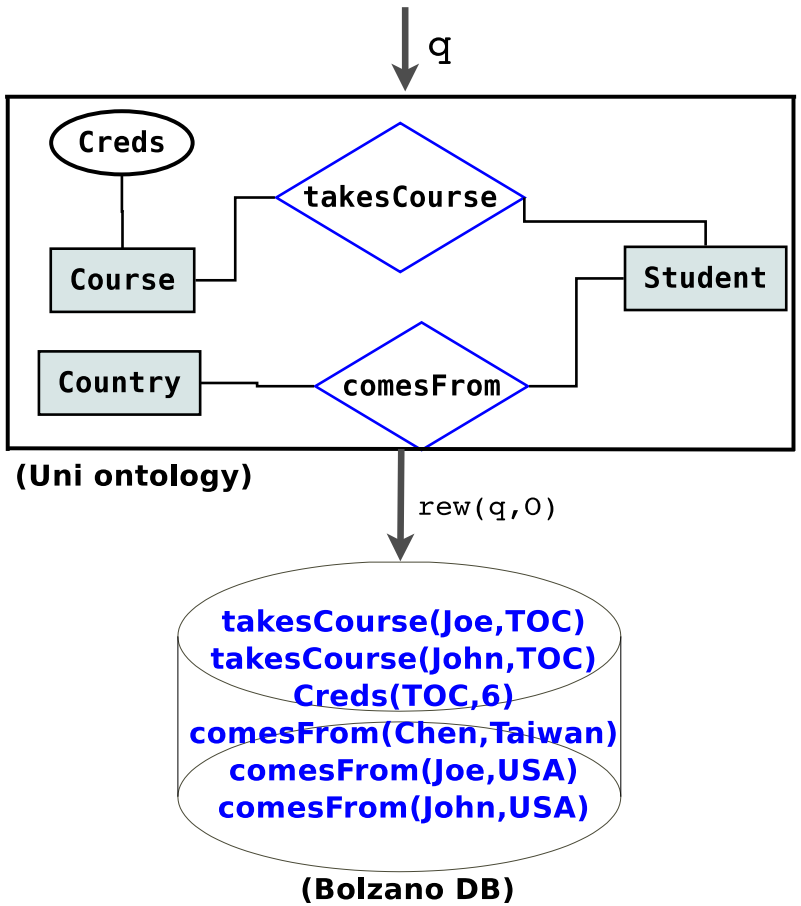
Which is the number of students, per country, who study something?



$$q(x, count(z)) \leftarrow Student(x) \wedge \exists y(takesCourse(x, y)) \wedge comesFrom(x, z) \wedge Country(z))$$

Example

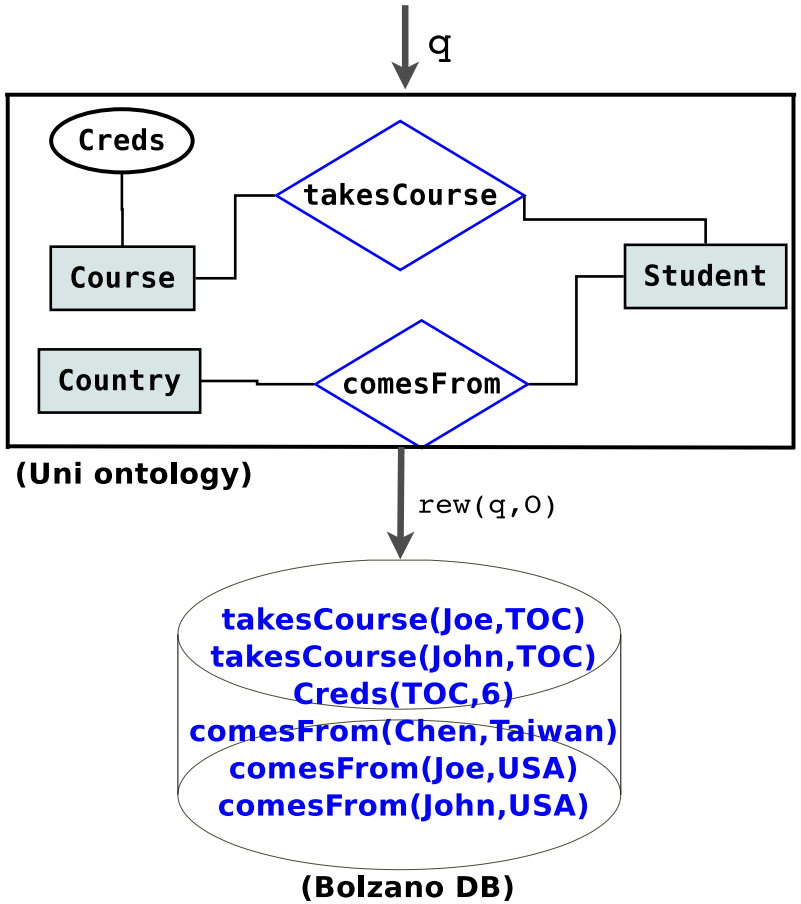
Which is the number of students, per country, who study something?



$$q(x, count(z)) \leftarrow Student(x) \wedge \exists y(takesCourse(x, y)) \wedge comesFrom(x, z) \wedge Country(z))$$

Example

Which is the number of students, per country, who study something?



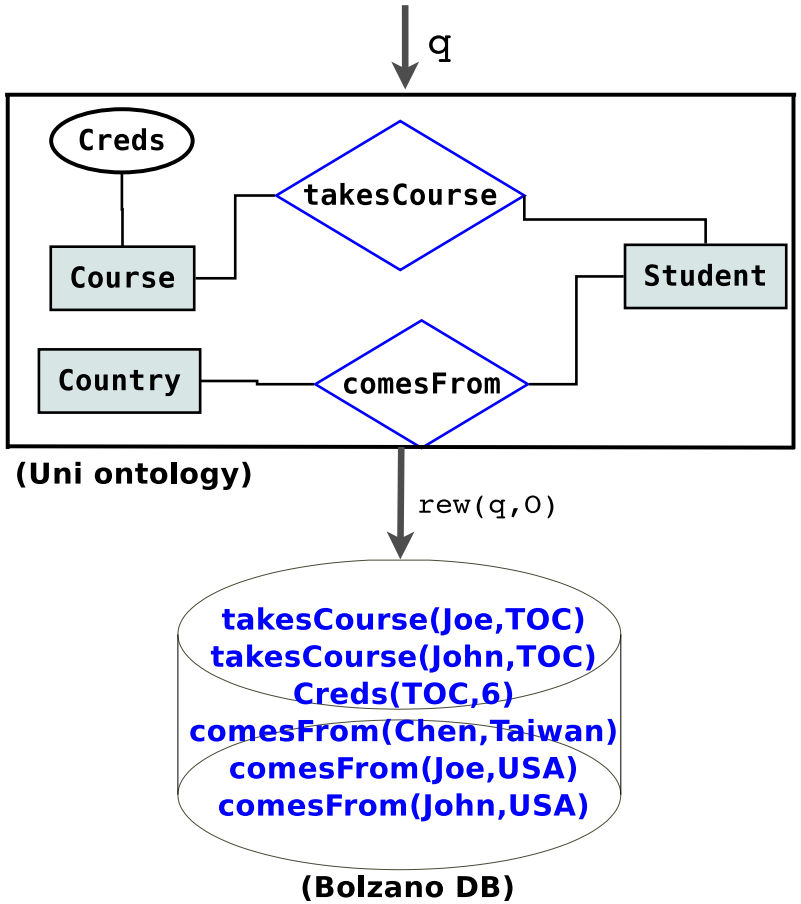
$$q(x, count(z)) \leftarrow Student(x) \wedge \exists y(takesCourse(x, y)) \wedge comesFrom(x, z) \wedge Country(z))$$

After evaluating and grouping

$$H_{\text{USA}} = \{\text{John,Joe}\}$$
$$H_{\text{Taiwan}} = \{\}$$

Example

Which is the number of students, per country, who study something?



$$q(x, count(z)) \leftarrow Student(x) \wedge \exists y(takesCourse(x, y)) \wedge comesFrom(x, z) \wedge Country(z))$$

Therefore

$$cert(q, \mathcal{O}_{\mathsf{Uni}}, \mathcal{D}_{\mathsf{Bz}}) = \{(\mathsf{USA}, 2), (\mathsf{Taiwan}, 0)\}$$

ATQ-English and Formal Semantics

- To express ATQs we use computational semantics compositional translations $\tau(\cdot)$

ATQ-English and Formal Semantics

- To express ATQs we use computational semantics compositional translations $\tau(\cdot)$

- Given $\{e, t, \mathbb{N}, \mathbb{Q}\}$ as basic types we define aggregate determiners like

$$\tau(\text{the average number of}) := \lambda P^{e\to\mathbb{N}}.avg(P)\colon (e\to\mathbb{N})\to\mathbb{Q}$$
$$\tau(\text{the number of}) := \lambda P^{e\to\mathbb{N}}.count(P)\colon (e\to\mathbb{N})\to\mathbb{Q}$$

- We make use of semantically annotated grammars, i.e., context-free grammars with rules like

$$Q_{wh} \longrightarrow I_i\ S_{g_i}? \qquad \tau(Q_{wh}) := \lambda\bar{z}.\tau(I_i)(\lambda i.\tau(S_{g_i}))$$
$$S_{g_i} \longrightarrow NP_{g_i}\ VP_i \qquad \tau(S_{g_i}) := \tau(NP_{g_i})(\tau(VP_i))$$

ATQ-English: Sample Lexicon

$\tau(\text{the number of}) := \lambda P^{e\to\mathbb{N}}.\mathit{count}(P)\colon (e\to\mathbb{N})\to\mathbb{Q}$

$\tau(\text{who}) := \lambda P^{e\to\mathbb{N}}\lambda Q^{e\to\mathbb{N}}.\lambda x^{e}.(P(x)\wedge Q(x))\colon (e\to\mathbb{N})\to((e\to\mathbb{N})\to(e\to\mathbb{N}))$

$\tau(\text{something}) := \lambda P^{e\to t}.\exists y^{e}P(y)\colon (e\to\mathbb{N})\to t$

$\tau(\text{per country}) := \lambda P^{e\to\mathbb{N}}.\lambda y^{e}.(P(y)\wedge \mathit{Country}(z)\wedge \mathit{comesFrom}(y,z))\colon (e\to\mathbb{N})\to(e\to\mathbb{N})$

$\tau(\text{per country}) := \lambda Q^{e\to\mathbb{N}}.\lambda P^{e\to\mathbb{N}}\lambda y^{e}.(P(y)\wedge \mathit{Country}(z)\wedge \mathit{comesFrom}(y,z)\wedge Q(z))\colon$
$(e\to\mathbb{N})\to((e\to\mathbb{N})\to(e\to\mathbb{N}))$

$\tau(\text{which}) := \lambda R^{e\to(\mathbb{Q}\to t)}.\lambda n^{\mathbb{Q}}.R(n,m)\colon (e\to(\mathbb{Q}\to t))\to(\mathbb{Q}\to t)$

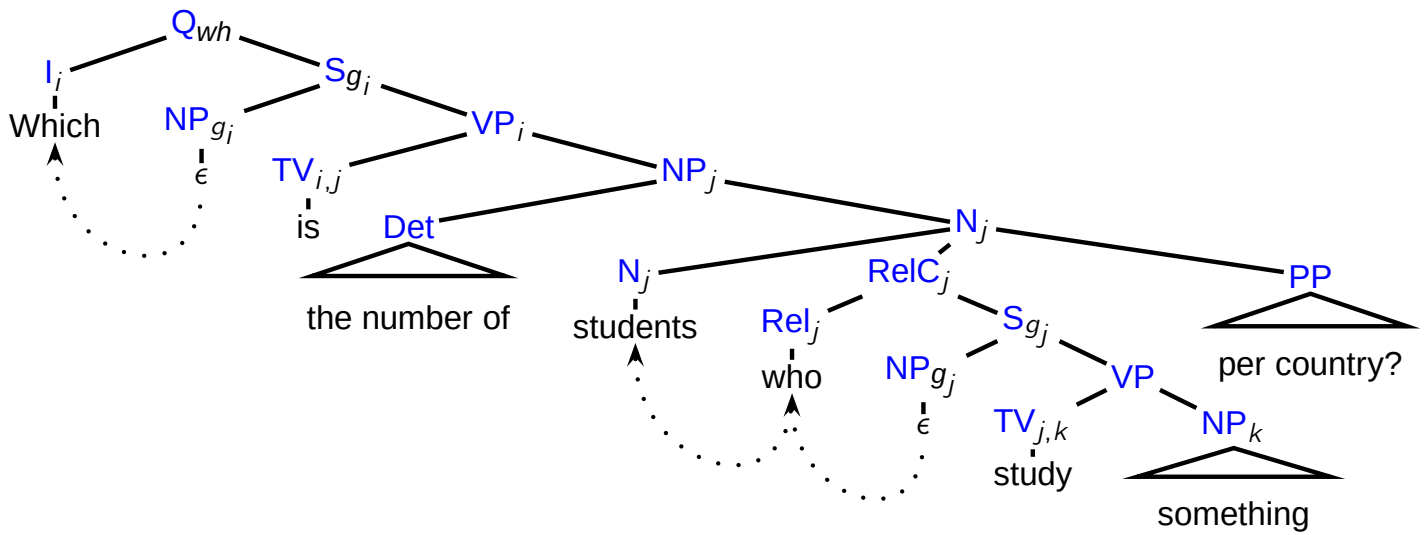
$\tau(\text{is}) := \lambda n^{\mathbb{Q}}.\lambda m^{\mathbb{Q}}.n\approx m\colon \mathbb{Q}\to(\mathbb{Q}\to t)$

$\tau(\epsilon) := \lambda P^{\mathbb{Q}\to t}.P(n)\colon (\mathbb{Q}\to t)\to t$

$\tau(\text{students}) := \lambda x^{e}.\mathit{Student}(x)\colon e\to\mathbb{N}$

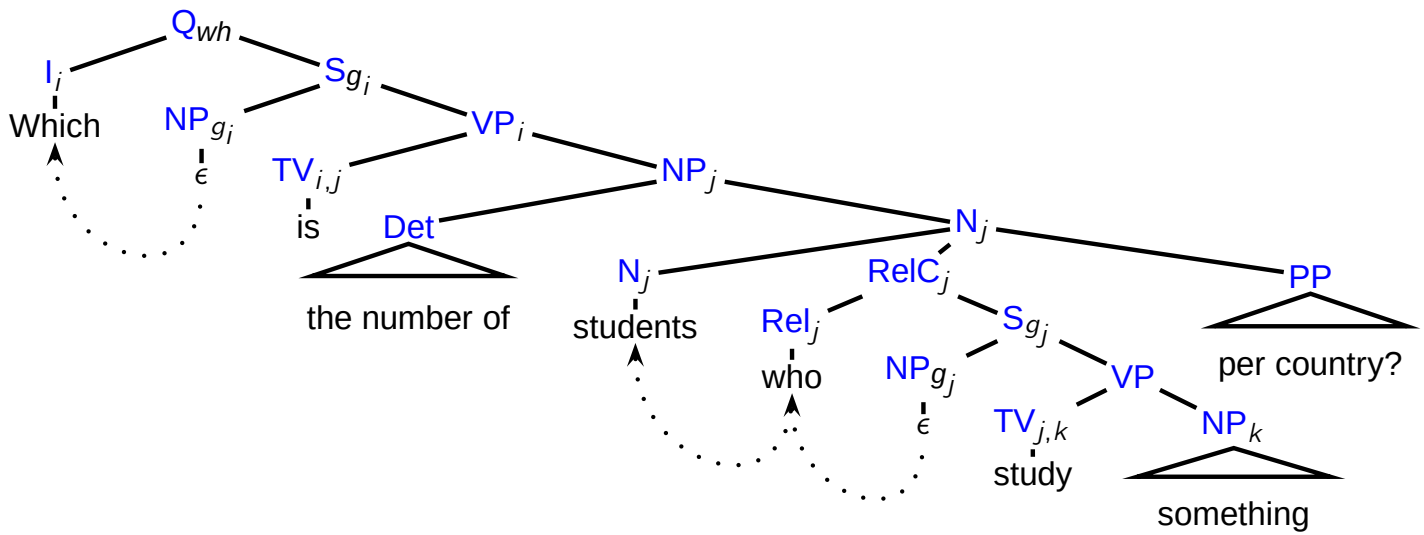
$\tau(\text{study}) := \lambda\alpha^{(e\to\mathbb{N})\to\mathbb{N}}.\lambda x^{e}.\alpha(\lambda y^{e}.\mathit{takesCourse}(x,y))\colon ((e\to\mathbb{N})\to\mathbb{N})\to(e\to\mathbb{N})$

ATQ-English: Expressing Aggregations and Grouping



$$\lambda z^e.\lambda m^{\mathbb{Q}}.m \approx \mathit{count}(\lambda x^e.\mathit{Student}(x) \wedge \exists y^e(\mathit{takesCourse}(x,y)) \wedge \mathit{comesFrom}(x,z) \wedge \mathit{Country}(z)) : e \rightarrow (\mathbb{Q} \rightarrow t)$$

ATQ-English: Expressing Aggregations and Grouping



$$q(x, count(z)) \leftarrow Student(x) \wedge \exists y(takesCourse(x, y)) \wedge comesFrom(x, z) \wedge Country(z))$$

Data Complexity of ATQ-English

- Which is the number of students, per country, who study something?

```
SELECT s1.CName, COUNT(s.SName)
FROM comesFrom s1
WHERE EXISTS (SELECT *
    FROM takesCourse s2
    WHERE s1.SName = s2.SName)
GROUP BY s1.CName
```

Data Complexity of ATQ-English

- Which is the number of students, per country, who study something?

```
SELECT s1.CName, COUNT(s.SName)
FROM comesFrom s1
WHERE EXISTS (SELECT *
    FROM takesCourse s2
    WHERE s1.SName = s2.SName)
GROUP BY s1.CName
```

- Answering an ATQ is in **LOGSPACE** in data complexity when combined with ontologies from the *DL-Lite* family

Beyond ATQs and TSQs

- Disjunctions $\psi(x) \vee \psi'(x)$ of TSCs do not increase data complexity either

Beyond ATQs and TSQs

- Disjunctions $\psi(x) \vee \psi'(x)$ of TSCs do not increase data complexity either

- But negation does:

  Which is the number of students who do not come from Italy?

```
SELECT COUNT(s1.SName)
FROM comesFrom s1
WHERE s1.CName != "Italy"
```

Beyond ATQs and TSQs

- Disjunctions $\psi(x) \vee \psi'(x)$ of TSCs do not increase data complexity either

- Complexity jumps to **coNP**-hard with

  - negations of TSCs, $\neg\psi(x)$
  - universally restricted TSCs, $\forall x(R(x,y) \rightarrow \psi(y))$
  - comparisons, $t\,\theta\,t'$, $\theta \in \{=, >, <, \geq, \leq\}$

Conclusions

- We have proposed to measure the performance of CLIs through the data complexity of data access
- We have expressed in CL ATQs
- We have shown that
  - aggregations, disjunctions, conjuctions and existential quatifications are cheap to use in CLIs to OBDASs
  - extending coverage to negation, comparisons and universal restrictions is costly
- We can capture in CL and answer efficiently a significant class of SQL aggregate queries over OBDASs

Data Complexity

	$\vee$-ATQs/TSQs	$\leq$-ATQs/TSQs	$\forall$-ATQs/TSQs	$\neg$-ATQs/TSQs
DL-Lite	in **LogSpace**	**coNP**-complete	**coNP**-complete	**coNP**-complete
$\mathcal{ALCI}$	**coNP**-complete	**coNP**-hard	**coNP**-complete	**coNP**-complete
OWL	**coNP**-hard	**coNP**-hard	**coNP**-hard	**coNP**-hard

- We derive a **coNP** upper bound when $\mathcal{O}$ and q are in the 2-variable fragment of FOL [Pratt 2009]

Data Complexity

	$\vee$-ATQs/TSQs	$\leq$-ATQs/TSQs	$\forall$-ATQs/TSQs	$\neg$-ATQs/TSQs
DL-Lite	in **LogSpace**	**coNP**-complete	**coNP**-complete	**coNP**-complete
$\mathcal{ALCI}$	**coNP**-complete	**coNP**-hard	**coNP**-complete	**coNP**-complete
OWL	**coNP**-hard	**coNP**-hard	**coNP**-hard	**coNP**-hard

- We derive a **coNP** upper bound when $\mathcal{O}$ and q are in the 2-variable fragment of FOL [Pratt 2009]
- But evidence from question corpora suggests that such **coNP**-hard constructs are infrequent [Bernardi et al. 2007]

coNP-*hardness of* $\forall$*-TSQs: Reduction from* 2+2-SAT

A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses in **NP**-complete [Scharf, 1994]

Encode Φ into DB $\mathcal{D}_\Phi$

$$\vdots$$

$$P(c_i, l_{i,1}), P(c_i, l_{i,2}), N(c_i, n_{i,1}), N(c_i, n_{i,2})$$

$$\vdots$$

$$A_t(\text{true})$$

Consider $\mathcal{O}_\Phi$

$$\forall x(A_f(x) \rightarrow \neg A_t(x))$$

coNP-*hardness of* $\forall$*-TSQs: Reduction from* 2+2-SAT

A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses in **NP**-complete [Scharf, 1994]

Ask TCQ q_Φ of body ψ_Φ

$$\begin{aligned}
&\exists c \exists l_1 \exists l_2 \exists l_3 \exists l_4 (\\
&P_1(c, l_1) \wedge \forall v_1 (\mathit{Val}(l_1, v_1) \rightarrow A_f(v_1)) \wedge \\
&P_2(c, l_2) \wedge \forall v_2 (\mathit{Val}(l_2, z_2) \rightarrow A_f(v_2)) \wedge \\
&N_1(c, l_3) \wedge \exists v_2 (\mathit{Val}(l_3, v_3) \wedge A_t(v_3)) \wedge \\
&N_2(c, l_4) \wedge \exists v_3 (\mathit{Val}(l_4, v_4) \wedge A_t(v_4)))
\end{aligned}$$

coNP-*hardness of* $\forall$*-TSQs: Reduction from* 2+2-SAT

A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses in **NP**-complete [Scharf, 1994]

We claim

$$() \notin \mathit{cert}(q_\Phi, \mathcal{O}_\Phi, \mathcal{D}_\Phi) \text{ iff } \Phi \text{ has no model}$$

coNP-*hardness of* $\forall$*-TSQs: Reduction from* 2+2-SAT

A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses in **NP**-complete [Scharf, 1994]

"$\Rightarrow$" given a $\mathcal{D}$ s.t.

- $\mathcal{D} \models \mathcal{O}_\Phi$
- $\mathcal{D}_\Phi \subseteq \mathcal{D}$
- $\mathcal{D} \not\models \psi_\Phi$

Then

$$\begin{aligned} \forall v(\mathit{Val}(I, v) \wedge A_f(v)) \quad &\Rightarrow_{\mathcal{D}} \quad \forall v(\mathit{Val}(I, v) \wedge \neg A_f(v)) \\ &\equiv_{\mathcal{D}} \quad \neg\exists v_i(\mathit{Val}(I, v) \wedge A_t(v)) \end{aligned}$$

coNP-*hardness of* $\forall$*-TSQs: Reduction from* 2+2-SAT

A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses in **NP**-complete [Scharf, 1994]

Define ta $\delta(\cdot)$ by putting

$$\delta(I) = \text{true iff } (I, \text{true}) \in \mathit{Val}^{\mathcal{D}}$$

coNP-*hardness of* $\forall$*-TSQs: Reduction from* 2+2-SAT

A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses in **NP**-complete [Scharf, 1994]

"$\Leftarrow$": Symmetrical argument

coNP-*hardness of* $\forall$*-TSQs: Reduction from* 2+2-SAT

A conjunction of 2+2 clauses is a conjunction $\Phi := C_1 \wedge \cdots \wedge C_k$

$$C_i := p_{i1} \vee p_{i2} \vee \neg n_{i1} \vee \neg n_{i2}$$

SAT for 2+2 clauses in **NP**-complete [Scharf, 1994]

Therefore:

$$() \notin \mathit{cert}(q_\Phi, \mathcal{O}_\Phi, \mathcal{D}_\Phi) \text{ iff } \Phi \text{ has no model}$$

Moral: conjunction, universal and existential quantification in queries and disjointness in ontologies blows up data complexity!