Datalog-Based Data Access over Ontology Knowledge Bases
Unit 3 – Query Answering

Thomas Eiter
Institut für Informationssysteme, TU Wien

ICCL Summer School 2013, August 29-30, 2013

Austrian Science Fund (FWF) grants P20841, P24090
Unit Outline

1. Introduction

2. Query Answering over Relational Databases

3. Queries over Ontologies

4. Query Answering Techniques

5. Query Answering in $\mathcal{LDL}^+$

6. Query Answering in $\mathcal{SROEL}(\cap, \times)$

7. Query Answering in Horn-$\mathcal{SHIQ}$

8. Discussion and Conclusion
Introduction

- Traditionally, Description Logics were used to formalize domains and reason about concept relationships.

- Standard Reasoning services comprised satisfiability testing, instance checking, and classification.

- In recent years, more attention has been devoted to database-style query answering over ontologies:
  - Ontology Based Data Access (OBDA)
  - Enterprise Application Integration

- Issue: Methods for Query Answering
  - Focus: database technology \(\rightarrow\) Datalog
1. Introduction

Ontology Based Data Access (OBDA) is a key application of DLs. Hence, query answering in DLs is a crucial problem.

Evaluate a conjunctive query over an ABox $\mathcal{A}$, taking into account the constraints expressed by a DL TBox $\mathcal{T}$.

```
hasDevelopedCapital(x) ← country(x), hasCapital(x, y), city(y), hasHDI(y, high)
hasDevelopedCapital(Brazil) ← country(Brazil) hasHDI(Brasilia, high)
hasCapital(Brazil, Brasilia), city(Brasilia),
```

\[ \mathcal{A} \]

- country(Brazil)
- capital(Brasilia)
- isInLocatedIn(Brasilia, RegiãoCentroOeste)
- isInLocatedIn(RegiãoCentroOeste, Brazil)
- hasHDI(Brasilia, high)

\[ \mathcal{T} \]

- trans(isLocatedIn)
  - country $\sqsubseteq \exists$hasCapital.capital
  - hasCapital $\sqsubseteq$ isInLocatedIn$^-$
  - country $\sqsubseteq 1$ isInLocatedIn$^-$ .capital
  - country $\sqsubseteq \forall$hasCapital.city

T. Eiter / TU Wien  ICCL 2013 29/08/2013 4/49
Queries over Relational Databases

- **Conjunctive query (CQ):** rule

\[ q(\vec{X}) \leftarrow A_1(\vec{X}_1, \vec{Y}_1), \ldots, A_m(\vec{X}_n, \vec{Y}_n) \]

\[ \text{Body}(\vec{X}, \vec{Y}) \]

where all \(X_i\) are sublists from \(\vec{X} = X_1, \ldots, X_n\), and the \(\vec{Y}_i\) are disjoint from \(\vec{X}\)

- **Union of CQs (UCQs):** set of rules \(q(\vec{X}) \leftarrow \text{Body}_i(\vec{X}, \vec{Y}_i)\)

- **First-Order (FO) query:** FO formula \(\phi(\vec{X}) \ (\approx \text{SQL}; \ q(\vec{X}) \leftarrow \phi(\vec{X}))\)

- **Datalog query:** set of rules “defining” predicate \(q(\vec{X})\)
  - acyclic (hierarchic)
  - recursive
  - with/without negation (multiple answer sets: cautious inference)

**Boolean Query:** \(\vec{X}\) is void (i.e., empty list of free/head variables)
Semantics

<table>
<thead>
<tr>
<th>country(Brazil)</th>
<th>isLocatedIn(Brasilia, RegiãoCentroOeste)</th>
<th>hasHDI(Brasilia, high)</th>
</tr>
</thead>
<tbody>
<tr>
<td>capital(Brasilia)</td>
<td>isLocatedIn(RegiãoCentroOeste, Brazil)</td>
<td></td>
</tr>
</tbody>
</table>

- **Datalog perspective:**
  - database = set $D$ of facts
  - evaluate Datalog program $P = Q \cup D$ (no db predicates in rule heads)
  - query answers = all tuples $\vec{c}$ such that $P \models q(\vec{c})$ (i.e., $q(\vec{c}) \in LM(P)$)

  **Example:** has_some_HDI($X$) ← isLocatedIn($Y$, $X$), hasHDI($Y$, high)
  answers: RegiãoCentroOeste

- **First-order perspective:**
  - apply closed world assumption (CWA) on $D$, to obtain a single Herbrand model $I_D$
  - evaluate $Q = \Phi(\vec{X})$ over $I_D$
  - query answers = all tuples $\vec{c}$ such that $I_D \models \Phi(\vec{c})$

Note: CQs, UCQs amount to FO queries $\exists \vec{Y}.Body(\vec{X}, \vec{Y}), \lor_i \exists \vec{Y}.Body_i(\vec{X}, \vec{Y}_i)$
Complexity of Relational Query Answering

Traditional complexity analysis: decision problems (yes/no answer)

Complexity assessment of query answering:

**Query Output Tuple (QOT) Problem**

Decide whether given tuple \( \vec{c} \) is in the answer of query \( Q \) over input database \( D \)

Two common measures of complexity:

- *combined complexity*: in terms of \( Q \) and \( D \)
- *data complexity*: only in terms of data \( D \)

More often data complexity is considered
## Complexity Overview

<table>
<thead>
<tr>
<th>query $Q$</th>
<th>data complexity</th>
<th>combined complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>conjunctive queries</td>
<td>$AC_0$</td>
<td>NP-complete</td>
</tr>
<tr>
<td>union of CQs</td>
<td>$AC_0$</td>
<td>NP-complete</td>
</tr>
<tr>
<td>FO queries</td>
<td>$AC_0$</td>
<td>PSPACE-complete</td>
</tr>
<tr>
<td>propositional LP</td>
<td>$AC_0$</td>
<td>PTIME-complete</td>
</tr>
<tr>
<td>Datalog</td>
<td>PTIME-complete</td>
<td>EXPTIME-complete</td>
</tr>
<tr>
<td>stratified Datalog $\neg$</td>
<td>PTIME-complete</td>
<td>EXPTIME-complete</td>
</tr>
<tr>
<td>well-founded Datalog $\neg$</td>
<td>PTIME-complete</td>
<td>EXPTIME-complete</td>
</tr>
<tr>
<td>answer set Datalog $\neg$</td>
<td>co-NP-complete</td>
<td>co-NEXPTIME-complete</td>
</tr>
<tr>
<td>disjunctive Datalog $\neg$</td>
<td>co-NP-complete</td>
<td>co-NEXPTIME-complete</td>
</tr>
<tr>
<td>stratified / answer sets</td>
<td>$\Pi_2^p$-complete</td>
<td>co-NEXPTIME$^{NP}$-complete</td>
</tr>
</tbody>
</table>

**Note:**

- NP-hardness of CQs even for *predicate arity 2*
- PTIME for *acyclic queries*: graph with edges between variables co-occurring in some query atom is acyclic (e.g., monadic queries)
  (more general: bounded *treewidth*)
Queries over Ontologies

- Syntax: as above (but usually restricted)

- Semantics:
  - free (answer) variables \( \vec{X} \) must be bound to named individuals for evaluation, unnamed individuals play a role!
  - *open world assumption (OWA)*: all models of an ontology are relevant

**Definition**

Given an ontology \( \mathcal{O} \) and a FO query \( Q(\vec{X}) \), an *answer to \( Q \) is a ground substitution \( \theta \) of \( \vec{X} \) to the individuals such that \( \mathcal{O} \models Q(\vec{X}\theta) \).

*By \( \text{ans}(Q, \mathcal{O}) \) we denote the set of all answers of \( Q(\vec{X}) \).*

- For CQ \( Q(\vec{X}) \), this means \( \text{Body}(\vec{X}, \vec{Y}) \) has in each model of \( \mathcal{O} \) a homomorphomic embedding (“match”) \( \pi \) extending \( \theta \)

- For Datalog queries, similar definitions are possible
  - need semantics of combined rules and ontology!
Example, cont’d

\[ \begin{array}{|c|}
\hline
A \\
\hline
country(Brazil) \\
capital(\textit{Brasilia}) \\
\text{isLocatedIn}(\textit{Brasilia}, \textit{Região Centro Oeste}) \\
\text{isLocatedIn}(\textit{Região Centro Oeste}, Brazil) \\
\text{hasHDI}(\textit{Brasilia}, \textit{high}) \\
\hline
\end{array} \quad \begin{array}{|c|}
\hline
\textit{T} \\
\hline
\text{trans(isLocatedIn)} \\
country \sqsubseteq \exists \text{hasCapital}. \text{capital} \\
\text{hasCapital} \sqsubseteq \text{isLocatedIn}\neg \\
country \sqsubseteq \leq 1 \text{isLocatedIn}\neg . \text{capital} \\
country \sqsubseteq \forall \text{hasCapital}. \text{city} \\
\hline
\end{array} \]

\[
\text{hasDevelopedCapital}(x) \leftarrow \text{country}(x), \text{hasCapital}(x, y), \text{city}(y), \text{hasHDI}(y, \textit{high})
\]

- Let \( \theta = \{ x/\text{Brazil} \} \), and consider \( \pi = \theta \cup \{ y/\textit{Brasilia} \} \):
  - \( \pi \) matches \( \text{country}(\text{Brazil}), \text{hasHDI}(\textit{Brasilia}, \textit{high}) \) in the ABox
  - \( (\mathcal{T}, \mathcal{A}) \models \text{hasCapital}(\textit{Brasil}, \textit{Brasilia}) \) and \( (\mathcal{T}, \mathcal{A}) \models \text{city}(\textit{Brasilia}) \)
  - hence the \( \pi \) is a match in every model of \( \mathcal{O} \)

- query answer \textit{Brazil}
- no other \( \theta \) leads to a query answer
### Complexity

- Genuine queries over ontologies increase expressiveness ... but also complexity
- Decidability of CQ Answering might get lost
- (Full) CQ answering not supported by reasoners
- Many bad complexity results
  - $\mathcal{ALCIT}$ and $\mathcal{SH}$ are 2-EXP\textsc{time}-complete in combined complexity
  - already $\mathcal{AL}$ is intractable in data complexity \(\rightarrow\) bad news!
  - acyclic CQs are NP-hard in combined complexity already for DL-Lite\(^R\) \[Kikot et al., 2011\]
- Many algorithms of theoretical flavor (don’t seem efficiently implementable)

### Question

Practical query answering using existing efficient database technology (RDBMS, Datalog engines, etc.)?
Techniques for CQ Answering

Many different approaches for CQ Answering have been developed that adapt known techniques for standard reasoning, e.g.:

- Reduction to concept satisfiability (e.g., rolling up)
- Modified tableaux [Levy and Rousset, 1998], [Ortiz et al., 2008a] and resolution-based [Hustadt et al., 2005] techniques
- Tree automata based algorithms Calvanese et al. [2007,2009]
- Knots (mosaics and types) [Ortiz et al., 2008b], E et al. [2009,2008]
- Rewriting: transformation to reasoning problem in other logic (e.g., FO Logic, Datalog, ...)

T. Eiter / TU Wien
ICCL 2013 29/08/2013
Forest-Shaped Models

- Deciding query entailment implies considering all models.
- However, for many DLs it is sufficient to consider forest-shaped models: the individuals are arbitrarily connected and roots of trees.

Key property: given a KB $\mathcal{O}$ and a CQ $Q$, $\mathcal{O} \not\models Q$ iff there is a forest-shaped model of $\mathcal{O}$ in which there is no mapping for $Q$.

This property lies at the core of most existing query answering algorithms and complexity bounds.
Query Rewriting – Lightweight DLs

For lightweight DLs, query answering using database technologies

- for $\mathcal{DL}$-Lite
  - Query rewriting compiles $Q$ and $T$ into a UCQ/FO query $Q^T$
  - $Q^T$ can be evaluated over $\mathcal{A}$ only with off-the-shelf RDBMSs
  - Many, many papers on better and shorter rewritings

- for $\mathcal{EL}$
  - Query rewriting into Datalog (e.g., Requiem), no FO rewritability
  - Alternative: the combined approach
    - TBox partially materialized in $\mathcal{A}$ (polynomial expansion)
    - $Q$ rewritten into a FO query over the expanded data
    - evaluation possible with off-the-shelf RDBMS

- for OWL 2 RL:
  - Query rewriting into Datalog ($dl$-safety)
Datalog-Rewritability

- FO-rewritability excludes recursion
- Query answering is *not* FO-rewritable in more expressive DLs
  - e.g., in \( \mathcal{EL}, \mathcal{SHIQ} \)
- But, it may be expressible in Datalog
- Note:

**Theorem (Vardi, Immerman)**

Datalog\(^+\) (Datalog with input negation) captures \( \text{PTIME} \) on ordered databases (i.e., in presence of a successor relation).
Datalog-Rewritable DLs

Definition (Datalog-rewritable)

A DL $\mathcal{DL}$ is *Datalog-rewritable* if there exists a transformation $\Phi_{\mathcal{DL}}$ from $\mathcal{DL}$ KBs to Datalog programs such that, for any $\mathcal{DL}$ KB $\mathcal{O}$,

1. $\mathcal{O} \models Q(o)$ iff $\Phi_{\mathcal{DL}}(\mathcal{O}) \models Q(o)$ for any concept or role name $Q$ from $\mathcal{O}$, and individuals $o$ from $\mathcal{O}$;

2. $\Phi_{\mathcal{DL}}$ is *modular*, i.e., for $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ an ABox, $\Phi_{\mathcal{DL}}(\mathcal{O}) = \Phi_{\mathcal{DL}}(\mathcal{T}) \cup \mathcal{A}$;

Further properties: A DL $\mathcal{DL}$ is

- *polynomial Datalog-rewritable*, if $\mathcal{DL}$ is Datalog-rewritable and $\Phi_{\mathcal{DL}}(\mathcal{O})$ is computable in polynomial time;

- *non-uniform Datalog-rewritable*, if only condition (1) of Datalog-rewritability holds for $\mathcal{DL}$. 
Example Datalog-Rewritable DLs

- **$\mathcal{LDL}^+$** [Heymans et al., 2010]: lightweight ontology language, extending in essence core OWL 2 RL with singleton nominals, role conjunctions, and transitive closure

- **$\mathcal{SROEL}(\sqcap, \times)$** [Krötzsch, 2010]: superset of OWL 2 EL [Motik et al., 2008] resp. $\mathcal{EL}^+$
  - disregarding datatypes
  - adding (restricted) conjunction of roles ($R \sqcap S$), local reflexivity ($\text{Self}$), concept production ($C \times D \sqsubseteq T, R \sqsubseteq C \times D$)

- **$\mathcal{SROEL}(\times)$** [Krötzsch, 2011]

- **Horn-$\mathcal{SHIQ}$** [Ortiz et al., 2010]: Horn fragment of $\mathcal{SHIQ}$

- **$\mathcal{SROIQ}$-RL** [Bozzato and Serafini, 2013]: restriction of $\mathcal{SROIQ}$ for OWL 2 RL
\( \mathcal{LDL}^+ \)

- \( \mathcal{LDL}^+ \) forbids in axioms \( X \subseteq Y \)
  - disjunction \( C \sqcup D \) in \( Y \)
  - existentials \( \exists R \) in \( Y \)

- Viewing \( X \subseteq Y \) as rule \( Y \leftarrow X \), it distinguishes head (h) and body (b) concepts/roles, for occurrence in \( Y \) resp. \( X \)

- \( \mathcal{LDL}^+ \) shares properties with datalog programs:
  - It can express transitive closure (via an operator \( ^+ \))
  - An \( \mathcal{LDL}^+ \) ontology \( \mathcal{O} \) has a least model in each domain
  - For query answering, we can exclude *unnamed individuals* (i.e., use the *active* domain of individuals occurring in \( \mathcal{O} \))
**$\mathcal{LDL}^+$ Ontologies**

An $\mathcal{LDL}^+$ ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A}\rangle$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$, where

- $\mathcal{T}$ is a set of
  - **terminological axioms** $B \sqsubseteq H$, where $B$ is a $b$-concept and $H$ is an $h$-concept, and
  - **role axioms** $S \sqsubseteq T$, where $S$ is a $b$-role and $T$ is an $h$-role,

- $\mathcal{A}$ is a set of assertions of the form $C(o)$ and $S(o_1, o_2)$ where $C$ is an $h$-concept and $S$ is an $h$-role.

**Example (overloaded person)**

\[
L = \left\{ \begin{array}{l}
(\geq 2 \text{ PaptoRev.}) \sqsubseteq \text{ Over} \\
\text{Over} \sqsubseteq \forall \text{Super}^+.\text{Over} \\
\text{Super}(\text{tom}, \text{sue}); \text{ Super}(\text{sue}, \text{joe})
\end{array} \right\}
\]

- someone who has at least 2 papers to review is overloaded
- overload means all supervised persons in hierarchy are overloaded
Syntax of $\mathcal{LDL}^+$ – Roles

head (h-) and body (b-) restrictions on roles in $\mathcal{LDL}^+$ axioms

- **h-roles** (h for head) $S, T$ are
  - (i) role names $R$,
  - (ii) role inverses $S^-$,
  - (iii) role conjunctions $S \cap T$, and
  - (iv) role top $\top^2$;

- **b-roles** (b for body) $S, T$ are the same as h-roles, plus
  - (v) role disjunctions $S \sqcup T$,
  - (vi) role sequences $S \circ T$,
  - (vii) transitive closures $S^+$, and
  - (viii) role nominals $\{(o_1, o_2)\}$, where $o_1, o_2$ are individuals.
Syntax of $\mathbb{LDL}^+$ – Concepts

head (h-) and body (b-) restrictions on concepts in $\mathbb{LDL}^+$ axioms

- **basic concepts** $C, D$ are concept names $A$, $\top$, and conjunctions $C \sqcap D$;
- **h-concepts** are
  (i) **basic concepts** $B$, and
  (ii) **value restrictions** $\forall S.B$ where $S$ is a b-role;
- **b-concepts** $C, D$ are
  (i) **basic concepts** $B$,
  (ii) **disjunctions** $C \sqcup D$,
  (iii) **exists restrictions** $\exists S.C$,
  (iv) **atleast restrictions** $\geq nS.C$, and
  (v) **nominals** $\{o\}$, where $S$ is a b-role, and $o$ is an individual.
Transformation of $\mathcal{LDL}^+$ to Datalog

The transformation $\Phi_{\mathcal{LDL}^+}(\mathcal{O})$ of an $\mathcal{LDL}^+$ ontology $\mathcal{O}$ to Datalog contains the following elements:

- for convenience, we can “inline” the A-Box into the T-Box using nominals: $C(a) \leadsto \{a\} \subseteq C$; $R(a, b) \leadsto \{(a, b)\} \subseteq R$
- transformation of the $\mathcal{LDL}^+$ axioms in $\mathcal{O}$;
- transformation of the closure of $\mathcal{O}$.

Definition (closure)

The *closure* of an $\mathcal{LDL}^+$ knowledge base $\mathcal{O}$, denoted $\text{clos}(\mathcal{O})$, as the smallest set containing

- all subexpressions that occur in $\mathcal{O}$ (both roles and concepts) except value restrictions, and
- for each role name occurring in $\mathcal{O}$, its inverse.
Transformation Rules

- **Axiom translation:**

  \[
  \begin{align*}
  B \subseteq H & \quad H(X) \leftarrow B(X) \\
  B \subseteq \forall E.A & \quad A(Y) \leftarrow B(X), E(X, Y). \\
  S \subseteq T & \quad T(X, Y) \leftarrow S(X, Y)
  \end{align*}
  \]

- **Closure translation:**

  \[
  \begin{array}{ll}
  \text{role name } P & P(X, Y) \leftarrow P^-(Y, X) \\
  \text{concept name } A & \top(X) \leftarrow A(X) \\
  \text{role name } (R) & \top(X) \leftarrow R(X, Y) \quad \top(Y) \leftarrow R(X, Y) \\
  \top & \top^2(X, Y) \leftarrow \top(X), \top(Y).
  \end{array}
  \]

  \[
  \begin{align*}
  D &= \{o\} & D(o) & \leftarrow \\
  D &= D_1 \cap D_2 & D(X) & \leftarrow D_1(X), D_2(X) \\
  D &= D_1 \cup D_2 & D(X) & \leftarrow D_1(X) & D(X) & \leftarrow D_2(X) \\
  D &= \exists E.D_1 & D(X) & \leftarrow E(X, Y), D_1(Y) \\
  D &= \geq n E.D_1 & D(X) & \leftarrow E(X, Y_1), D(Y_1), \ldots, E(X, Y_n), D(Y_n), \\
  & & & Y_1 \neq Y_2, \ldots, Y_i \neq Y_j, \ldots, Y_{n-1} \neq Y_n
  \end{align*}
  \]

  \[
  \begin{align*}
  E &= \{(o_1, o_2)\} & E(o_1, o_2) & \leftarrow \\
  E &= F^- & E(X, Y) & \leftarrow F(Y, X) \\
  E &= E_1 \cap E_2 & E(X, Y) & \leftarrow E_1(X, Y), E_2(X, Y) \\
  E &= E_1 \cup E_2 & E(X, Y) & \leftarrow E_1(X, Y) & E(X, Y) & \leftarrow E_2(X, Y) \\
  E &= E_1 \circ E_2 & E(X, Y) & \leftarrow E_1(X, Z), E_2(Z, Y) \\
  E &= F^+ & E(X, Y) & \leftarrow F(X, Y) & E(X, Y) & \leftarrow F(X, Z), E(Z, Y)
  \end{align*}
  \]
Example (cont’d)

Inline A-Box:
\[ O = \left\{ \begin{array}{l}
(\geq 2 \text{PaptoRev}.\top) \sqsubseteq \text{Over} \\
\text{Over} \sqsubseteq \forall \text{Super}^+.\text{Over} \\
\{(\text{tom}, \text{sue})\} \sqcup \{(\text{sue}, \text{joe})\} \sqsubseteq \text{Super}
\end{array} \right\} \]

\[ \text{clos}(L) = \left\{ \text{Super}, \text{Super}^-, \text{Super}^+, \text{Over}, \text{Over}^-, \text{PaptoRev}, \text{PaptoRev}^-, \top, (\geq 2 \text{PaptoRev}.\top), \{(\text{tom}, \text{sue})\}, \{(\text{sue}, \text{joe})\}, \{(\text{tom}, \text{sue})\} \sqcup \{(\text{sue}, \text{joe})\} \right\} \]

\[ \Phi_{\mathcal{DL}^+}(O): \]
\[ R^-(Y,X) \leftarrow R(X,Y) \quad R(Y,X) \leftarrow R^-(X,Y) \quad R \in \{\text{Over}, \text{Super}, \text{PaptoRev}\} \]
\[ \top(X) \leftarrow R(X,Y) \quad \top(Y) \leftarrow R(X,Y) \]
\[ \text{Super}^+(X,Y) \leftarrow \text{Super}(X,Y) \]
\[ \text{Super}^+(X,Y) \leftarrow \text{Super}^+(X,Z), \text{Super}(Z,Y) \]
\[ (\geq 2 \text{PaptoRev}.\top)(X,Y) \leftarrow \text{PaptoRev}(X,Y_1), \top(Y_1), \]
\[ \text{PaptoRev}(X,Y_2), \top(Y_2), Y_1 \neq Y_2 \]
\[ \{(\text{tom}, \text{sue})\}(\text{tom}, \text{sue}) \leftarrow . \quad \{(\text{sue}, \text{joe})\}(\text{sue}, \text{joe}) \leftarrow . \]
\[ \{(\text{tom}, \text{sue})\} \sqcup \{(\text{sue}, \text{joe})\}(X,Y) \leftarrow \{(\text{tom}, \text{sue})\}(X,Y) \]
\[ \{(\text{tom}, \text{sue})\} \sqcup \{(\text{sue}, \text{joe})\}(X,Y) \leftarrow \{(\text{sue}, \text{joe})\}(X,Y) \]
Formal Properties

Theorem

For every \( \mathcal{LDL}^+ \) ontology \( \mathcal{O} \),

(i) \( \mathcal{O} \models C(a) \iff \Phi_{\mathcal{LDL}^+}(\mathcal{O}) \models C(a) \)

(ii) \( \mathcal{O} \models R(a, b) \iff \Phi_{\mathcal{LDL}^+}(\mathcal{O}) \models R(a, b) \).

Notes:

- \( \Phi_{\mathcal{LDL}^+}(\mathcal{O}) \) can be constructed in polynomial time from \( \mathcal{O} \) (unary encoding of counting \( \geq nR \))
- can be evaluated in polynomial time (rule matching is polynomial)
- for true modularity: suppress inlining, add ABox assertions as facts.
- the above result extends to CQs and UCQs \( Q(\vec{X}) \):
  \[ \vec{c} \in \text{ans}(Q, \mathcal{O}) \iff \Phi_{\mathcal{LDL}^+}(\mathcal{O}) \cup Q(\vec{X}) \models q(\vec{c}) \]
**SROEL(∩, ×)**

- **SROEL(∩, ×)** is in essence a superset of OWL 2 EL

**Differences:**
- disregards datatypes
- adding conjunction of roles (\(R ∩ S\)), local reflexivity (\(Self\)), concept production (\(C × D ⊆ T, R ⊆ C × D\))
- restrictions on role occurrences in a KB (simplicity, range restrictions), but not role regularities

- **SROEL(∩, ×)** has polynomial complexity (sat, instance checking)

- [Krötzsch, 2010] describes a proof system for instance checking over a SROEL(∩, ×) ontology

- This proof system can be naturally encoded in a logic program, viewing axioms \(\alpha\) as facts and inference rules \(\frac{\alpha_1, \ldots, \alpha_n}{\alpha}\) as rules \(\alpha \leftarrow \alpha_1, \ldots, \alpha_n\)

- A universal (schematic) encoding in Datalog is possible
**SROEL (∩, ×)**, cont’d

**Key aspects:**

- It is sufficient to generate a small part of a canonical forest-shaped model.

- More precisely, only new elements directly connected to some individual, due to existential axioms \( A ⊑ ∃R.B \).

- For uniform (ABox independent) encoding, *share new elements*.
Transformation of $\textit{SROEL}(\sqcap, \times)$ to Datalog

- $\textit{SROEL}(\sqcap, \times)$ proof system for $\mathcal{O}$:
  - the axioms $C \sqsubseteq D$, $C(a)$ etc of $\mathcal{O}$ can be understood as facts
    E.g., $C \sqsubseteq D$ viewed as $\sqsubseteq(C, D)$ (infix)
  - view the inference rules $\frac{\alpha}{\alpha_1, \ldots, \alpha_n}$ as LP rules $\alpha \leftarrow \alpha_1, \ldots, \alpha_n$
    E.g., $\frac{C \sqsubseteq D, C(a)}{D(a)}$ can be viewed as rule $D(a) \leftarrow \sqsubseteq(C, D), C(a)$

- Use reification to obtain a Datalog representation
  \[ \Phi_{\mathcal{EL}}(\mathcal{O}) = I_{\text{inst}}(\mathcal{O}) \cup P_{\text{inst}}(\mathcal{O}) \]
  where $\mathcal{O}_{\text{inst}}$ encodes $\mathcal{O}$ and $P_{\text{inst}}$ is a fixed set of rules (schemata)
  - names: $C \rightsquigarrow \text{cls}(C)$; $R \rightsquigarrow \text{rol}(R)$; $a \rightsquigarrow \text{nom}(a)$
  - assertions: e.g. $C(a) \rightsquigarrow \text{isa}(a, C)$; $R(a, b) \rightsquigarrow \text{triple}(a, R, b)$
  - axioms: e.g. $A \sqsubseteq C \rightsquigarrow \text{subClass}(A, C)$,

- Make reified rules generic using variables
  E.g. $\text{isa}(a, D) \leftarrow \text{subClass}(C, D), \text{isa}(a, C)$ gets
  $\text{isa}(X, Z) \leftarrow \text{subClass}(Y, Z), \text{isa}(X, Y)$
Encoding $I_{\text{inst}}(\mathcal{O})$

<table>
<thead>
<tr>
<th>$C(a) \leadsto isa(a, C)$</th>
<th>$R(a, b) \leadsto triple(a, R, b)$</th>
<th>$a \in N_I \leadsto nom(a)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\top \sqsubseteq C \leadsto top(C)$</td>
<td>$A \sqsubseteq \bot \leadsto bot(A)$</td>
<td>$A \in N_C \leadsto cls(A)$</td>
</tr>
<tr>
<td>${a} \sqsubseteq C \leadsto subClass(a, C)$</td>
<td>$A \sqsubseteq {c} \leadsto subClass(A, c)$</td>
<td>$R \in N_R \leadsto rol(R)$</td>
</tr>
<tr>
<td>$A \sqsubseteq C \leadsto subClass(A, C)$</td>
<td>$A \sqcap B \sqsubseteq C \leadsto subConj(A, B, C)$</td>
<td></td>
</tr>
<tr>
<td>$\exists R.\text{Self} \sqsubseteq C \leadsto subSelf(R, C)$</td>
<td>$A \sqsubseteq \exists R.\text{Self} \leadsto supSelf(A, R)$</td>
<td></td>
</tr>
<tr>
<td>$\exists R.A \sqsubseteq C \leadsto subEx(R, A, C)$</td>
<td>$A \sqsubseteq \exists R.B \leadsto supEx(A, R, B, e^{A \sqsubseteq \exists R.B})$</td>
<td></td>
</tr>
<tr>
<td>$R \sqsubseteq T \leadsto subRole(R, T)$</td>
<td>$R \circ S \sqsubseteq T \leadsto subRConj(R, S, T)$</td>
<td></td>
</tr>
<tr>
<td>$R \sqsubseteq C \times D \leadsto supProd(R, C, D)$</td>
<td>$A \times B \sqsubseteq R \leadsto supProd(A, B, R)$</td>
<td></td>
</tr>
<tr>
<td>$R \sqcap S \sqsubseteq T \leadsto subRConj(R, S, T)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Encode axiom $\alpha \leadsto I_{\text{inst}}(\alpha)$
- Encode individual $s \leadsto I_{\text{inst}}(s)$
- $I_{\text{inst}}(\mathcal{O}) = \{I_{\text{inst}}(\alpha) \mid \alpha \in L\} \cup \{I_{\text{inst}}(s) \mid s \in N_I \cup N_C \cup N_R\}$
  - use constants $e^{A \sqsubseteq \exists R.B}$ for elements enforced by existential axioms $A \sqsubseteq \exists R.B$
  - encode in $supEx(A, R, B, e^{A \sqsubseteq \exists R.B})$ the pattern $\xrightarrow{A \circ \bigodot R \bigodot B}$
  - “share” $e^{A \sqsubseteq \exists R.B}$ for individuals $a, b$ belonging to $A$
Example

Consider

\[ \mathcal{O} = \{ A(a), A \sqsubseteq \exists R.B, B \sqsubseteq C, \exists R.C \sqsubseteq D \} \]

\( \mathcal{O} \) is translated to

\[ I_{inst}(\mathcal{O}) = \left\{ \begin{array}{l}
isa(a, A), \supEx(A, R, B, e^{A \sqsubseteq \exists R.B}), \subClass(B, C), \\
\subEx(R, C, D), \nom(a), \cls(A), \cls(B), \cls(C), \cls(D), \rol(R) \end{array} \right\}. \]
Inference Rules (Datalog Encoding)

Datalog program $P_{inst}$: instance inference

\[
\begin{align*}
\text{isa}(X, Z) & \leftarrow \text{top}(Z), \text{isa}(X, Z') \\
\text{isa}(X, Y) & \leftarrow \text{bot}(Z), \text{isa}(U, Z), \text{isa}(X, Z'), \text{cls}(Y) \\
\text{isa}(X, Z) & \leftarrow \text{subClass}(Y, Z), \text{isa}(X, Y) \\
\text{isa}(X, Z) & \leftarrow \text{subConj}(Y_1, Y_2, Z), \text{isa}(X, Y_1), \text{isa}(X, Y_2) \\
\text{isa}(X, Z) & \leftarrow \text{subEx}(V, Y, Z), \text{triple}(X, V, X'), \text{isa}(X', Y) \\
\text{isa}(X, Z) & \leftarrow \text{subEx}(V, Y, Z), \text{self}(X, V), \text{isa}(X, Y) \\
\text{isa}(X', Z) & \leftarrow \text{supEx}(Y, V, Z, X'), \text{isa}(X, Y) \\
\text{isa}(X, Z) & \leftarrow \text{subSelf}(V, Z), \text{self}(X, V) \\
\text{isa}(X, Z_1) & \leftarrow \text{supProd}(V, Z_1, Z_2), \text{triple}(X, V, X') \\
\text{isa}(X, Z_1) & \leftarrow \text{supProd}(V, Z_1, Z_2), \text{self}(X, V) \\
\text{isa}(X', Z_2) & \leftarrow \text{supProd}(V, Z_1, Z_2), \text{triple}(X, V, X') \\
\text{isa}(X, Z_2) & \leftarrow \text{supProd}(V, Z_1, Z_2), \text{self}(X, V) \\
\text{isa}(X, X) & \leftarrow \text{nom}(X) \\
\text{isa}(Y, Z) & \leftarrow \text{isa}(X, Y), \text{nom}(Y), \text{isa}(X, Z) \\
\text{isa}(X, Z) & \leftarrow \text{isa}(X, Y), \text{nom}(Y), \text{isa}(Y, Z)
\end{align*}
\]
Inference Rules (Datalog Encoding), cont’d

Datalog program $P_{inst}$: role and $Self$ inference

\[
\begin{align*}
\text{triple}(X, W, X') & \leftarrow \text{subRole}(V, W), \text{triple}(X, V, X') \\
\text{triple}(X, W, X'') & \leftarrow \text{subRChain}(U, V, W), \text{triple}(X, U, X'), \text{triple}(X', V, X'') \\
\text{triple}(X, W, X') & \leftarrow \text{subRChain}(U, V, W), \text{self}(X, U), \text{triple}(X, V, X') \\
\text{triple}(X, W, X') & \leftarrow \text{subRChain}(U, V, W), \text{triple}(X, U, X'), \text{self}(X', V) \\
\text{triple}(X, W, X) & \leftarrow \text{subRChain}(U, V, W), \text{self}(X, U), \text{self}(X, V) \\
\text{triple}(X, W, X') & \leftarrow \text{subRConj}(V_1, V_2, W), \text{triple}(X, V_1, X'), \text{triple}(X, V_2, X') \\
\text{triple}(Z, U, Y) & \leftarrow \text{isa}(X, Y), \text{nom}(Y), \text{triple}(Z, U, X) \\
\text{triple}(X, V, X') & \leftarrow \text{supEx}(Y, V, Z, X'), \text{isa}(X, Y) \\
\text{triple}(X, W, X') & \leftarrow \text{subProd}(Y_1, Y_2, W), \text{isa}(X, Y_1), \text{isa}(X', Y_2) \\
\text{self}(X, V) & \leftarrow \text{nom}(X), \text{triple}(X, V, X) \\
\text{self}(X, W) & \leftarrow \text{subRole}(V, W), \text{self}(X, V) \\
\text{self}(X, W) & \leftarrow \text{subRConj}(V_1, V_2, W), \text{self}(X, V_1), \text{self}(X, V_2) \\
\text{self}(X, W) & \leftarrow \text{subProd}(Y_1, Y_2, W), \text{isa}(X, Y_1), \text{isa}(X, Y_2) \\
\text{self}(X, V) & \leftarrow \text{supSelf}(Y, V), \text{isa}(X, Y)
\end{align*}
\]
Instance Queries

- $\Phi_{\mathcal{E}L}(\mathcal{O}) = P_{\text{inst}} \cup I_{\text{inst}}(\mathcal{O})$ can be used to decide satisfiability
- $\Phi_{\mathcal{E}L}(\mathcal{O})$ can be used to answer instance queries

**Theorem**

*For every $SROEL(\sqcap, \times)$ ontology $\mathcal{O}$ and $a, b \in N_I$*

(i) $\mathcal{O} \models C(a)$ iff $\Phi_{\mathcal{E}L}(\mathcal{O}) \models \text{isa}(a, C)$

(ii) $\mathcal{O} \models R(a, b)$ iff $\Phi_{\mathcal{E}L}(\mathcal{O}) \models \text{triple}(a, R, b)$. 
Example, cont’d

Consider \( \mathcal{O} = \{ A(a), A \sqsubseteq \exists R.B, B \sqsubseteq C, \exists R.C \sqsubseteq D \} \)

\[
I_{\text{inst}}(\mathcal{O}) = \left\{ \text{isa}(a, A), \text{supEx}(A, R, B, e^{A \sqsubseteq \exists R.B}), \text{subClass}(B, C), \\
\text{subEx}(R, C, D), \text{nom}(a), \text{cls}(A), \text{cls}(B), \text{cls}(C), \text{cls}(D), \text{rol}(R) \right\} .
\]

- We have \( \mathcal{O} \models D(a) \)
- From \( \Phi_{\mathcal{EL}}(\mathcal{O}) \) we can derive \( I_{\text{inst}}(D(a)) = \text{isa}(a, D) \):
Example, cont’d

Consider $\mathcal{O} = \{ A(a), A \sqsubseteq \exists R.B, B \sqsubseteq C, \exists R.C \sqsubseteq D \}$

$I_{\text{inst}}(\mathcal{O}) = \{ \text{isa}(a, A), \text{supEx}(A, R, B, e^{A \sqsubseteq \exists R.B}), \text{subClass}(B, C), \text{subEx}(R, C, D), \text{nom}(a), \text{cls}(A), \text{cls}(B), \text{cls}(C), \text{cls}(D), \text{rol}(R) \}$.

- We have $\mathcal{O} \models D(a)$
- From $\Phi_{\mathcal{E}\mathcal{L}}(\mathcal{O})$ we can derive $I_{\text{inst}}(D(a)) = \text{isa}(a, D)$:
  - apply $\text{isa}(X', Z) \leftarrow \text{supEx}(Y, V, Z, X'), \text{isa}(X, Y)$:
    $\text{isa}(e^{A \sqsubseteq \exists R.B}, B)$
Example, cont’d

Consider \( \mathcal{O} = \{ A(a), A \sqsubseteq \exists R.B, B \sqsubseteq C, \exists R.C \sqsubseteq D \} \)

\[
I_{\text{inst}}(\mathcal{O}) = \left\{ \text{isa}(a, A), \supEx(A, R, B, e^{A \sqsubseteq \exists R.B}), \text{subClass}(B, C), \supEx(R, C, D), \text{nom}(a), \text{cls}(A), \text{cls}(B), \text{cls}(C), \text{cls}(D), \text{rol}(R) \right\}.
\]

- We have \( \mathcal{O} \models D(a) \)
- From \( \Phi_{\mathcal{E}\mathcal{L}}(\mathcal{O}) \) we can derive \( I_{\text{inst}}(D(a)) = \text{isa}(a, D) \):
  - apply \( \text{isa}(X', Z) \leftarrow \supEx(Y, V, Z, X'), \text{isa}(X, Y) : \)
    \[
    \text{isa}(e^{A \sqsubseteq \exists R.B}, B)
    \]
  - apply \( \text{isa}(X, Z) \leftarrow \text{subClass}(Y, Z), \text{isa}(X, Y) : \)
    \[
    \text{isa}(e^{A \sqsubseteq \exists R.B}, C)
    \]
Example, cont’d

Consider $\mathcal{O} = \{ A(a), A \sqsubseteq \exists R.B, B \sqsubseteq C, \exists R.C \sqsubseteq D \}$

$I_{inst}(\mathcal{O}) = \{ \text{isa}(a, A), \supEx(A, R, B, e^{A \sqsubseteq \exists R.B}), \text{subClass}(B, C), \subEx(R, C, D), \text{nom}(a), \text{cls}(A), \text{cls}(B), \text{cls}(C), \text{cls}(D), \text{rol}(R) \}$.

- We have $\mathcal{O} \models D(a)$
- From $\Phi_{\mathcal{EL}}(\mathcal{O})$ we can derive $I_{inst}(D(a)) = \text{isa}(a, D)$:
  - apply $\text{isa}(X', Z) \leftarrow \supEx(Y, V, Z, X'), \text{isa}(X, Y)$:
    $\text{isa}(e^{A \sqsubseteq \exists R.B}, B)$
  - apply $\text{isa}(X, Z) \leftarrow \text{subClass}(Y, Z), \text{isa}(X, Y)$:
    $\text{isa}(e^{A \sqsubseteq \exists R.B}, C)$
  - apply $\text{triple}(X, V, X') \leftarrow \supEx(Y, V, Z, X'), \text{isa}(X, Y)$
    $\text{triple}(a, R, e^{A \sqsubseteq \exists R.B})$
Example, cont’d

Consider $\mathcal{O} = \{ A(a), A \sqsubseteq \exists R.B, B \sqsubseteq C, \exists R.C \sqsubseteq D \}$

$I_{\text{inst}}(\mathcal{O}) = \{ \text{isa}(a, A), \text{supEx}(A, R, B, e^{A \sqsubseteq \exists R.B}), \text{subClass}(B, C), \text{subEx}(R, C, D), \text{nom}(a), \text{cls}(A), \text{cls}(B), \text{cls}(C), \text{cls}(D), \text{rol}(R) \} \cdot$

- We have $\mathcal{O} \models D(a)$

- From $\Phi_{\mathcal{EL}}(\mathcal{O})$ we can derive $I_{\text{inst}}(D(a)) = \text{isa}(a, D)$:
  
  - apply $\text{isa}(X', Z) \leftarrow \text{supEx}(Y, V, Z, X'), \text{isa}(X, Y)$:
    
    $\text{isa}(e^{A \sqsubseteq \exists R.B}, B)$
  
  - apply $\text{isa}(X, Z) \leftarrow \text{subClass}(Y, Z), \text{isa}(X, Y)$:
    
    $\text{isa}(e^{A \sqsubseteq \exists R.B}, C)$
  
  - apply $\text{triple}(X, V, X') \leftarrow \text{supEx}(Y, V, Z, X'), \text{isa}(X, Y)$
    
    $\text{triple}(a, R, e^{A \sqsubseteq \exists R.B})$
  
  - apply $\text{isa}(X, Z) \leftarrow \text{subEx}(V, Y, Z), \text{triple}(X, V, X'), \text{isa}(X', Y)$
    
    $\text{isa}(a, D)$
Extensions

- Above encoding does not work for conjunctive queries
  - But for “DL-safe” queries (all variables bound to ABox-individuals)

- An extension to negative instance queries $\neg C(a)$ is possible:
  $\Phi_{\mathcal{E}L}(O)$

- A variant of $\Phi_{\mathcal{E}L}(O)$ is used by the DReW solver which omits concept products but supports domain and range assertions for roles

- [Krötzsch, 2011] gave a similar Datalog “materialization calculus” for $\mathcal{SROEL}(\times)$ instance checking, and proves that 3-ary predicates arity are necessary for any sound and complete such calculus

- Most recently, Bozzato and Serafini [2013] extended the Datalog materialization calculus for instance checking in $\mathcal{SROIQ}$-RL
Query Answering in Horn-\textit{SHIQ}

- \textit{SHIQ} is an expressive DL (cf. OWL Lite)
  - transitive roles (\textit{S}), role hierarchies (\textit{H}), inverses (\textit{I})
  - qualified number restrictions (\textit{Q})

- Horn fragment (Horn-\textit{SHIQ}): eliminate positive disjunction $\sqcup$ on right hand side

- Horn-\textit{SHIQ} has useful features missing in $\mathcal{EL}$ and $\mathcal{DL}$-Lite

\begin{quote}
\textit{trans}(\text{isLocatedIn}) \quad \text{country} \sqsubseteq \forall \text{hasCapital} \cdot \text{city} \quad \text{country} \sqsubseteq \leq 1 \text{isLocatedIn} \neg \text{.capital}
\end{quote}

- CQ Answering for Horn-\textit{SHIQ} is \textbf{tractable in data complexity (\textsc{PTIME}-complete)}

- The combined complexity of CQs is not higher than for satisfiability testing (\textsc{EXPTIME}-complete)

- Its features make CQ answering for Horn-\textit{SHIQ} significantly more complex than for $\mathcal{EL}$
**Issues**

- Match the query *Q partially between graph part and trees*  
  \(\Rightarrow\) tree-shaped query parts

- Inverse roles allow to move up and down the tree  
  \(\Rightarrow\) connect different trees

- Transitive roles: how far to go for a match in a tree?
Datalog Query Answering for Horn-$\mathcal{SHIQ}$

Ortiz et al. [2010]: CQ rewriting to Datalog (big predicate arities; impractical)

E_ et al. [2012a,2012b]: better rewriting

Three components:

- **UOC rewriting**: CQ $Q \leadsto$ UCQ $\text{rew}_T(Q)$ (depends on the TBox $T$)
- **TBox saturation**: enrich $T$ with relevant axioms for rewriting ($\Xi(T)$)
- **ABox completion**: $T$ is rewritten into a set of Datalog rules $\text{cr}(T)$ to “complete” the graph part

Answering $Q$ over $(T, A)$ amounts to evaluating the Datalog program

$$A \cup \text{cr}(T) \cup \text{rew}_T(q)$$

- One can evaluate $\text{rew}_T(Q)$ over the completion of $A$ (with no additional unnamed objects)
- $\text{rew}_T(q)$ can be exponential, but has manageable size for real queries and ontologies
The rewriting algorithm

Main idea:

- Eliminate query variables that can be matched at unnamed objects
  - Query matches have tree-shaped parts
  - We clip off the variables \( x \) that can be leaves
  - Replace them by constraints \( \text{D}(y) \) on their parent variables \( y \)
  - The added atoms \( \text{D}(y) \) ensure the existence of a match for \( x \)

- In the resulting queries all variables are matched to named objects

A Horn-SHIF TBox \( \mathcal{T} \) is in normal form, if GCIs in \( \mathcal{T} \) have the forms:

(F1) \[ A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B, \]

(F2) \[ A_1 \sqsubseteq \exists r.B, \]

(F3) \[ A_1 \sqsubseteq \forall r.B, \]

(F4) \[ A_1 \sqsubseteq \leq 1 r.B, \]

where \( A_1, \ldots, A_n, B \) are concept names and \( r \) is a role.

Normalize \( \mathcal{T} \) (efficiently doable, [Kazakov, 2009], [Krötzsch et al., 2007])
One Step of Query Rewriting

\[ q(x_1) \leftarrow r(x_1, x_2), r(x_1, x_4), r(x_2, x_3), s(x_3, x_4), A(x_1), B(x_4), B'(x_2), C(x_3) \]
One Step of Query Rewriting

1. Select the non-distinguished variable $x_3$
One Step of Query Rewriting

1. Select the non-distinguished variable $x_3$

2. Ensure that $x_3$ has only incoming edges
   - replace $r(x, y)$ by $r^-(y, x)$ as needed
One Step of Query Rewriting

1. Select the non-distinguished variable $x_3$

2. Ensure that $x_3$ has only incoming edges
   ▶ replace $r(x, y)$ by $r^-(y, x)$ as needed

3. Merge the predecessors
   ▶ if $x_3$ is a leaf of a tree, they must be mapped together
One Step of Query Rewriting

1. Select the non-distinguished variable $x_3$
2. Ensure that $x_3$ has only incoming edges
   - replace $r(x, y)$ by $r^-(y, x)$ as needed
3. Merge the predecessors
   - if $x_3$ is a leaf of a tree, they must be mapped together
4. Find an axiom that enforces an $(r \cap s^-)$-child that is $C$
   - fail if $\mathcal{T}$ does not imply such an axiom
One Step of Query Rewriting

1. Select the non-distinguished variable $x_3$
2. Ensure that $x_3$ has only incoming edges
   - replace $r(x, y)$ by $r^-(y, x)$ as needed
3. Merge the predecessors
   - if $x_3$ is a leaf of a tree, they must be mapped together
4. Find an axiom that enforces an $(r \sqcap s^-)$-child that is $C$
   - fail if $T$ does not imply such an axiom
5. Drop $x_3$ and add $D(x_2)$
One Step of Query Rewriting

1. Select the non-distinguished variable $x_3$

2. Ensure that $x_3$ has only incoming edges
   - replace $r(x, y)$ by $r^-(y, x)$ as needed

3. Merge the predecessors
   - if $x_3$ is a leaf of a tree, they must be mapped together

4. Find an axiom that enforces an $(r \cap s^-)$-child that is $C$
   - fail if $\mathcal{T}$ does not imply such an axiom

5. Drop $x_3$ and add $D(x_2)$
Another Step of Query Rewriting

The query using the axiom is rewritten to

\[ A \sqsupseteq \exists R_2.A_3 \]

\[ A_1 \xrightarrow{R_1} x_1 \]
\[ A_2 \xrightarrow{R_2} A_3 \xrightarrow{R_3} x_3 \]

\[ A_2, A \xrightarrow{R_1} x_2 \]
\[ A_4 \xrightarrow{R_3} x_4 \]
Transitive Roles

To handle transitive roles in the query $Q$:

- introduce a new variable between eliminated variable and some of its predecessors
- eliminate sets of variables
  
  variables connected in the query may be mapped to same element
  
  (reach the element on paths of different length)

Note:

- the number of variables in $Q$ does not increase (reuse of variables possible)
- only an exponential number of queries are possible
- the labels on edges of the query graph increase

Thus, rewriting terminates
TBox Saturation

- A set $\Xi(\mathcal{T})$ of relevant axioms is computed in advance
  - Tailored resolution calculus for Horn-$\mathcal{ALCHIQ}$
  - Adaptation of existing consequence driven procedures for satisfiability [Kazakov, 2009], [Ortiz et al., 2010]

Example Rules (all: Appendix)

$$M \sqsubseteq \exists S. (N \cap N') \quad N \sqsubseteq A$$

$$\frac{M \sqsubseteq \exists S. (N \cap N' \cap A)}{M \sqsubseteq \exists S. (N \cap N') \cap A} \quad R^c_{\sqsubseteq}$$

$$M \sqsubseteq \exists(S \cap inv(r)).(N \cap A) \quad A \sqsubseteq \forall r. B$$

$$\frac{M \sqsubseteq B}{M \sqsubseteq \forall r. B} \quad R^\forall$$

- The rewriting step simply searches for an axiom in $\Xi(\mathcal{T})$
**ABox Completion Rules**

The completion rules $cr(\mathcal{T})$ are straightforward:

1. $B(y) \leftarrow A(x), r(x, y)$ for each $A \sqsubseteq \forall r.B \in \mathcal{T}$
2. $B(x) \leftarrow A_1(x), \ldots, A_n(x)$ for all $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \in \Xi(\mathcal{T})$
3. $r(x, y) \leftarrow r_1(x, y), \ldots, r_n(x, y)$ for all $r_1 \sqcap \ldots \sqcap r_n \sqsubseteq r \in \mathcal{T}$
4. $\bot(x) \leftarrow A(x), r(x, y_1), r(x, y_2), B(y_1), B(y_2), y_1 \neq y_2$
   
   for each $A \sqsubseteq \leq 1 r.B \in \mathcal{T}$

5. $\Gamma \leftarrow A(x), A_1(x), \ldots, A_n(x), r(x, y), B(y)$

   for all $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq \exists(r_1 \sqcap \ldots \sqcap r_m).B_1 \sqcap \ldots \sqcap B_k$ and $A \sqsubseteq \leq 1 r.B$ of $\Xi(\mathcal{T})$ such that $r=r_i$ and $B=B_j$ for some $i, j$ with $\Gamma \in \{B_1(y), \ldots, B_k(y), r_1(x, y), \ldots, r_k(x, y)\}$
Query Answering Algorithm

**Algorithm Horn-$SHIQ$-CQ:**

**Input:** normal Horn-$SHIQ$ KB $\mathcal{O} = (\mathcal{T}, \mathcal{A})$, conjunctive query $Q$

**Output:** query answers

$\Xi(\mathcal{T}) \leftarrow \text{Saturate}(\mathcal{T})$;

$\text{rew}_\mathcal{T}(Q) \leftarrow \text{Rewrite}(Q, \Xi(\mathcal{T}))$;

$\text{cr}(\mathcal{T}) \leftarrow \text{CompletionRules}(\mathcal{T})$;

$P \leftarrow \mathcal{A} \cup \text{cr}(\mathcal{T}) \cup \text{rew}_\mathcal{T}(Q)$;

$\text{ans} \leftarrow \{\vec{u} \mid q(\vec{u}) \in \text{Datalog-eval}(P)\}; \quad \triangleright \text{call Datalog reasoner}$

**Theorem**

*For satisfiable Horn-$SHIQ$ $\mathcal{O}$ in normal form and CQ $Q$, the algorithm Horn-$SHIQ$-CQ outputs $\text{ans}(Q, \mathcal{O})$. It runs (properly implemented) polynomial in data complexity and exponential in combined complexity.*
Further Aspects

- **Beyond UCQs**: Horn-$SHIQ$ rewriting above is extendible to *weakly DL-safe* Datalog programs

- **Tractable combined complexity**: polynomial-time rewriting of acyclic CQs for $DL$-Lite and $ELH$ to Datalog [Bienvenu et al., 2013]
  - rules with bounded number of variables
  - extends to $k$-acyclic CQs (removing $k$ role atoms yields acyclic CQ)

- **Datalog extensions**: datalog with new values (Datalog $\pm$, existential rules)
  - embed ontologies
  - construct forest-shaped model using the *chase* algorithm
  - employ syntactic / semantic conditions

- **Rewriting size**: (nonrecursive) Datalog can be more succinct than FO rewriting (e.g., Gottlob and Schwentick [2011] for $DL$-Lite)

- **Non-uniform rewriting**: modifications of the ABox a la “combined approach” [Kontchakov et al., 2010] (dependent on the TBox but independent of the query)
Conclusion

- Query answering over ontologies is an active area
- Big need: *ontology based data access (OBDA)*
- Query Rewriting to RDMBS technology is a promising approach
- CQ answering over lightweight but also more expressive ontologies can be rewritten to Datalog
- A number of open research issues (theory, systems, applications)
Appendix: Negative Instance Queries in $SROEL(\sqcap, \times)$

- For negative instance queries $\neg C(a)$ use simple reduction:
  $\mathcal{O} \models \neg C(a) \iff \mathcal{O} \cup \{C(a)\}$ is unsatisfiable
  $\iff \mathcal{O} \cup \{C(a)\} \models \bot(o)$ for any individual $o$.

- For instance retrieval of $\neg C(X)$, use an ‘indexed variant’ of $P_{inst}$ for each case ($\mathcal{O} \cup \{C(a)\}$):
  - $isa(X, Y) \leadsto isa_n(X, Y, 'C', a)$
  - similarly $triple(X, Y, Z) \leadsto triple_n(X, Y, Z, 'C', a)$
  - modify $P_{inst}$ to $P_{\neg inst}$
  - use rules special $P_{\neg}$ to test $\mathcal{O} \models \neg C(a)$ via deriving $isnota(a, C)$ from $isa_n(\cdot, \bot, 'C', a)$

- Let $\Phi_{\neg EL}(\mathcal{O}) = P_{\neg inst} \cup I_{inst}(\mathcal{O}) \cup P_{\neg}$

**Theorem**

For every $SROEL(\sqcap, \times)$ ontology $\mathcal{O}$, $\mathcal{O} \models \neg C(a)$ if and only if $\Phi_{\neg EL}(\mathcal{O}) \models isnota(a, C)$. 
Appendix: Horn-\textit{SHIQ} TBox Saturation Rules

\[
\begin{align*}
M \sqsubseteq \exists S.(N \sqcap N') &\quad N \sqsubseteq A & M \sqsubseteq \exists (S \sqcap S').N & S \sqsubseteq r & M \sqsubseteq \exists (S \sqcap S' \sqcap r).N & R_c^- \\
M \sqsubseteq \exists S.(N \sqcap N' \sqcap A) & M \sqsubseteq \exists (S \sqcap S').N & S \sqsubseteq r & M \sqsubseteq \exists (S \sqcap S' \sqcap r).N & R_c^- \\
M \sqsubseteq \exists S.(N \sqcap \bot) & M \sqsubseteq \bot & M \sqsubseteq \exists (S \sqcap r).N & A \sqsubseteq \forall r.B & M \sqsubseteq \exists (S \sqcap r).N \sqcap (N \sqcap B) & R \sqcup \\
M \sqsubseteq \exists (S \sqcap inv(r)).(N \sqcap A) & A \sqsubseteq \forall r.B & M \sqsubseteq B & M \sqsubseteq \exists (S \sqcap inv(r)).(N \sqcap A) & A \sqsubseteq \forall r.B & M \sqsubseteq B & R \sqcup \\
M \sqsubseteq \exists (S \sqcap r).(N \sqcap B) & A \sqsubseteq \leq 1 r.B & M \sqsubseteq \exists (S' \sqcap r).(N' \sqcap B) & M \sqcap M' \sqcap A \sqsubseteq \exists (S \sqcap S' \sqcap r).N \sqcap N' & R \leq \\
M \sqsubseteq \exists (S \sqcap inv(r)).(N_1 \sqcap N_2 \sqcap A) & A \sqsubseteq \leq 1 r.B & N_1 \sqcap A \sqsubseteq \exists (S' \sqcap r).(N' \sqcap B \sqcap C) & M \sqcap B \sqsubseteq C & M \sqcap B \sqsubseteq \exists (S \sqcap inv(S' \sqcap r)).(N_1 \sqcap N_2 \sqcap A) & R \leq \\
\end{align*}
\]

\(M^{(i)}, N^{(i)}, (\text{resp., } S^{(i)})\) are conjunctions of atomic concepts (roles); \(A, B\) are atomic concepts
Meghyn Bienvenu, Magdalena Ortiz, Mantas Simkus, and Guohui Xiao.
Tractable queries for lightweight description logics.

Loris Bozzato and Luciano Serafini.
Materialization calculus for contexts in the semantic web.

Diego Calvanese, Thomas Eiter, and Magdalena Ortiz.
Answering regular path queries in expressive description logics: An automata-theoretic approach.

Diego Calvanese, Thomas Eiter, and Magdalena Ortiz.
Regular path queries in expressive description logics with nominals.
Thomas Eiter, Georg Gottlob, Magdalena Ortiz, and Mantas Šimkus. Query answering in the description logic Horn-SHIQ.

Thomas Eiter, Carsten Lutz, Magdalena Ortiz, and Mantas Šimkus. Query answering in description logics with transitive roles.

Thomas Eiter, Magdalena Ortiz, Mantas Šimkus, Kien Trung-Tran, and Guohui Xiao. Query rewriting for Horn-SHIQ plus rules.

Thomas Eiter, Magdalena Ortiz, Mantas Šimkus, Kien Trung-Tran, and Guohui Xiao. Towards practical query answering for Horn-SHIQ.
Georg Gottlob and Thomas Schwentick.
Rewriting ontological queries into small nonrecursive datalog programs.
In Rosati et al. [2011].

Stijn Heymans, Thomas Eiter, and Guohui Xiao.
Tractable reasoning with DL-programs over Datalog-rewritable description logics.

Ullrich Hustadt, Boris Motik, and Ulrike Sattler.
A decomposition rule for decision procedures by resolution-based calculi.

Y. Kazakov.
Consequence-driven reasoning for Horn $SHIQ$ ontologies.

Stanislav Kikot, Roman Kontchakov, and Michael Zakharyaschev.
On (in)tractability of OBDA with OWL 2 QL.
In Rosati et al. [2011].
Roman Kontchakov, Carsten Lutz, David Toman, Frank Wolter, and Michael Zakharyaschev. The combined approach to query answering in dl-lite.


Markus Krötzsch. Efficient inferencing for OWL EL.

Markus Krötzsch. Efficient rule-based inferencing for OWL EL.

Alon Y. Levy and Marie-Christine Rousset. Combining Horn rules and description logics in CARIN.


