Data Complexity of Answering Ontology-Mediated Queries with a Covering Axiom

Olga Gerasimova

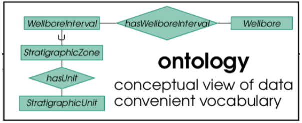
Research advisor: professor M.V. Zakharyaschev

Department of Data Analysis and Artificial Intelligence
Faculty of Computer Science
National Research University Higher School of Economics

19 November, 2018

Ontology in Computer Science

An **ontology** is an engineering artefact that

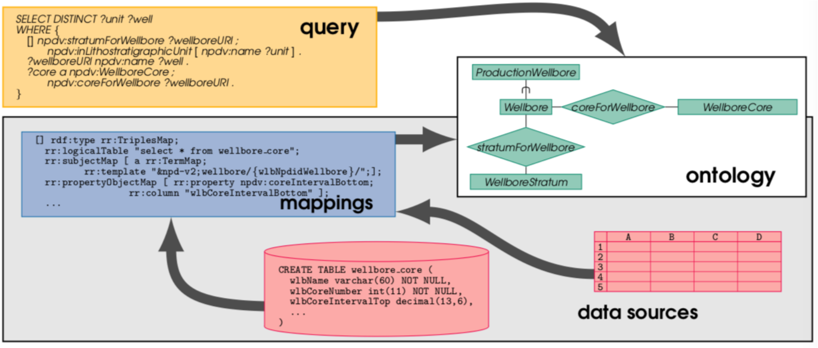


- is constituted by a **specific vocabulary** used to describe a certain domain of interest;
- is a set of **explicit assumptions (rules)** regarding the intended meaning of the vocabulary;
- is a **formal and machine manipulable** model of a domain of interest.

Definition

Ontology – a formal report of *individuals*, *classes* and *properties* (or relations) that are considered in the framework of a domain of interest.

Ontology-based data access (OBDA)



1. gives a high-level conceptual view of the data
2. provides the user with a convenient vocabulary for queries
3. allows the system to enrich incomplete data with background knowledge
4. supports queries to multiple and heterogeneous data sources

Mapping

For example, we have the table

title

movie ID	title	year	kind
728	'Django Unchained'	2012	1
1257	'Game of Thrones'	2011	2
2543	'Blade Runner'	1982	1

and the ontology vocabulary containing **classes** such as *Movie*, *Title*, *Year*, *Kind* and **relations** such as *has_title*, *was_released_in* and *has_kind*.

We need to translate the table into the data schemes related to the ontology vocabulary via mappings. Then, we obtain:

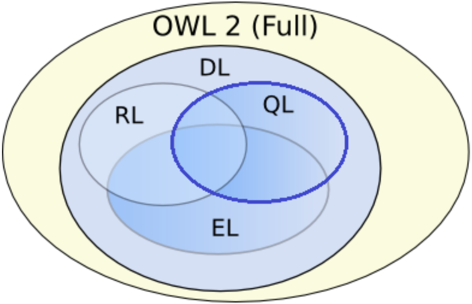
– *Movie*(728), *Movie*(1257), *Movie*(2543), *Title*(Django Unchained), *Title*(Game of Thrones), *Title*(Blade Runner), *Year*(2012), *Year(2011)*, *Year*(1982), *Kind*(1), *Kind*(2);

– *has_title*(728,Django Unchained), *was_released_in*(728,2012), *has_kind*(728,1), . . .

Description Logic

Definition

Description Logic (DL) is an area of knowledge representation and reasoning in Artificial Intelligence and the Semantic Web that studies **logic-based formalisms** whose languages operate with **concepts** to represent classes of individuals in an application domain, and **roles** to represent binary relations between the individuals.



OWL2QL (OWL – Web Ontology Language) contains *individual names* a_i, *concept names* A_i, and *role names* P_i ($i = 1; 2; ...$).

$$
\begin{array}{llllll}
R & ::= & P_i & \mid & P_i^-, & \\
B & ::= & \bot & \mid & A_i & \mid \quad \exists R, \\
C & ::= & B & \mid & \exists R.B &
\end{array}
$$

TBox, T, – a finite set of (1) concept and role inclusions, (2) concept and role disjointness constraints of the forms

$$B \sqsubseteq C, \qquad R_1 \sqsubseteq R_2$$

$$B_1 \sqcap B_2 \sqsubseteq \bot, \qquad R_1 \sqcap R_2 \sqsubseteq \bot$$

ABox, A, – a finite set of assertions of the form $A_k(a_i)$ and $P_k(a_i; a_j)$ and *inequality constraints* $a_i \neq a_j$ for $i \neq j$.
T and A together constitute **the knowledge base** $K = (T; A)$.

Query Rewriting

- The user formulates a *query* **q** in the vocabulary of a given *ontology* **T** .

The task of an OBDA system

To 'rewrite' q and T into a new query q_0 in the vocabulary of the data such that, for any possible data A (in this vocabulary), the answers to q over $(T; A)$ are precisely the same as the answers to q_0 over A.

Example

(1) $Dancer \sqsubseteq \exists isTrainedBy.Choreographer$

(2) $\exists participates.Competition \sqsubseteq Dancer$

(3) $Choreographer \sqsubseteq MasterOfSport$

(4) $StandartCh \sqsubseteq Choreographer$

(5) $LatinaCh \sqsubseteq Choreographer$

(6) $StandartCh \sqcap LatinaCh \sqsubseteq \bot$

(7) $isTrainedBy^{-} \sqsubseteq trains$

(8) $trains \sqsubseteq isTrainedBy^{-}$

$$q(x) = \exists y\ (MasterOfSport(y) \wedge trains(y, x))$$

$$q'(x) = \exists y\ [(\mathbf{MasterOfSport(y)} \vee StandartCh(y) \vee LatinaCh(y) \vee Choreographer(y$$

$$\wedge\ (\mathbf{trains(y,x)} \vee isTrainedBy(x, y))] \vee Dancer(x) \vee\ \exists z\ participates(x, z)$$

$\mathcal{A} = \{Dancer(a), Choreographer(b), trains(b, c), participates(d, e)\}$
Answer to $q(x)$ according to $\mathcal{A}$: $\{\varnothing\}$
Answer to $q'(x)$ according to $\mathcal{A}$: $\{a, c, d\} \Leftrightarrow q(x)$ according to $(\mathcal{T}; \mathcal{A})$

Data complexity of OMQ answering

Big problem 1: how to determine the data complexity of answering a given ontology-mediated query $(\mathcal{O}, \boldsymbol{q})$ with an ‘expressive’ ontology $\mathcal{O}$ and a Boolean conjunctive query $\boldsymbol{q}$?

That is, the complexity of deciding whether $\mathcal{O}, \mathcal{D} \models \boldsymbol{q}$ for a given input data instance $\mathcal{D}$

Big problem 1′: how to determine whether $(\mathcal{O}, \boldsymbol{q})$ is

- FO-rewritable, that is, whether there is an FO-sentence $\boldsymbol{q}'$ such that, for any data D, we have $\mathcal{O}, \mathcal{D} \models \boldsymbol{q}$ iff $\mathcal{D} \models \boldsymbol{q}'$ (AC^0)
- linear-datalog-rewritable (in NL)
- datalog-rewritable (in PTime)

Our little problem

Covering axiom

$A \sqsubseteq T \sqcup F$ class A is covered by classes T and F

Example: $Animal \sqsubseteq Female \sqcup Male$

Global covering and disjointness

$\top \sqsubseteq T \sqcup F$ everything is covered by T and F

Disjointness: $T \sqcap F \sqsubseteq \bot$ $\top \sqsubseteq Alive \sqcup Dead, \quad Alive \sqcap Dead \sqsubseteq \bot$

Notation:

$\mathrm{Cov}_\top = \{\top \sqsubseteq F \sqcup T\}$

$\mathrm{Cov}_\top^\bot = \{\top \sqsubseteq F \sqcup T, \; F \sqcap T \sqsubseteq \bot\}$

$\mathrm{Cov}_A = \{A \sqsubseteq F \sqcup T\}$

$\mathrm{Cov}_A^\bot = \{A \sqsubseteq F \sqcup T, \; F \sqcap T \sqsubseteq \bot\}$

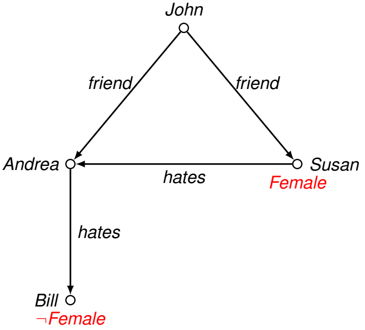
Problem

Classify <u>syntactically</u> CQs $\boldsymbol{q}$ w.r.t. complexity of answering $(\mathrm{Cov}, \boldsymbol{q})$

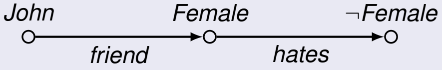
Andrea example

Consider the ABox $\mathcal{A}$:

(John; Susan) : friend
(John; Andrea) : friend
(Susan; Andrea) : hates
(Andrea; Bill) : hates
Susan: Female
Bill : ¬Female



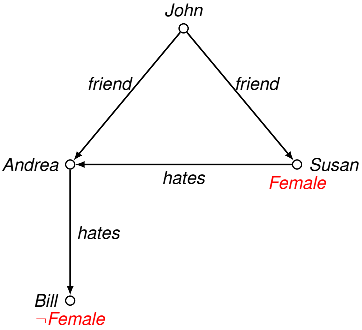
Does John have a female friend who hates a male (not female) person?



Andrea example

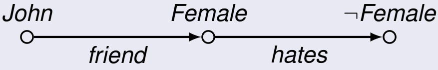
Consider the ABox $\mathcal{A}$:

(John; Susan) : friend
(John; Andrea) : friend
(Susan; Andrea) : hates
(Andrea; Bill) : hates
Susan: Female
Bill : ¬Female



$\mathcal{T} = \{\top \sqsubseteq \mathit{Female} \sqcup \mathit{Male}\}$

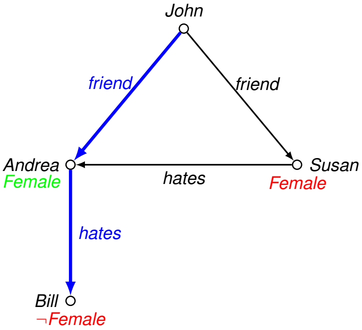
Does John have a female friend who hates a male (not female) person?



Andrea example

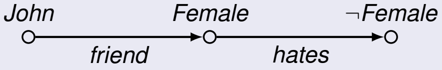
Consider the ABox $\mathcal{A}$:

(John; Susan) : friend
(John; Andrea) : friend
(Susan; Andrea) : hates
(Andrea; Bill) : hates
Susan: Female
Bill : ¬Female



$\mathcal{T} = \{\top \sqsubseteq \mathit{Female} \sqcup \mathit{Male}\}$

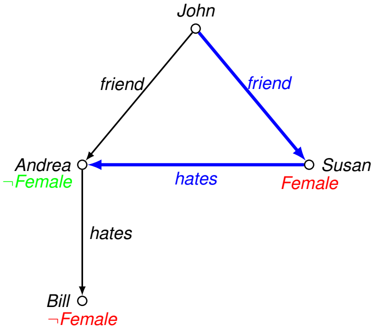
Does John have a female friend who hates a male (not female) person?



Andrea example

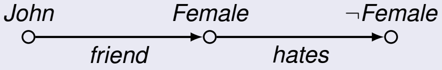
Consider the ABox $\mathcal{A}$:

(John; Susan) : friend
(John; Andrea) : friend
(Susan; Andrea) : hates
(Andrea; Bill) : hates
Susan: Female
Bill : ¬Female

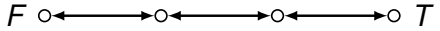
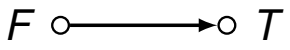
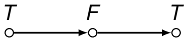


$\mathcal{T} = \{\top \sqsubseteq \mathit{Female} \sqcup \mathit{Male}\}$

Does John have a female friend who hates a male (not female) person?



Typical examples for $\mathrm{Cov}_A = \{A \sqsubseteq F \sqcup T\}$

Complexity	CQ $\boldsymbol{q}$	Explanation
AC^0		if $\boldsymbol{q}$ has only F atoms but no T, then the F can be ignored
L		checks undirected reachability:  the answer to $\boldsymbol{Q}$ is 'yes'
NL		checks directed reachability:  the answer to $\boldsymbol{Q}$ is 'yes'
P		evaluates monotone Boolean circuits C
coNP		checks CNF satisfiability

Naïve idea: classify $\boldsymbol{q}$ according to the number of occurrences of T, F

AC^0

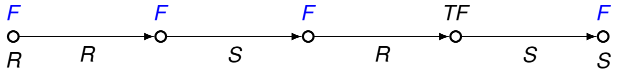
$\boldsymbol{q}$ has NO ***solitary occurrences*** of F (or T)

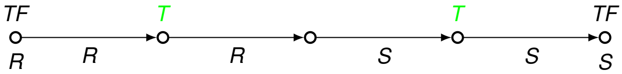
$\Downarrow$

$\boldsymbol{Q} = (\mathrm{Cov}, \boldsymbol{q})$ is in AC^0

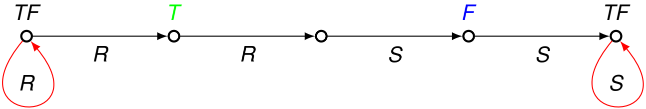
$\Downarrow$

$\boldsymbol{q}$ is a rewriting of $\boldsymbol{Q}$





necessary & sufficient condition for ***path* CQs**, but not in general:

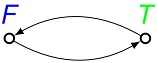


L

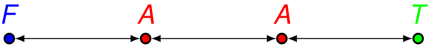
q has ***exactly one*** atom $F(x)$ and ***exactly one*** atom $T(y)$, for $x \neq y$

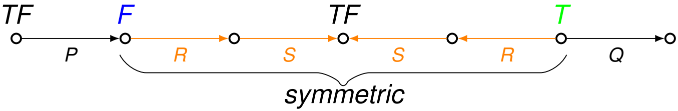
$\Downarrow$

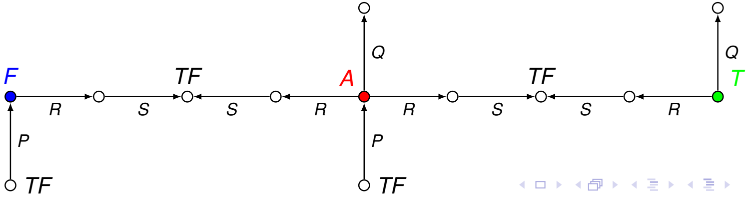
answering (Cov, ***q***) is L-hard

- $q_1 =$ 

is a *symmetric* query (L-complete)

  $\mathcal{A}_1 =$ 

- $q_2 =$ 

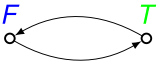
  $\mathcal{A}_2 =$ 

L

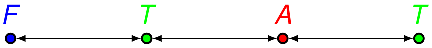
q has ***exactly one*** atom $F(x)$ and ***exactly one*** atom $T(y)$, for $x \neq y$

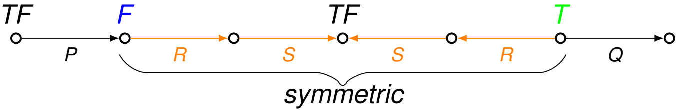
$\Downarrow$

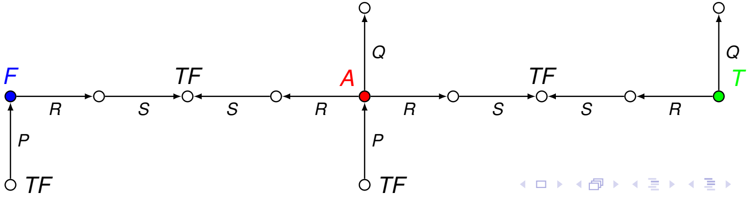
answering (Cov, ***q***) is L-hard

- $q_1 =$ 

is a *symmetric* query (L-complete)

  $\mathcal{A}_1 =$ 

- $q_2 =$ 

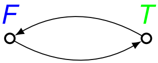
  $\mathcal{A}_2 =$ 

L

q has ***exactly one*** atom $F(x)$ and ***exactly one*** atom $T(y)$, for $x \neq y$

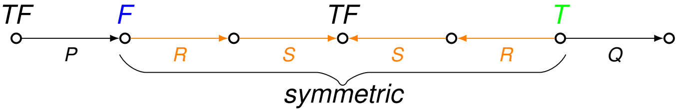
$\Downarrow$

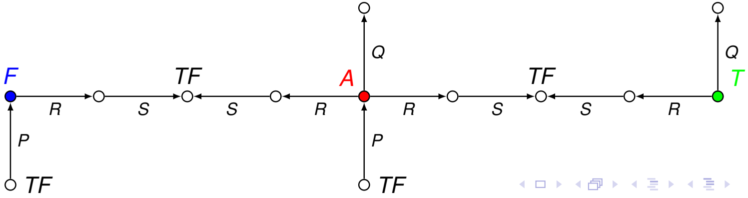
answering (Cov, $\boldsymbol{q}$) is L-hard

- $q_1 =$ 

is a *symmetric* query (L-complete)

  $\mathcal{A}_1 =$ 

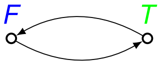
- $q_2 =$ 

  $\mathcal{A}_2 =$ 

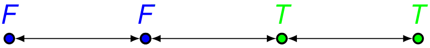
q has ***exactly one*** atom $F(x)$ and ***exactly one*** atom $T(y)$, for $x \neq y$

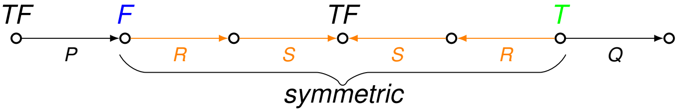
$\Downarrow$

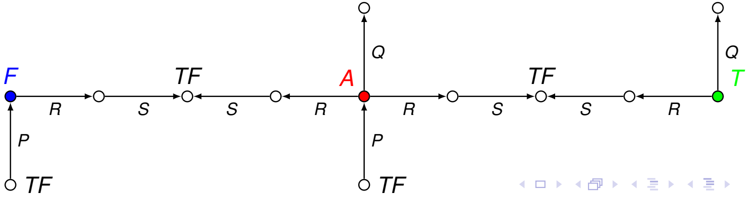
answering (Cov, ***q***) is L-hard

- $q_1 =$ 

is a *symmetric* query (L-complete)

  $\mathcal{A}_1 =$ 

- $q_2 =$ 

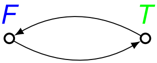
  $\mathcal{A}_2 =$ 

L

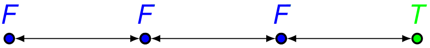
q has ***exactly one*** atom $F(x)$ and ***exactly one*** atom $T(y)$, for $x \neq y$

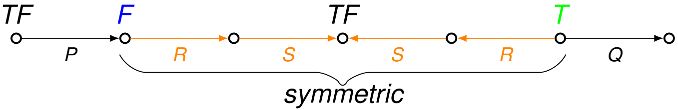
$\Downarrow$

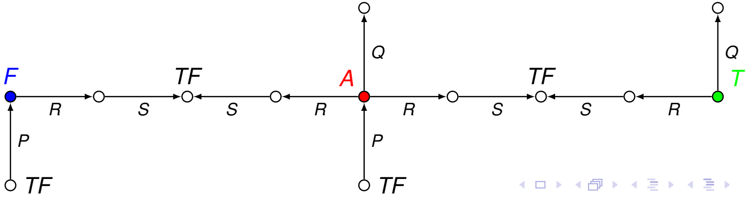
answering (Cov, ***q***) is L-hard

- $q_1 =$ 

is a *symmetric* query (L-complete)

  $\mathcal{A}_1 =$ 

- $q_2 =$ 

  $\mathcal{A}_2 =$ 

NL

path CQs

$+$

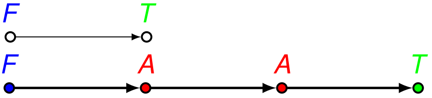
at least one solitary $F(x)$ and ***at least one solitary*** $T(y)$

$\Downarrow$

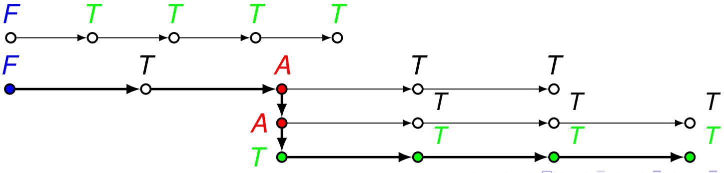
answering (Cov, $\boldsymbol{q}$) is NL-hard

Examples of NL-*complete* queries:

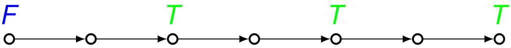
- $q_1 =$ / $\mathcal{A}_1 =$



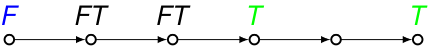
- $q_2 =$ / $\mathcal{A}_2 =$

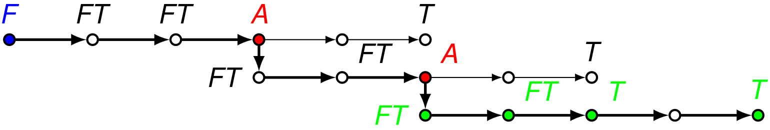


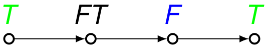
NL-complete queries

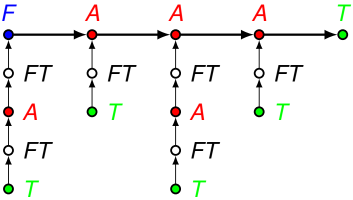
- $q_3 =$ 

is similar to q_2

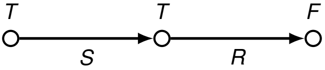
- $q_4 =$ 

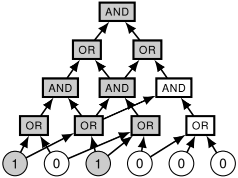
  $\mathcal{A}_4 =$ 

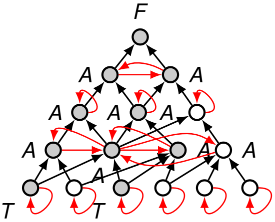
- $q_5 =$ 

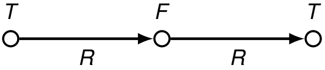
  $\mathcal{A}_5 =$ 

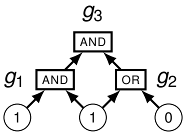
P-hardness via monotone circuits

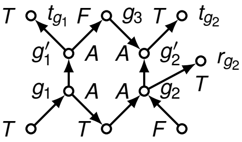
- $\boldsymbol{q}_1 =$ 

$\mathfrak{C}_1$: 

$\mathcal{A}^{\alpha}_{\mathfrak{C}_1}$: 

- $\boldsymbol{q}_2 =$ 

$\mathfrak{C}_2$: 

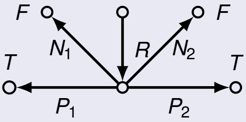
$\mathcal{A}^{\alpha}_{\mathfrak{C}_2}$: 

We can see that $\mathfrak{C}(\vec{\alpha}) = 1$ iff Cov, $\mathcal{A}^{\alpha}_{\mathfrak{C}} \models \boldsymbol{q}$.

coNP-hardness

Schaerf's (1993) example

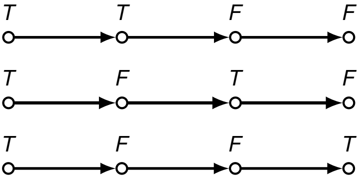
$(\mathrm{Cov}_\top, \boldsymbol{q})$ with $\boldsymbol{q}$ on the right is coNP-complete



Conjecture

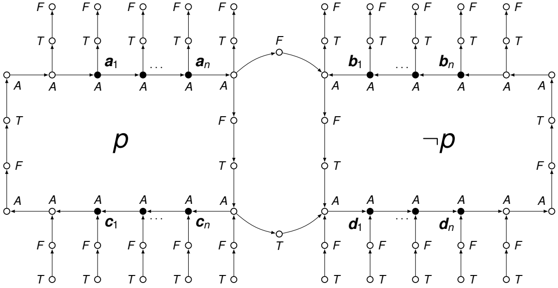
For (tree-shaped) $\boldsymbol{q}$ with at least two solitary F and two solitary T, $(\mathrm{Cov}_A, \boldsymbol{q})$ is coNP-hard

This is so for all 2-2 path-CQ of length 4

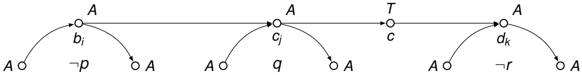


Proving coNP-hardness by reduction to 3CNF

Gadget for p and $\neg p$ in a 3CNF with n clauses:



Gadget for one clause $c = (p \vee q \vee \neg r)$:



The resulting ABox $\mathcal{A}_\psi$ is such that ψ is satisfiable iff $\text{Cov}_A, \mathcal{A}_\psi \not\models \boldsymbol{q}$

Conclusion

- We have found quite a few syntactic and semantic sufficient and/or necessary conditions for OMQ (Cov; q) to be in this or that complexity class
- We are still aiming at a general complete classification, but the connections to other notoriously hard problems indicate that achieving this aim will be difficult

Thank you for attention!