Ontology-Based Query Answering: Advanced Topics
Outline

Knowledge graphs and ontologies

Automated reasoning

Ontology-based query answering: The basics

Ontology-based query answering: Advanced topics
  Inconsistency handling
  Reasoning on annotated knowledge graphs
  Existential rules as query language
References

- Neil Immerman: Descriptive complexity.
- Leonid Libkin: Elements of Finite Model Theory.
- Sebastian Rudolph, Michal Thomazo: Characterization of the Expressivity of Existential Rule Queries. IJCAI 2015: 3103-3109
Inconsistency Handling
Handling Inconsistent Data

In real world data often contains errors

- human errors
- automatic extraction
- outdated information

Likely to be inconsistent with the ontology (today: focus on the case where the ontology is assumed reliable)

Standard semantics: everything is entailed from an inconsistent knowledge base!
Handling Inconsistent Data

In real world data often contains errors
  ➤ human errors
  ➤ automatic extraction
  ➤ outdated information

Likely to be inconsistent with the ontology (today: focus on the case where the ontology is assumed reliable)

Standard semantics: everything is entailed from an inconsistent knowledge base!

It is not always possible to resolve the inconsistencies (lack of information, time, permission...)

Alternative semantics: meaningful answers to queries despite inconsistencies
Example

\[ T = \{ \text{AP} \sqsubseteq \text{Prof}, \text{FP} \sqsubseteq \text{Prof}, \text{AP} \sqsubseteq \neg \text{FP} \} \]
\[ A = \{ \text{AP}(\text{ann}), \text{FP}(\text{ann}), \text{Postdoc}(\text{alex}) \} \]

Which assertions would be reasonable to infer?
Inconsistency-Tolerant Semantics

Many inconsistency-tolerant semantics have been proposed.

A semantics $S$ associates a set of answers to every KB and query:
- if the KB is satisfiable, should return certain answers
- for unsatisfiable KBs, give different answers than classical semantics

Write $\langle T, A \rangle \models_S q(\bar{a})$ if $\bar{a}$ is an answer to $q$ w.r.t. $\langle T, A \rangle$ under semantics $S$. 
Consistency Properties

A (consistent) $\mathcal{T}$-support of $q(\bar{a})$ is a subset $C \subseteq A$ such that

- $\langle \mathcal{T}, C \rangle$ is satisfiable
- $\langle \mathcal{T}, C \rangle \models q(\bar{a})$

Semantics $S$ satisfies the consistent support property if whenever $\langle \mathcal{T}, A \rangle \models_S q(\bar{a})$, there exists a $\mathcal{T}$-support $C \subseteq A$ of $q(\bar{a})$

- consistent explanation/justification for the query result
Consistency Properties

A (consistent) $\mathcal{T}$-support of $q(\bar{a})$ is a subset $C \subseteq \mathcal{A}$ such that

- $\langle \mathcal{T}, C \rangle$ is satisfiable
- $\langle \mathcal{T}, C \rangle \models q(\bar{a})$

Semantics $S$ satisfies the **consistent support property** if whenever $\langle \mathcal{T}, \mathcal{A} \rangle \models_S q(\bar{a})$, there exists a $\mathcal{T}$-support $C \subseteq \mathcal{A}$ of $q(\bar{a})$

- consistent explanation/justification for the query result

Semantics $S$ satisfies the **consistent results property** if for every KB $\langle \mathcal{T}, \mathcal{A} \rangle$, there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $\langle \mathcal{T}, \mathcal{A} \rangle \models_S q(\bar{a})$

- set of query results is jointly consistent with the ontology
- safe to combine query results
Comparing Semantics

Given two semantics $S$ and $S'$

- $S'$ is an under-approximation (or sound approximation) of $S$ if $\langle T, A \rangle \models_{S'} q(\bar{a})$ implies $\langle T, A \rangle \models_{S} q(\bar{a})$

- $S'$ is an over-approximation (or complete approximation) of $S$ if $\langle T, A \rangle \models_{S} q(\bar{a})$ implies $\langle T, A \rangle \models_{S'} q(\bar{a})$
Repairs

Many semantics are based upon the notion of repair: inclusion-maximal subset of the data consistent with the ontology.

Possible worlds, different ways of achieving consistency while retaining as much of the original data as possible.

<table>
<thead>
<tr>
<th>TBox</th>
<th>ABox</th>
<th>Repair</th>
<th>R₁</th>
<th>Repair</th>
<th>R₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>AProf ⊆ Prof</td>
<td>AProf(ann)</td>
<td>AProf(ann)</td>
<td>Postdoc(alex)</td>
<td>FProf(ann)</td>
<td>Postdoc(alex)</td>
</tr>
</tbody>
</table>
**Plausible Answers: AR Semantics**

AR (ABox Repair) answers: hold no matter which repair is chosen

\[ \langle \mathcal{T}, \mathcal{A} \rangle \models_{AR} q(\vec{a}) \iff \langle \mathcal{T}, \mathcal{R} \rangle \models q(\vec{a}) \] for every repair \( \mathcal{R} \)

**TBox**
- AProf \sqsubseteq Prof
- FProf \sqsubseteq Prof
- AProf \sqsubseteq \neg FProf

**ABox**
- AProf(ann)
- FProf(ann)
- Postdoc(alex)

**Repair**
- \( \mathcal{R}_1 \)
- AProf(ann)
- Postdoc(alex)
- \( \mathcal{R}_2 \)
- FProf(ann)
- Postdoc(alex)

**Consequences**
- Consequences(\( \mathcal{R}_1 \))
- Prof(ann)
- Consequences(\( \mathcal{R}_2 \))
Surest Answers: IAR Semantics

IAR (Intersection AR) answers: hold in the repairs intersection

\[ \langle \mathcal{T}, \mathcal{A} \rangle \models_{IAR} q(\bar{a}) \iff \langle \mathcal{T}, \mathcal{R}^\cap \rangle \models q(\bar{a}) \text{ with } \mathcal{R}^\cap \text{ repairs intersection} \]

### TBox
- AProf ⊆ Prof
- FProf ⊆ Prof
- AProf ⊆ ¬FProf

### ABox
- AProf(ann)
- FProf(ann)
- Postdoc(alex)

### Repair

#### Repair \( \mathcal{R}_1 \)
- AProf(ann)
- Postdoc(alex)

#### Repair \( \mathcal{R}_2 \)
- FProf(ann)
- Postdoc(alex)

### Consequences

#### Consequences(\( \mathcal{R}_1 \))
- AProf(ann)

#### Consequences(\( \mathcal{R}^\cap \))
- Postdoc(alex)

#### Consequences(\( \mathcal{R}_2 \))
Possible Answers: Brave Semantics

Brave answers: hold in some repair

\[ \langle T, A \rangle \models_{\text{brave}} q(\bar{a}) \iff \langle T, \mathcal{R} \rangle \models q(\bar{a}) \text{ for some } \mathcal{R} \]

TBox \quad T

\begin{align*}
\text{AP} & \sqsubseteq \text{Prof} \\
\text{FP} & \sqsubseteq \text{Prof} \\
\text{AP} & \sqsubseteq \lnot \text{FP}
\end{align*}

ABox \quad A

\begin{align*}
\text{AP}(\text{ann}) \\
\text{FP}(\text{ann}) \\
\text{Postdoc}(\text{alex})
\end{align*}

Repair \quad \mathcal{R}_1

\begin{align*}
\text{AP}(\text{ann}) \\
\text{Postdoc}(\text{alex})
\end{align*}

Repair \quad \mathcal{R}_2

\begin{align*}
\text{FP}(\text{ann}) \\
\text{Postdoc}(\text{alex})
\end{align*}
AR, IAR and Brave Semantics

Which consistency properties are satisfied by AR, IAR, brave?

How do the three semantics compare?
AR, IAR and Brave Semantics

- AR is the most well-known and accepted semantics
  - cautious reasoning used in many area (belief revision...)
  - consistent query answering in databases
- but AR is usually **intractable** (coNP-complete in data complexity for DL-Lite and $\mathcal{EL}$)
- IAR and brave are under- and over-approximations of AR
  - IAR most cautious: disregard all facts involved in some contradiction
  - brave least cautious: all answers supported by some consistent set of facts
- IAR and brave are tractable for DL-Lite
Some Other Inconsistency-Tolerant Semantics

- \textit{k-support semantics}
  - fine-grained under-approximation of AR
  - \( \langle \mathcal{T}, \mathcal{A} \rangle \models_{k-sup} q(\bar{a}) \) iff there exist \( C_1, \ldots, C_k \) \( \mathcal{T} \)-supports of \( q(\bar{a}) \) such that every repair contains at least one of the \( C_i \)
  - \( 1 \)-support = IAR
  - \( \langle \mathcal{T}, \mathcal{A} \rangle \models_{k-sup} q(\bar{a}) \Rightarrow \langle \mathcal{T}, \mathcal{A} \rangle \models_{k+1-sup} q(\bar{a}) \)
  - \( \langle \mathcal{T}, \mathcal{A} \rangle \models_{AR} q(\bar{a}) \Leftrightarrow \exists k \geq 1, \langle \mathcal{T}, \mathcal{A} \rangle \models_{k-sup} q(\bar{a}) \)
Some Other Inconsistency-Tolerant Semantics

- **$k$-support semantics**
  - fine-grained under-approximation of AR
  - $\langle \mathcal{T}, \mathcal{A} \rangle \models_{k-sup} q(\bar{a})$ iff there exist $C_1, \ldots, C_k$ $\mathcal{T}$-supports of $q(\bar{a})$ such that every repair contains at least one of the $C_i$
  - $1$-support $=$ IAR
  - $\langle \mathcal{T}, \mathcal{A} \rangle \models_{k-sup} q(\bar{a}) \Rightarrow \langle \mathcal{T}, \mathcal{A} \rangle \models_{k+1-sup} q(\bar{a})$
  - $\langle \mathcal{T}, \mathcal{A} \rangle \models_{AR} q(\bar{a}) \Leftrightarrow \exists k \geq 1, \langle \mathcal{T}, \mathcal{A} \rangle \models_{k-sup} q(\bar{a})$

- **$k$-defeater semantics**
  - fine-grained over-approximation of AR
  - $\langle \mathcal{T}, \mathcal{A} \rangle \models_{k-def} q(\bar{a})$ iff there does not exist a $\mathcal{T}$-consistent $S \subseteq \mathcal{A}$ such that $|S| \leq k$ and $\langle \mathcal{T}, S \cup C \rangle \models \bot$ for every minimal $\mathcal{T}$-support $C$ of $q(\bar{a})$
  - $0$-defeater $=$ brave
  - $\langle \mathcal{T}, \mathcal{A} \rangle \models_{k+1-def} q(\bar{a}) \Rightarrow \langle \mathcal{T}, \mathcal{A} \rangle \models_{k-def} q(\bar{a})$
  - for every KB, there exists $k$ such that $\langle \mathcal{T}, \mathcal{A} \rangle \models_{AR} q(\bar{a}) \Leftrightarrow \langle \mathcal{T}, \mathcal{A} \rangle \models_{k-def} q(\bar{a})$
Some Other Inconsistency-Tolerant Semantics

- ICR (Intersection Closed Repairs) semantics
  - under-approximation of AR and over-approximation of IAR
  - intersects the closures of the repairs (closure of $\mathcal{R} = \text{set of assertions entailed from } \langle \mathcal{T}, \mathcal{R} \rangle$)
  - same as AR for queries without quantifier
Some Other Inconsistency-Tolerant Semantics

- **ICR (Intersection Closed Repairs) semantics**
  - under-approximation of AR and over-approximation of IAR
  - intersects the closures of the repairs (closure of $\mathcal{R} =$ set of assertions entailed from $\langle \mathcal{T}, \mathcal{R} \rangle$)
  - same as AR for queries without quantifier

- **CAR and ICAR semantics**
  - define semantics that are (almost) syntax-independent
  - apply closure operator on original ABox
  - need alternative notion of closure for inconsistent KB: set of assertions with a $\mathcal{T}$-support in $\mathcal{A}$
  - closed ABox repairs: maximally complete standard ABox repairs with facts from the closure of $\mathcal{A}$
  - apply AR (CAR) or IAR (ICAR) using closed ABox repairs
  - do not satisfy consistent support!

$\mathcal{T} = \{ A \sqsubseteq B, C \sqsubseteq D, A \sqsubseteq \neg C \}, \ \mathcal{A} = \{ A(a), C(a) \}$,

$q = B(x) \land D(x)$
Exercise: AR, IAR, brave, $k$-supp, $k$-def, ICR ?

\[ \mathcal{T} = \{ \text{AP} \sqsubseteq \text{Prof}, \text{F} \sqsubseteq \text{Prof}, \text{Prof} \sqsubseteq \text{PhD}, \text{Postdoc} \sqsubseteq \text{PhD}, \text{PhD} \sqsubseteq \text{Person}, \exists \text{Teach} \sqsubseteq \text{Person}, \exists \text{Teach}^- \sqsubseteq \text{Course}, \text{Prof} \sqsubseteq \exists \text{WorkFor}, \text{Student} \sqsubseteq \exists \text{MemberOf}, \text{WorkFor} \sqsubseteq \text{MemberOf}, \text{AP} \sqsubseteq \neg \text{F} \sqsubseteq \neg \text{Postdoc}, \text{Student} \sqsubseteq \neg \text{Prof}, \text{Person} \sqsubseteq \neg \text{Course}, \exists \text{MemberOf}^- \sqsubseteq \neg \text{Postdoc} \} \]
Exercise: AR, IAR, brave, $k$-supp, $k$-def, ICR ?

$$\mathcal{T} = \{ \text{AP} \sqsubseteq \text{Prof}, \text{FP} \sqsubseteq \text{Prof}, \text{P} \sqsubseteq \text{PhD}, \text{Post} \sqsubseteq \text{PhD}, \text{PhD} \sqsubseteq \text{Person}, \exists \text{Teach} \sqsubseteq \text{Person}, \exists \text{Teach}^- \sqsubseteq \text{Course}, \text{P} \sqsubseteq \exists \text{WorkFor}, \text{S} \sqsubseteq \exists \text{MemberOf}, \text{W} \sqsubseteq \text{MemberOf}, \text{AP} \sqsubseteq \neg \text{FP}, \text{P} \sqsubseteq \neg \text{Post}, \text{S} \sqsubseteq \neg \text{P}, \text{P} \sqsubseteq \neg \text{C}, \exists \text{MemberOf}^- \sqsubseteq \neg \text{Post} \}$$

$$\mathcal{A}_a = \{ \text{AP}(\text{ann}), \text{FP}(\text{ann}), \text{P}(\text{ann}), \text{Teach}(\text{ann}, c_a), \text{Teach}(\text{ann}, \text{ann}) \}$$

$$q(x) = \exists y z \text{PhD}(x) \land \text{MemberOf}(x, y) \land \text{Teach}(x, z)$$
Exercise: AR, IAR, brave, $k$-supp, $k$-def, ICR?

$\mathcal{T} = \{\text{AP} \sqsubseteq \text{Prof}, \text{FProf} \sqsubseteq \text{Prof}, \text{Prof} \sqsubseteq \text{PhD}, \text{Postdoc} \sqsubseteq \text{PhD}, \text{PhD} \sqsubseteq \text{Person}, \exists \text{Teach} \sqsubseteq \text{Person}, \exists \text{Teach}^\perp \sqsubseteq \text{Course}, \text{Prof} \sqsubseteq \exists \text{WorkFor}, \text{Student} \sqsubseteq \exists \text{MemberOf}, \text{WorkFor} \sqsubseteq \text{MemberOf}, \text{AP} \sqsubseteq \neg \text{FProf}, \text{Prof} \sqsubseteq \neg \text{Postdoc}, \text{Student} \sqsubseteq \neg \text{Prof}, \text{Person} \sqsubseteq \neg \text{Course}, \exists \text{MemberOf}^\perp \sqsubseteq \neg \text{Postdoc}\}$

$\mathcal{A}_b = \{\text{AP}(bob), \text{FProf}(bob), \text{Postdoc}(bob), \text{MemberOf}(bob, dpt), \text{Teach}(bob, c_b)\}$

$q(x) = \exists y z \text{PhD}(x) \land \text{MemberOf}(x, y) \land \text{Teach}(x, z)$
Exercise: AR, IAR, brave, $k$-supp, $k$-def, ICR?

$$\mathcal{T} = \{ \text{APref} \sqsubseteq \text{Prof}, \text{FProf} \sqsubseteq \text{Prof}, \text{Prof} \sqsubseteq \text{PhD}, \text{Postdoc} \sqsubseteq \text{PhD}, \text{PhD} \sqsubseteq \text{Person, } \exists \text{Teach} \sqsubseteq \text{Person, } \exists \text{Teach}^- \sqsubseteq \text{Course}, \text{Prof} \sqsubseteq \exists \text{WorkFor}, \text{Student} \sqsubseteq \exists \text{MemberOf}, \text{WorkFor} \sqsubseteq \text{MemberOf}, \text{APref} \sqsubseteq \neg \text{FProf}, \text{Prof} \sqsubseteq \neg \text{Postdoc}, \text{Student} \sqsubseteq \neg \text{Prof}, \text{Person} \sqsubseteq \neg \text{Course, } \exists \text{MemberOf}^- \sqsubseteq \neg \text{Postdoc} \}$$

$$\mathcal{A}_c = \{ \text{APref}(\text{carl}) , \text{Teach}(\text{carl}, c_{c1}), \text{Teach}(\text{carl}, c_{c2}), \text{Teach}(c_{c1}, c_{c2}), \text{Teach}(c_{c2}, c_{c1}) \}$$

$$q(x) = \exists y z \text{PhD}(x) \land \text{MemberOf}(x, y) \land \text{Teach}(x, z)$$
Exercise: AR, IAR, brave, \( k\)-supp, \( k\)-def, ICR ?

\[ T = \{ \text{AP} \sqsubseteq \text{Prof}, \text{FP} \sqsubseteq \text{Prof}, \text{Prof} \sqsubseteq \text{PhD}, \text{Postdoc} \sqsubseteq \text{PhD}, \]
\[ \text{PhD} \sqsubseteq \text{Person}, \exists \text{Teach} \sqsubseteq \text{Person}, \exists \text{Teach}^\sim \sqsubseteq \text{Course}, \]
\[ \text{Prof} \sqsubseteq \exists \text{WorkFor}, \text{Student} \sqsubseteq \exists \text{MemberOf}, \text{WorkFor} \sqsubseteq \text{MemberOf}, \]
\[ \text{AP} \sqsubseteq \neg \text{FP}, \text{Prof} \sqsubseteq \neg \text{Postdoc}, \text{Student} \sqsubseteq \neg \text{Prof}, \]
\[ \text{Person} \sqsubseteq \neg \text{Course}, \exists \text{MemberOf}^\sim \sqsubseteq \neg \text{Postdoc} \} \]

\[ A_d = \{ \text{AP}(\text{dan}), \text{Teach}(\text{dan}, c_{d1}), \text{Teach}(\text{dan}, c_{d2}), \]
\[ \text{AP}(c_{d1}), \text{AP}(c_{d2}) \} \]

\[ q(x) = \exists yz \text{PhD}(x) \land \text{MemberOf}(x, y) \land \text{Teach}(x, z) \]
Semantics Based Upon Preferred Repairs

Idea: some repairs are more likely than others

Defined preferred repairs based on

- cardinality
- priority levels
- weights
- ...

AR/IAR/brave/... semantics based upon most preferred repairs
Some Complexity Results for DL-Lite

Results apply to all languages that satisfy

- minimal $\mathcal{T}$-supports for $q(\overline{a})$ contain at most $|q|$ assertions
- minimal $\mathcal{T}$-inconsistent subsets have cardinality at most 2
- CQ answering and satisfiability can be performed by FO rewriting (so in AC0 in data complexity)
CQ entailment under AR semantics is \textit{coNP-complete} in data complexity.

Upper bound: guess $\mathcal{R} \subseteq \mathcal{A}$ and verify that $\mathcal{R}$ is a repair and $\langle \mathcal{T}, \mathcal{R} \rangle \not\models q(\bar{a})$

\begin{itemize}
  \item $\langle \mathcal{T}, \mathcal{R} \rangle \not\models q(\bar{a})$ in AC0
  \item repair checking in PTIME?
\end{itemize}
CQ entailment under AR semantics is coNP-complete in data complexity.

Upper bound: guess $\mathcal{R} \subseteq \mathcal{A}$ and verify that $\mathcal{R}$ is a repair and $\langle \mathcal{T}, \mathcal{R} \rangle \not\models q(\bar{a})$
- $\langle \mathcal{T}, \mathcal{R} \rangle \not\models q(\bar{a})$ in AC0
- repair checking in PTIME?

Lower bound: reduction from UNSAT: $\varphi = c_1 \land \cdots \land c_m$ conjunction of clauses over variables $x_1, \ldots, x_k$, build a KB $\langle \mathcal{T}, \mathcal{A} \rangle$ and query $q(\bar{a})$ such that $\varphi$ is unsatisfiable iff $\langle \mathcal{T}, \mathcal{A} \rangle \models_{AR} q(\bar{a})$
Some Complexity Results for DL-Lite

CQ entailment under IAR and brave semantics is in \( \text{PTIME} \) in data complexity

Any idea of \( \text{PTIME} \) algorithms?
Some Complexity Results for DL-Lite

CQ entailment under IAR and brave semantics is in $\text{PTime}$ in data complexity

Any idea of $\text{PTime}$ algorithms?

Actually, CQ entailment under IAR and brave semantics is in $\text{AC0}$ in data complexity

Can use FO-rewriting to compute IAR and brave answers
FO Rewriting for IAR Semantics

Idea: modify UCQ-rewriting to ensure ABox assertions matching CQs are not involved in any contradictions

\[ T = \{ \text{AP} \sqsubseteq \text{Prof}, \text{FProf} \sqsubseteq \text{Prof}, \text{Prof} \sqsubseteq \text{PhD}, \text{Postdoc} \sqsubseteq \text{PhD}, \text{PhD} \sqsubseteq \text{Person}, \exists \text{Teach} \sqsubseteq \text{Person}, \exists \text{Teach}^\neg \sqsubseteq \text{Course}, \text{Prof} \sqsubseteq \exists \text{WorkFor}, \text{Student} \sqsubseteq \exists \text{MemberOf}, \text{WorkFor} \sqsubseteq \text{MemberOf}, \text{AP} \sqsubseteq \neg \text{FProf}, \text{Prof} \sqsubseteq \neg \text{Postdoc}, \text{Student} \sqsubseteq \neg \text{Prof}, \text{Person} \sqsubseteq \neg \text{Course}, \exists \text{MemberOf}^\neg \sqsubseteq \neg \text{Postdoc} \} \]

\[ q_1(x) = \text{PhD}(x) \]

\[ q_2(x) = \exists y \text{MemberOf}(x, y) \]

\[ q_3(x) = \exists y \text{Prof}(x) \land \text{Teach}(x, y) \]
FO Rewriting for Brave Semantics

Idea: modify UCQ-rewriting to ensure each CQ can only match \( \mathcal{T} \)-consistent subsets of ABox

\[ \mathcal{T} = \{ \text{AProf} \sqsubseteq \text{Prof}, \text{FProf} \sqsubseteq \text{Prof}, \text{Prof} \sqsubseteq \text{PhD}, \text{Postdoc} \sqsubseteq \text{PhD}, \text{PhD} \sqsubseteq \text{Person}, \exists \text{Teach} \sqsubseteq \text{Person}, \exists \text{Teach}^\neg \sqsubseteq \text{Course}, \text{Prof} \sqsubseteq \exists \text{WorkFor}, \text{Student} \sqsubseteq \exists \text{MemberOf}, \text{WorkFor} \sqsubseteq \text{MemberOf}, \text{AProf} \sqsubseteq \neg \text{FProf}, \text{Prof} \sqsubseteq \neg \text{Postdoc}, \text{Student} \sqsubseteq \neg \text{Prof}, \text{Person} \sqsubseteq \neg \text{Course}, \exists \text{MemberOf}^\neg \sqsubseteq \neg \text{Postdoc} \} \]

\[ q_1(x) = \exists y \text{PhD}(x) \land \text{MemberOf}(x, y) \]

\[ q_2(x) = \exists y \text{Prof}(x) \land \text{Teach}(x, y) \]
More FO Rewritings

$k$-support and $k$-defeater semantics are also FO-rewritable. Any idea for the general shape of the rewritings?
### Complexity Picture for DL-Lite

<table>
<thead>
<tr>
<th></th>
<th>Data Complexity</th>
<th>Combined Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CQs</td>
<td>IQs</td>
</tr>
<tr>
<td>classical</td>
<td>in AC0</td>
<td>in AC0</td>
</tr>
<tr>
<td>AR</td>
<td>coNP-co</td>
<td>coNP-co</td>
</tr>
<tr>
<td>IAR</td>
<td>in AC0</td>
<td>in AC0</td>
</tr>
<tr>
<td>brave</td>
<td>in AC0</td>
<td>in AC0</td>
</tr>
<tr>
<td>k-support</td>
<td>in AC0</td>
<td>in AC0</td>
</tr>
<tr>
<td>k-defeater</td>
<td>in AC0</td>
<td>in AC0</td>
</tr>
<tr>
<td>ICR</td>
<td>coNP-co</td>
<td>coNP-co</td>
</tr>
<tr>
<td>CAR</td>
<td>coNP-co</td>
<td>in AC0</td>
</tr>
<tr>
<td>ICAR</td>
<td>in AC0</td>
<td>in AC0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>CQs</th>
<th>IQs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NP-co</td>
<td>in PTIME</td>
</tr>
<tr>
<td>AR</td>
<td>Π₂-co</td>
<td>coNP-co</td>
</tr>
<tr>
<td>IAR</td>
<td>NP-co</td>
<td>in PTIME</td>
</tr>
<tr>
<td>brave</td>
<td>NP-co</td>
<td>in PTIME</td>
</tr>
<tr>
<td>k-support</td>
<td>NP-co</td>
<td>in PTIME</td>
</tr>
<tr>
<td>k-defeater</td>
<td>NP-co</td>
<td>in PTIME</td>
</tr>
<tr>
<td>ICR</td>
<td>Δ₂[Olog(n)]-co</td>
<td>coNP-co</td>
</tr>
<tr>
<td>CAR</td>
<td>Π₂-co</td>
<td>in PTIME</td>
</tr>
<tr>
<td>ICAR</td>
<td>NP-co</td>
<td>in PTIME</td>
</tr>
</tbody>
</table>

Note on AC0 cases:

- FO-rewritings, but rewritings may be huge and not efficiently evaluated over databases
- alternative PTIME algorithms based on supports and conflicts may be more efficient in practice
A Practical Approach to Query Answering under AR Semantics

- Exploit tractable approximations:
  \( \text{IAR} \implies \text{AR} \) and not brave \( \implies \) not AR

- for remaining cases (brave and not IAR): reduce AR entailment to SAT and use a SAT solver
Research Problems on Inconsistency Handling

- Inconsistency-tolerant semantics
- Practical algorithms, implementations
- Beyond DL-Lite: unbounded size of query supports and conflicts
- Explanations of query results
- Improving data quality, helping user to resolve inconsistencies
Reasoning on Annotated Knowledge Graphs
Attributed Description Logics

Bridging the gap between knowledge graphs and description logics

**Knowledge graphs**: directed labelled graphs with annotations

![Graph Example]

- Unlimited number of annotations
- No relationship between spouse(gabor, ryan) and spouse(ryan, gabor)

Two Wikidata statements (on Zsa Zsa Gabor and Jack Ryan pages):

- Zsa Zsa Gabor
  - **start time**: 1975
  - **end time**: 1976

- Jack Ryan
  - **start time**: 21 January 1975
  - **end time**: 24 August 1976
Attributed Description Logics
Bridging the gap between knowledge graphs and description logics

Description logic ontologies: assertions (\sim directed labelled graphs)
+ relationships between concepts and roles

\{ \text{spouse}(\text{gabor, ryan}), \text{spouse} \preceq \text{spouse}^- \} \models \text{spouse}(\text{ryan, gabor})

\rightarrow \text{Allow for reasoning}
\rightarrow \text{No annotations}
Attributed Description Logics
Bridging the gap between knowledge graphs and description logics

Attributed description logic ontologies: annotated assertions + relationships between annotated concepts and roles

\{
  \text{spouse} (\text{gabor}, \text{ryan})@[start : 1975, end : 1976],
  \text{spouse}@[X] \sqsubseteq \text{spouse}^-@[X]
\}

\models \text{spouse}(\text{ryan}, \text{gabor})@[start : 1975, end : 1976]
Attributed Description Logics

Annotations

Annotations with specifiers \( \sim \) sets of attribute-value pairs

- set variables \( X \)
  - \( \text{spouse@}X \subseteq \text{spouse}^- @X \)
Attributed Description Logics

Annotations

Annotations with specifiers ~ sets of attribute-value pairs

- set variables $X$
  - $\text{spouse} \subseteq \text{spouse}^\neg \subseteq X$

- closed specifiers $[a_1 : v_1, \ldots, a_n : v_n]$ (specific annotation sets)
  - $\text{spouse}(\text{gabor}, \text{ryan})@[\text{start} : 1975, \text{end} : 1976]$

where

- $a_i$ is an individual name,
- $v_i$ is either an individual name, an object variable, or an expression of the form $X.a$ that refers to the set of all values of attribute $a$ in $X$
Attributed Description Logics

Annotations

Annotations with specifiers ~ sets of attribute-value pairs

- set variables $X$
  - spouse@$X$ ⊆ spouse¬@$X$

- closed specifiers $[a_1 : v_1, \ldots, a_n : v_n]$ (specific annotation sets)
  - spouse(gabor, ryan)@[start : 1975, end : 1976]

- open specifiers $[a_1 : v_1, \ldots, a_n : v_n]$ (lower bounds)
  - $X : [\text{classification} : \text{public}]$ spouse@$X$ ⊆ spouse¬@$X$

where

- $a_i$ is an individual name,
- $v_i$ is either an individual name, an object variable, or an expression of the form $X.a$ that refers to the set of all values of attribute $a$ in $X$
Attributed Description Logics

Annotations

Annotations with specifiers \sim sets of attribute-value pairs
Attributed Description Logics

Annotations

Annotations with specifiers \sim sets of attribute-value pairs

\begin{align*}
\text{spouse}(\text{taylor}, \text{burton})@\text{[start} : 1964] ,
\text{spouse}(\text{taylor}, \text{burton})@\text{[start} : 1975] ,
\end{align*}
Attributed Description Logics

Annotations

Annotations with **specifiers** $\sim$ sets of **attribute-value** pairs

- spouse(taylor, burton)@[start : 1964],
- spouse(taylor, burton)@[start : 1975],
Attributed Description Logics

Annotations

Annotations with specifiers $\sim$ sets of attribute-value pairs

\[
\begin{align*}
\text{spouse(taylor, burton)} & \text{@[start : 1964]}, \\
\text{spouse(taylor, burton)} & \text{@[start : 1975]}, \\
\text{spouse(taylor, todd)} & \text{@[classification : public, start : 1957, end : 1958, witness : fisher, witness : moreno, source : src1]}, \\
X & \text{[classification : public, start : } x, \text{end : } y] \ (\text{spouse@X } \sqsubseteq \\
\text{spouse}^- & \text{@[classification : public, start : } x, \text{end : } y, \text{witness : } X.\text{witness]})
\end{align*}
\]
Attributed Description Logics

Annotations

Annotations with specifiers $\sim$ sets of attribute-value pairs

\[
\text{spouse(taylor, burton)@[start : 1964],}
\text{spouse(taylor, burton)@[start : 1975],}
\text{spouse(taylor, todd)@[classification : public, start : 1957, end : 1958,}
\hspace{1cm} \text{witness : fisher, witness : moreno, source : src1],}
\text{X : [classification : public, start : x, end : y] (spouse@X \sqsubseteq}
\hspace{1cm} \text{spouse@@[classification : public, start : x, end : y, witness : X.witness])}
\]

\[
\models \text{spouse(todd, taylor)@[classification : public, start : 1957, end : 1958,}
\hspace{1cm} \text{witness : fisher, witness : moreno]}
\]
Attributed DL-Lite$_R$: satisfiability and CQ entailment is \textsc{PSPACE}-complete in combined complexity (standard DL-Lite$_R$: satisfiability in \textsc{PTime} and CQ entailment \textsc{NP}-complete)

Can achieve tractability by imposing (strong) restrictions on the ontology

$\rightarrow$ Research problems

$\quad$ find interesting fragments with good computational properties

$\quad$ practical algorithms, implementation
Existential Rules as Query Language
Recall of the Formal Problem

Given a database $D$, a set of existential rules $\mathcal{R}$ and a Boolean conjunctive query $q$, does it hold that:

$$D, \mathcal{R} \models q$$
Recall of the Formal Problem

Given a database $D$, a set of existential rules $\mathcal{R}$ and a Boolean conjunctive query $q$, does it hold that:

$$D, \mathcal{R} \models q$$

Two visions of that problem:

- $(D, \mathcal{R})$ is a so-called knowledge base, and we want to query it using (here conjunctive) queries; this is a classical view in the knowledge representation field.
Recall of the Formal Problem

Given a database $D$, a set of existential rules $\mathcal{R}$ and a Boolean conjunctive query $q$, does it hold that:

$$D, \mathcal{R} \models q$$

Two visions of that problem:

- $(D, \mathcal{R})$ is a so-called knowledge base, and we want to query it using (here conjunctive) queries; this is a classical view in the knowledge representation field
- $D$ is a database and we express queries using $\mathcal{R}, q$; this is a classical view in the database field.
Query Languages

- here, Boolean queries
- we have already mentionned several query languages: conjunctive queries, union of conjunctive queries, first-order queries, regular path queries, Datalog...
- how to compare the expressivity of different query languages?
“Recall”: Isomorphism

Let $A$ and $B$ be two set of atoms.

- a homomorphism from $A$ to $B$ is a mapping $\pi$ from $\text{terms}(A)$ to $\text{terms}(B)$ such that if $p(x_1, \ldots, x_k) \in A$, then $p(\pi(x_1), \ldots, \pi(x_k)) \in B$

- an isomorphism from $A$ to $B$ is a bijective homomorphism $\pi$ for which $\pi^{-1}$ is also a homomorphism.
Abstract Definition of Boolean Queries

Let $P_e$ be a finite set of predicates (called extensional predicates), and let $\Delta$ be a set of domain elements.

- a database on $P_e$ is a finite set of atoms of predicate in $P_e$ with terms in $\Delta$
- a Boolean query (on $P_e$) is a set of databases on $P_e$ that is closed under homomorphism.

In particular:
- no constants!
Existential Rules as A Query Language

Consider a finite set of predicates \( \mathcal{P}_i \), disjoint from \( \mathcal{P}_e \). An existential rule query is a set of existential rules such that:

- any atom that appears in a rule head has a predicate that belongs to \( \mathcal{P}_i \);
- there exists a special predicate \textit{goal}.

A database \( D \) belongs to an existential rule query \( \mathcal{R} \) if and only if:
\[ D, \mathcal{R} \models \text{goal} \]
Consider a finite set of predicates $\mathcal{P}_i$, disjoint from $\mathcal{P}_e$. An existential rule query is a set of existential rules such that:

- any atom that appears in a rule head has a predicate that belongs to $\mathcal{P}_i$;
- there exists a special predicate $\text{goal}$.

A database $D$ belongs to an existential rule query $\mathcal{R}$ if and only if: $D, \mathcal{R} \models \text{goal}$

A Datalog query is an existential rule query where no rule contains an existentially quantified variable.
Describing Expressivity of a Query Language

- the complexity of a query language gives *some* information about which query can be expressed... or maybe more accurately, about which query *cannot* be expressed in the query language
- however, it does not tell the whole story.
Some Examples of Queries

Let $\mathcal{P}_e = \{ E, A, B \}$, where $E$ is a binary predicate, $A$ and $B$ are unary predicates. $E$ is used to represent the edges of non-oriented graph (possibly with self-loops). Which query language among UCQ, Datalog, existential rules express the following queries:

- there exists in the represented graph two nodes (possibly the same) linked by an edge, where one is of type $A$ and the other is of type $B$
- there exists two nodes $x$ and $y$, s.t. $x$ is of type $A$, $y$ is of type $B$, and there is a path from $x$ to $y$
- the graph represented by $E$ is 3-colorable
- the graph represented by $E$ is not 3-colorable
Some Examples of Queries

First Query

There exists in the represented graph two nodes (possibly the same) linked by an edge, where one is of type \( A \) and the other is of type \( B \).

This is actually expressible by a conjunctive query:

\[
q() : \neg \exists x \ \exists y \ E(x, y) \land A(x) \land A(y)
\]

Hence, it is also expressible by a Datalog query and an existential rule query.
Some Examples of Queries

Second Query
There exists two nodes $x$ and $y$, s.t. $x$ is of type $A$, $y$ is of type $B$, and there is a path from $x$ to $y$

We notice that:

- the query is not expressible in first-order logic, hence in UCQs: formal proof not covered in this course (check the references if you want more details).
- but there is a Datalog program computing it:
  
  $A(x) \rightarrow R_A(x)$
  $R_A(x) \land E(x, y) \rightarrow R_A(y)$
  $R_A(y) \land E(x, y) \rightarrow R_A(x)$
  $R_A(x) \land B(x) \rightarrow \text{goal}$

Hence, it is also expressible by an existential rule query.
Some Examples of Queries

Third Query
The graph represented by $E$ is 3-colorable

- this query is NP-complete: hence it cannot be expressed by UCQs or Datalog, as all queries expressed are computable in polynomial time
- what about existential rule queries?

Closure under Homomorphism
A query is said closed under homomorphism if it holds that if $D \models q$ and there is a homomorphism from $D$ to $D'$, then $D' \models q$.

Is 3-colorability closed under homomorphism? No
Are existential rule queries closed under homomorphism? Yes

Hence 3-colorability is not expressible by existential rule queries.
Some Examples of Queries

Third Query
The graph represented by $E$ is 3-colorable

- this query is NP-complete: hence it cannot be expressed by UCQs or Datalog, as all queries expressed are computable in polynomial time
- what about existential rule queries?

Closure under Homomorphism

A query is said closed under homomorphism is if it holds that if $D \in q$ and there is a homomorphism from $D$ to $D'$, then $D' \in q$.

- is 3-colorability closed under homomorphism?
Some Examples of Queries

Third Query

The graph represented by $E$ is 3-colorable

- this query is NP-complete: hence it cannot be expressed by UCQs or Datalog, as all queries expressed are computable in polynomial time
- what about existential rule queries?

Closure under Homomorphism

A query is said closed under homomorphism is if it holds that if $D \in q$ and there is a homomorphism from $D$ to $D'$, then $D' \in q$.

- is 3-colorability closed under homomorphism? No
Some Examples of Queries

Third Query
The graph represented by $E$ is 3-colorable

- this query is NP-complete: hence it cannot be expressed by UCQs or Datalog, as all queries expressed are computable in polynomial time
- what about existential rule queries?

Closure under Homomorphism
A query is said closed under homomorphism is if it holds that if $D \in q$ and there is a homomorphism from $D$ to $D'$, then $D' \in q$.

- is 3-colorability closed under homomorphism? No
- are existential rule queries closed under homomorphism?
Some Examples of Queries

Third Query
The graph represented by $E$ is 3-colorable

- this query is NP-complete: hence it cannot be expressed by UCQs or Datalog, as all queries expressed are computable in polynomial time
- what about existential rule queries?

Closure under Homomorphism
A query is said closed under homomorphism is if it holds that if $D \in q$ and there is a homomorphism from $D$ to $D'$, then $D' \in q$.

- is 3-colorability closed under homomorphism? No
- are existential rule queries closed under homomorphism? Yes
Some Examples of Queries

Third Query
The graph represented by $E$ is 3-colorable

- this query is NP-complete: hence it cannot be expressed by UCQs or Datalog, as all queries expressed are computable in polynomial time
- what about existential rule queries?

Closure under Homomorphism
A query is said closed under homomorphism if it holds that if $D \in q$ and there is a homomorphism from $D$ to $D'$, then $D' \in q$.

- is 3-colorability closed under homomorphism? No
- are existential rule queries closed under homomorphism? Yes

Hence 3-colorability is not expressible by existential rule queries.
Some Examples of Queries

Third Query

The graph represented by $E$ is not 3-colorable

- as before, this query is not expressible in UCQs or Datalog
- but it is now closed under homomorphism, so the preceding argument for existential rules does not apply

Any idea?
Complexity vs Expressivity

Let $C$ be a complexity class.

- there may be some queries computable in $C$ that are not expressible in a $C$-complete query language
- what is known about the expressivity of query languages?
Let $C$ be a complexity class.

- there may be some queries computable in $C$ that are not expressible in a $C$-complete query language
- what is known about the expressivity of query languages?

We’ll cover:

- a classical results for polynomial queries on ordered databases;
- the expressivity of existential rule queries.
Adding input negation to Datalog allows to have negative extensional literals in rule bodies:

- \( A(x), \neg C(x) \rightarrow R_A(x) \)
- \( R_A(x), E(x, y), \neg C(y) \rightarrow R_A(y) \)
- \( R_A(y), E(x, y), \neg C(x) \rightarrow R_A(x) \)
- \( R_A(x), B(x) \rightarrow \text{goal} \)

where \( A, B, C \) and \( E \) are extensional predicates.
Datalog with Linear Order

A linear order on the domain elements is defined using three predicates:

- `first`, which identifies a single domain element
- `last`, which identifies a single domain element
- `succ`, such that each element that is not the first has a unique antecedent by succ, and each element that is not the last has a unique successor by succ.
A linear order on the domain elements is defined using three predicates:

- `first`, which identifies a single domain element
- `last`, which identifies a single domain element
- `succ`, such that each element that is not the first has a unique antecedent by succ, and each element that is not the last has a unique successor by succ.

A query is expressible in Datalog with linear order if there exists a Datalog program possibly using `first`, `last`, and `succ`, such that whichever linear order over the domain elements of D is chosen, is such that $D, \Pi \models \text{goal}$ if and only if $D$ belongs to the query.
Show that the following query:

For all domain element $x$, it holds that $A(x)$

is not expressible in Datalog, but is expressible in Datalog with a linear order.
High Level View of the Construction

We now sketch the proof of the following result:

Any query computable in \( \text{PTime} \) can be computed by a Datalog program with input negation and access to a linear order on the domain elements.
High Level View of the Construction

We now sketch the proof of the following result:

Any query computable in $\mathsf{PTime}$ can be computed by a Datalog program with input negation and access to a linear order on the domain elements.

We first have to fix the encodings we use:

1. choose an encoding of the database on a Turing machine tape;
2. choose an encoding of a tape in a relational structure;
High Level View of the Construction

We now sketch the proof of the following result:

Any query computable in $PTime$ can be computed by a Datalog program with input negation and access to a linear order on the domain elements.

We first have to fix the encodings we use:

1. choose an encoding of the database on a Turing machine tape;
2. choose an encoding of a tape in a relational structure;

We then generate the corresponding relations and simulate the $PTime$ Turing machine:

1. generate the encoding of the tape encoding the database from the database;
2. use Datalog rules to simulate the run of the Turing machine.
Decide of an order of the relations $p_1, \ldots, p_n$

For $i$ from 1 to $n$, add the encoding of the table $p_i$:

- Add the encoding of all tuples being present in the table $p_i$
- the encoding of a tuple is the concatenation of the encodings of the individuals, separated by #
- the encoding of an individual is its rank in an arbitrary (but fixed) order of the individuals.
Encoding a Tape in a Relation

To represent the content of the tape, notice that:
- only a polynomial number of cells may potentially be used
- only a polynomial number of configurations may occur

We thus use $k$ columns to represent the address of the cell represented, and $k$ columns to represent the time stamp of the configuration.

We additionally use two positions:
- one to represent the element in the corresponding cell
- one to represent the position of the head, and the state of the Turing machine.
The Road So Far

What we have discussed:
- how to encode a database on a Turing Machine tape
- how to encode a Turing Machine tape on a relation

What is missing: how to generate the encoding and simulate the run of the Turing machine.
The Remaining Steps

Generating the Encoding:
- generate a \( succ_k \) relation encoding a lexicographic order on \( k \)-uples
- for each predicate, use the lexicographic order to iterate on \( k \)-uples and use input negation to check whether the corresponding atom is in the database
- if it is, add it to the represented tape, otherwise continue.

Simulating the Turing Machine:
- similar to what has been done in a previous course.
Expressivity of Existential Rule Queries

Any query expressible as an existential rule query has the following property:

- it is closed under homomorphism;
- it is recursively enumerable.

But are all recursively enumerable closed under homomorphism query expressible by an existential query?
Expressivity of Existential Rule Queries

Any query expressible as an existential rule query has the following property:

- it is closed under homomorphism;
- it is recursively enumerable.

But are all recursively enumerable closed under homomorphism query expressible by an existential query?

Yes.
Ideas of the Proof

We start from a Turing machine with the following property:

- if may not stop, but on any input tape that encodes a database belonging to the query, it outputs yes
- if it accepts the input encoding of a database $D$, and if there is a homomorphism from $D$ to $D'$, then it accepts the input encoding $D'$.

We need:

- a relational representation of the tape representing the database at hand
- to simulate the run of the Turing machine on this tape.
Already Known

We have already seen in a previous course how to simulate a Turing machine using existential rules. We have seen earlier today how to create a tape, but we made intensive use of two elements:

- a linear order on the domain elements;
- the ability to check for the absence of extensional atoms.

We do not have any of them at our disposal in the current setting.
Creating a Linear Order

In order to “simulate” a linear order, we are going to ‘guess” it:

- $ACDom(x) \rightarrow link(x, y) \land first(y) \land last(y)$
- $ACDom(x) \rightarrow link(x, y) \land first(y) \land partial(y)$
- $ACDom(x) \land partial(y) \rightarrow succ(y, z) \land link(x, z) \land partial(z)$
- $ACDom(x) \land partial(y) \rightarrow succ(y, z) \land link(x, z) \land last(z)$.
Effect of the Annotator

Let us check the effect of the annotator:
What Does it Bring Us?

We have sort of linear order, starting with first, ending with last, having a proper succ relation,... but:

- we have actually an infinity of them;
- some of them may not contain all the elements of the initial database;
- some of them may contain duplicates!
Dealing with the Limitations

- if we end up accepting a database by using an incomplete (but without repetition) linear order, this is fine! Because we know our query is closed under homomorphism
- one can also show that if we end up accepting a database by using a linear order with repetitions, this is also fine: we build a database $D'$ that would create this order as a correct linear order, and show that there is a homomorphism from $D'$ to $D$. 
Recall that to build the tape representation of the database, we used input negation to check whether an atom is present or not. Here, we adapt another approach:

- we enumerate all the tuples (as previously)
- for each tuple and each predicate, we guess whether the corresponding atom is present or not in the database.
Let $\Omega = \{ \omega_P \mid P \subseteq \mathcal{P}_e \}$.

$$first_k(x_1, \ldots, x_k, x_\ell) \rightarrow \omega(x_1, \ldots, x_k, x, x_\ell), \omega \in \Omega. \quad (1)$$

$$\omega(x_1, \ldots, x_k, x, x_\ell)$$
$$\quad \land succ_k(x_1, \ldots, x_k, x_{k+1}, \ldots, x_{2k}, x_\ell)$$
$$\rightarrow step(x, y, x_\ell) \land \omega'(x_{k+1}, \ldots, x_{2k}, y, x_\ell)$$
$$\quad \text{for all } \omega, \omega' \in \Omega. \quad (2)$$
Completing the Database – The Rules (2)

\[ \omega(x_1, \ldots, x_k, x, x_\ell) \land \text{step}(x, y, x_\ell) \rightarrow \omega(x_1, \ldots, x_k, y, x_\ell). \] (3)

\[ \omega_p(x_1, \ldots, x_k, x, x_\ell) \land \bigwedge_{i=1}^{k} \text{link}_\ell(y_i, x_i) \land p(y_1, \ldots, y_k) \]
\[ \rightarrow \text{allaccept}(x, x_\ell) \text{ if } p \notin P. \] (4)
Putting Things Together

So far, we have:

- generated arbitrarily many “linear orders” on (parts of) the set of individuals belonging to the initial database
- for each of them, generated the set of complete databases on the number of individuals appearing in the linear order

From these complete descriptions, we can create the tape of the Turing machine that semi-decides our query.
We finally accept in the following case:

“There exists a D-list such that for any way to choose the atoms of a database, either an atom of the original database is forgotten, or the tape generated from this D-list and the choice of atoms is accepted by the Turing machine recognizing our query”