UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

Ontology-Based Data Access

Lecture 8: DL-Lite, Rewriting, Unfolding

13 December, 2017

Foundations of Ontologies and Databases for Information Systems

CS5130 (Winter 17/18)

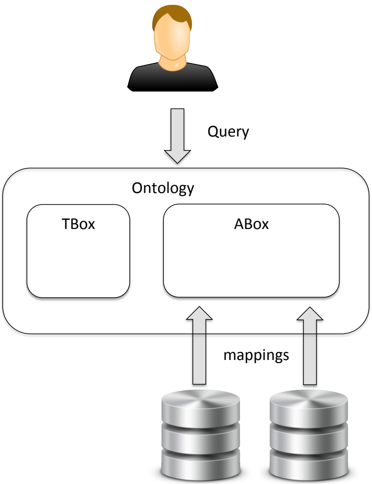
Recap of Lecture 7

Ontology-Based Data Access

- Use ontologies as interface

  ...
- to access (here: query)
- data stored in some format

  ...
- using mappings



- Talked about description logics as ontology representation language
- Semantics
- TODO: Tableaux Calculus

References

- Reasoning Web Summer School 2014 course by Kontchakov on Description Logics

  http://rw2014.di.uoa.gr/sites/default/files/slides/An_Introduction_to_Description_Logics.pdf

- Lecture notes by Calvanese in 2013/2014 course on Ontology and Database Systems

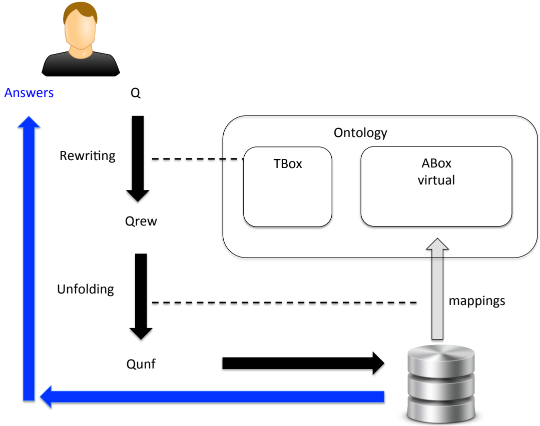
  https://www.inf.unibz.it/~calvanese/teaching/14-15-odbs/lecture-notes/

- Parts of Reasoning Web Summer School 2014 course by Ö. on Ontology-Based Data Access on Temporal and Streaming Data

  http://rw2014.di.uoa.gr/sites/default/files/slides/Ontology_Based_Data_Access_on_Temporal_and_Streaming_Data.pdf

OBDA in the Classical Sense

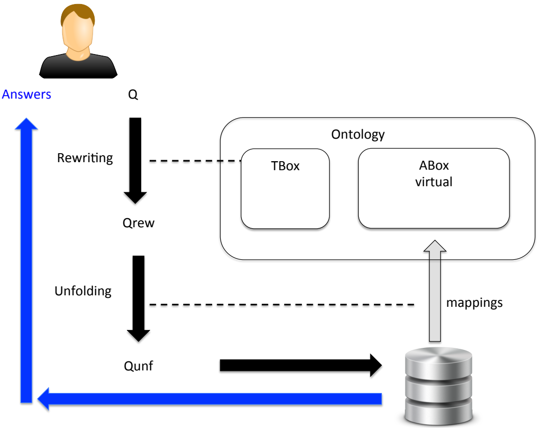
- Keep the data where they are because of large volume
- ABox not loaded into main memory, kept virtual



Rewriting

OBDA in the Classical Sense

- Query answering not with deduction but rewriting and unfolding
- Challenge: Complete and correct rewriting and unfolding



Definition

- $\mathcal{L}_{TBox}$ = Tbox language
- $\mathcal{L}_{oQ}$ = ontology query language
- $\mathcal{L}_{tQ}$ = target query language

Answering $\mathcal{L}_{TBox}$ queries is $\mathcal{L}_{tQ}$-rewritable iff for every TBox $\mathcal{T}$ over $\mathcal{L}_{TBox}$ and query Q in $\mathcal{L}_{oQ}$ there is a query Q_{rew} in $\mathcal{L}_{tQ}$ such that for all ABoxes $\mathcal{A}$:

$$cert(Q, \mathcal{T} \cup \mathcal{A}) = ans(Q_{rew}, DB(\mathcal{A}))$$

Definition

- $\mathcal{L}_{TBox}$ = Tbox language
- $\mathcal{L}_{oQ}$ = ontology query language
- $\mathcal{L}_{tQ}$ = target query language

Answering $\mathcal{L}_{TBox}$ queries is $\mathcal{L}_{tQ}$-rewritable iff for every TBox $\mathcal{T}$ over $\mathcal{L}_{TBox}$ and query Q in $\mathcal{L}_{oQ}$ there is a query Q_{rew} in $\mathcal{L}_{tQ}$ such that for all ABoxes $\mathcal{A}$:

$$cert(Q, \mathcal{T} \cup \mathcal{A}) = ans(Q_{rew}, DB(\mathcal{A}))$$

Definition (Minimal Herband Model $DB(\mathcal{A})$)

$DB(\mathcal{A}) = (\Delta, \cdot^{\mathcal{I}})$ for an Abox $\mathcal{A}$ with

- Δ = set of constants occurring in $\mathcal{A}$
- $c^{\mathcal{I}} = c$ for all constants;
- $A^{\mathcal{I}} = \{c \mid A(c) \in \mathcal{A}\}$;
- $r^{\mathcal{I}} = \{(c, d) \mid R(c, d) \in \mathcal{A}\}$

Rewriting for Different Languages

- Possibility of rewriting depends on expressivity balance between $\mathcal{L}_{TBox}$, $\mathcal{L}_{oQ}$, $\mathcal{L}_{tQ}$.
- One aims at computably feasible $\mathcal{L}_{tQ}$ queries
- In classical OBDA
  - $\mathcal{L}_{TBox}$: Language of the DL-Lite family
  - $\mathcal{L}_{oQ}$: Unions of conjunctive queries
  - $\mathcal{L}_{tQ}$: (Safe) FOL/SQL (in AC^0)

DL-Lite

DL-Lite

- Family of DLs underlying the OWL 2 QL profile
- Tailored towards the classical OBDA scenario
  - Captures (a large fragment of) UML
  - FOL-rewritability for ontology satisfiability checking and query answerings for UCQs
  - Used in many implementations of OBDA (QuOnto, Presto, Rapid, Nyaya, ontop etc.)
- We give a rough overview. For details consult, e.g.,

  Lit: Calvanese et al. Ontologies and databases: The DL-Lite approach. In Tessaris/Franconi, editors, Semantic Technologies for Informations Systems. 5th Int. Reasoning Web Summer School (RW 2009), pages 255–356. Springer, 2009.

  Lit: A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. J. Artif. Intell. Res. (JAIR), 36:1–69, 2009.

DL-Lite$_\mathcal{F}$

- Simple member of the family allowing functional constraints
- Syntax
  - Basic role $Q ::= P \mid P^-$ for $P \in N_R$
  - Roles: $R ::= Q \mid \neg Q$.
  - Basic concepts $B ::= A \mid \exists Q$ for $A \in N_C, Q \in N_R$
  - Concepts $C ::= B \mid \neg B \mid \exists R.C$
  - TBox: $B \sqsubseteq C$, $(func\ Q)$ ("Q is functional")
    (where $(func\ Q)$ is allowed in the TBox only if Q does not appear as $\exists Q.C$ on a rhs in the TBox)
  - ABox: $A(a), P(a, b)$
- Semantics as usual
  ($\exists Q$ shorthand for $\exists Q.\top$)
- Note
  - No qualified existential on lhs
  - Restriction on function role
  - Both due to rewritability

DL-Lite$_{\mathcal{F}}$

- Simple member of the family allowing functional constraints

- Syntax
  - Basic role $Q ::= P \mid P^-$ for $P \in N_R$
  - Roles: $R ::= Q \mid \neg Q$.
  - Basic concepts $B ::= A \mid \exists Q$ for $A \in N_C, Q \in N_R$
  - Concepts $C ::= B \mid \neg B \mid \exists R.C$

  - TBox: $B \sqsubseteq C$, $(func\ Q)$ ("Q is functional")
    (where $(func\ Q)$ is allowed in the TBox only if Q does not appear as $\exists Q.C$ on a rhs in the TBox)
  - ABox: $A(a), P(a, b)$
- Semantics as usual
  ($\exists Q$ shorthand for $\exists Q.\top$)

- Note
  - No qualified existential on lhs
  - Restriction on function role
  - Both due to rewritability

DL-Lite$_\mathcal{F}$

- Simple member of the family allowing functional constraints

- Syntax
  - Basic role $Q ::= P \mid P^-$ for $P \in N_R$
  - Roles: $R ::= Q \mid \neg Q$.
  - Basic concepts $B ::= A \mid \exists Q$ for $A \in N_C, Q \in N_R$
  - Concepts $C ::= B \mid \neg B \mid \exists R.C$

  - TBox: $B \sqsubseteq C$, $(func\ Q)$ ("Q is functional")
    (where $(func\ Q)$ is allowed in the TBox only if Q does not appear as $\exists Q.C$ on a rhs in the TBox)
  - ABox: $A(a), P(a, b)$
- Semantics as usual
  ($\exists Q$ shorthand for $\exists Q.\top$)

- Note
  - No qualified existential on lhs
  - Restriction on function role
  - Both due to rewritability

Properties

- DL-Lite$_{\mathcal{F}}$ enables basic UML conceptual modeling
  - ISA between classes ($\mathit{Professor} \sqsubseteq \mathit{Person}$)
  - Disjointness ($\mathit{Professor} \sqsubseteq \neg \mathit{Student}$)
  - Domain and range of roles: ($\mathit{Professor} \sqsubseteq \exists \mathit{teachesTo}$, $\exists \mathit{hasTutor}^- \sqsubseteq \mathit{Professor}$)
  - ...
- DL-Lite$_{\mathcal{F}}$ does not have finite model property

Example

- $\mathit{Nat} \sqsubseteq \exists \mathit{hasSucc}$, $\exists \mathit{hasSucc}^- \sqsubseteq \mathit{Nat}$, $(\mathit{funct}\ \mathit{hasSucc}^-)$,
- $\mathit{Zero} \sqsubseteq \mathit{Nat}$, $\mathit{Zero} \sqsubseteq \neg \exists \mathit{hasSucc}^-$, $\mathit{Zero}(0)$

  (see Exercise 8.1)

DL-Lite$_{\mathcal{R}}$

- Another simple member of the family; allows role hierarchies

- Syntax
  - Basic role $Q ::= P \mid P^-$ for $P \in N_R$
  - Roles $R ::= Q \mid \neg Q$.
  - Basic concepts $B ::= A \mid \exists Q$ for $A \in N_C, Q \in N_R$
  - Concepts $C ::= B \mid \neg B \mid \exists R.C$

  - TBox: $B \sqsubseteq C$, $R_1 \sqsubseteq R_2$
  - ABox: $A(a), P(a,b)$
- Semantics as usual

- Note
  - Again no qualified existential on lhs
  - DL-Lite$_{\mathcal{R}}$ has finite model property

Qualified Existentials

- Qualified existentials on rhs not necessary (if role inclusions and inverse allowed)
- Can be eliminated preserving satisfiably equivalence

Example

- Input: $Student \sqsubseteq \exists hasTutor.Professor$
- Output
  - $hasProfTutor \sqsubseteq hasTutor$
  - $Student \sqsubseteq \exists hasProfTutor$
  - $\exists hasProfTutor^{-} \sqsubseteq Prof$

- In the following: We assume w.l.o.g. that only non-qualified existentials are used (normalization)

DL-Lite$_{\mathcal{A}}$

- DL-Lite$_{\mathcal{A}}$ extends DL-Lite$_{\mathcal{F}}$ and DL-Lite$_{\mathcal{R}}$ by allowing for
  - attribute expressions (relation between objects and values)
  - identification assertions (corresponds to primary key constraints in DB)

- Restrictions for TBox: Roles (and attributes) appearing in functionality declarations or identification assertions must not appear on the rhs of role inclusions

Example (Football league example in DL-Lite$_{\mathcal{A}}$)

- $League \sqsubseteq \exists of$ ("Every league is the league .. .
- $\exists of^{-} \sqsubseteq Nation$ .. of some nation")

- $League \sqsubseteq \delta(hasYear)$ ("Every league has a year")
  (Here: $\delta(hasYear) =$ domain of attribute $hasYear$)
- $\rho(hasYear) \sqsubseteq xsd : positiveInteger$
  ("Range of hasYear are RDF literals of type positive integer")

- $(funct\ hasYear)$

- $(id\ League\ of, hasYear)$
  ("Leagues are uniquely determined by the nation and the year")
  General Form: $(id\ basicConcept\ path_1, \ldots, path_n)$) instances i, i' are

Identity assertions

- Path: $\pi \longrightarrow S|D?|\pi \circ \pi$
  - $S =$ basic role, atomic attribute (or inverse o atomic attribute)
  - $\circ =$ composition of paths
  - $D =$ basic concept or value domain
  - $?D =$ testing relation = identity on instances of D

- $fillers_\pi(i) =$ objects reachable from i via π
  $HAS - CHILD \circ Woman? =$ path connecting objects i with his/her daughters (its fillers)

- Identity assertions

$$(id\ B\ \pi_1, \dots, \pi_n))$$

Semantics: Different instances i, i' of B are distinguished by at least one of their fillers: There is π_j such that

$$fillers_{\pi_j}(i) \neq fillers_{\pi_j}(i')$$

Identity assertions

- Path: $\pi \longrightarrow S|D?|\pi \circ \pi$
  - $S =$ basic role, atomic attribute (or inverse o atomic attribute)
  - $\circ =$ composition of paths
  - $D =$ basic concept or value domain
  - $?D =$ testing relation = identity on instances of D

- $fillers_\pi(i)$ = objects reachable from i via π
  $HAS - CHILD \circ Woman?$ = path connecting objects i with his/her daughters (its fillers)

- Identity assertions

$$(id\ B\ \pi_1, \ldots, \pi_n))$$

  Semantics: Different instances i, i' of B are distinguished by at least one of their fillers: There is π_j such that

$$fillers_{\pi_j}(i) \neq fillers_{\pi_j}(i')$$

Rewritability of Query Answering

- UCQ over DL-Lite$_{\mathcal{A}}$ can be rewritten into FOL queries

Theorem

UCQs over DL-Lite$_{\mathcal{A}}$ are FOL-rewritable.

- We consider first the case where the ontology is satisfiable
- In this case rewriting is possible even into UCQs

- And in this case only positive inclusions (PIs) and not negative inclusions (NIs) are relevant for rewriting
  - PI: $A_1 \sqsubseteq A_2$, $\exists Q \sqsubseteq A_2$, $A_1 \sqsubseteq \exists Q_2$, $\exists Q_1 \sqsubseteq \exists Q_2$, $Q_1 \sqsubseteq Q_2$
  - NI: $A_1 \sqsubseteq \neg A_2$, $\exists Q_1 \sqsubseteq \neg A_2$, $A_1 \sqsubseteq \neg \exists Q_2$, $\exists Q_1 \sqsubseteq \neg \exists Q_2$, $Q_1 \sqsubseteq \neg Q_2$

Rewritability of Query Answering

- UCQ over DL-Lite$_{\mathcal{A}}$ can be rewritten into FOL queries

Theorem

UCQs over DL-Lite$_{\mathcal{A}}$ are FOL-rewritable.

- We consider first the case where the ontology is satisfiable
- In this case rewriting is possible even into UCQs

- And in this case only positive inclusions (PIs) and not negative inclusions (NIs) are relevant for rewriting
  - PI: $A_1 \sqsubseteq A_2$, $\exists Q \sqsubseteq A_2$, $A_1 \sqsubseteq \exists Q_2$, $\exists Q_1 \sqsubseteq \exists Q_2$, $Q_1 \sqsubseteq Q_2$
  - NI: $A_1 \sqsubseteq \neg A_2$, $\exists Q_1 \sqsubseteq \neg A_2$, $A_1 \sqsubseteq \neg \exists Q_2$, $\exists Q_1 \sqsubseteq \neg \exists Q_2$, $Q_1 \sqsubseteq \neg Q_2$

Example (Query answering by rewriting)

- $AssistantProf \sqsubseteq Prof$
- $\exists teaches^{-} \sqsubseteq Course$
- $Prof \sqsubseteq \exists teaches$

- $Prof(schroedinger)$
- $teaches(schroedinger, csCats)$
- $Course(csCats)$
- $Prof(einstein)$

$Q(x) = \exists y.teaches(x, y) \wedge Course(y)$

- QA by stepwise extension of the initial query
- Capture entailments of PIs in order to find also binding $x = einstein$
- Read PIs as rules applied from right to left

Example (Query answering by rewriting)

- $AssistantProf \sqsubseteq Prof$
- $\exists teaches^- \sqsubseteq Course$
- $Prof \sqsubseteq \exists teaches$

- $Prof(schroedinger)$
- $teaches(schroedinger, csCats)$
- $Course(csCats)$
- $Prof(einstein)$

$Q(x) = \exists y.teaches(x, y) \wedge Course(y)$

- $Q_{rew}(x) \leftarrow teaches(x, y), Course(y)$
- $Q_{rew}(x) \leftarrow teaches(x, y), teaches(\_, y)$
- $Q_{rew}(x) \leftarrow teaches(x, y)$ (after unification/reduction)
- $Q_{rew}(x) \leftarrow teaches(x, \_)$ (after anonymization)
- $Q_{rew}(x) \leftarrow Prof(x)$
- $Q_{rew}(x) \leftarrow AssistantProf(x)$

- Resulting query Q_{rew} is an UCQ and is called the perfect rewriting of Q
- $ans(Q_{rew}, DB(\mathcal{A})) = \{schroedinger, einstein\} = cert(Q, (\mathcal{T}, \mathcal{A}))$

Example (Query answering by rewriting)

- $AssistantProf \sqsubseteq Prof$
- $\exists teaches^- \sqsubseteq Course$
- $Prof \sqsubseteq \exists teaches$

- $Prof(schroedinger)$
- $teaches(schroedinger, csCats)$
- $Course(csCats)$
- $Prof(einstein)$

$Q(x) = \exists y.teaches(x, y) \wedge Course(y)$

- $Q_{rew}(x) \leftarrow teaches(x, y), Course(y)$
- $Q_{rew}(x) \leftarrow teaches(x, y), teaches(\_, y)$
- $Q_{rew}(x) \leftarrow teaches(x, y)$ (after unification/reduction)
- $Q_{rew}(x) \leftarrow teaches(x, \_)$ (after anonymization)
- $Q_{rew}(x) \leftarrow Prof(x)$
- $Q_{rew}(x) \leftarrow AssistantProf(x)$

- Resulting query Q_{rew} is an UCQ and is called the perfect rewriting of Q
- $ans(Q_{rew}, DB(\mathcal{A})) = \{schroedinger, einstein\} = cert(Q, (\mathcal{T}, \mathcal{A}))$

Example (Query answering by rewriting)

- $AssistantProf \sqsubseteq Prof$
- $\exists teaches^{-} \sqsubseteq Course$
- $Prof \sqsubseteq \exists teaches$

- $Prof(schroedinger)$
- $teaches(schroedinger, csCats)$
- $Course(csCats)$
- $Prof(einstein)$

$Q(x) = \exists y.teaches(x, y) \wedge Course(y)$

- $Q_{rew}(x) \leftarrow teaches(x, y), Course(y)$
- $Q_{rew}(x) \leftarrow teaches(x, y), teaches(\_, y)$
- $Q_{rew}(x) \leftarrow teaches(x, y)$ (after unification/reduction)
- $Q_{rew}(x) \leftarrow teaches(x, \_)$ (after anonymization)
- $Q_{rew}(x) \leftarrow Prof(x)$
- $Q_{rew}(x) \leftarrow AssistantProf(x)$

- Resulting query Q_{rew} is an UCQ and is called the perfect rewriting of Q
- $ans(Q_{rew}, DB(\mathcal{A})) = \{schroedinger, einstein\} = cert(Q, (\mathcal{T}, \mathcal{A}))$

Example (Query answering by rewriting)

- $AssistantProf \sqsubseteq Prof$
- $\exists teaches^{-} \sqsubseteq Course$
- $Prof \sqsubseteq \exists teaches$

- $Prof(schroedinger)$
- $teaches(schroedinger, csCats)$
- $Course(csCats)$
- $Prof(einstein)$

$Q(x) = \exists y.teaches(x, y) \wedge Course(y)$

- $Q_{rew}(x) \leftarrow teaches(x, y), Course(y)$
- $Q_{rew}(x) \leftarrow teaches(x, y), teaches(\_, y)$
- $Q_{rew}(x) \leftarrow teaches(x, y)$ (after unification/reduction)
- $Q_{rew}(x) \leftarrow teaches(x, \_)$ (after anonymization)
- $Q_{rew}(x) \leftarrow Prof(x)$
- $Q_{rew}(x) \leftarrow AssistantProf(x)$

- Resulting query Q_{rew} is an UCQ and is called the perfect rewriting of Q
- $ans(Q_{rew}, DB(\mathcal{A})) = \{schroedinger, einstein\} = cert(Q, (\mathcal{T}, \mathcal{A}))$

Example (Query answering by rewriting)

- $AssistantProf \sqsubseteq Prof$
- $\exists teaches^{-} \sqsubseteq Course$
- $Prof \sqsubseteq \exists teaches$

- $Prof(schroedinger)$
- $teaches(schroedinger, csCats)$
- $Course(csCats)$
- $Prof(einstein)$

$Q(x) = \exists y.teaches(x, y) \wedge Course(y)$

- $Q_{rew}(x) \leftarrow teaches(x, y), Course(y)$
- $Q_{rew}(x) \leftarrow teaches(x, y), teaches(\_, y)$
- $Q_{rew}(x) \leftarrow teaches(x, y)$ (after unification/reduction)
- $Q_{rew}(x) \leftarrow teaches(x, \_)$ (after anonymization)
- $Q_{rew}(x) \leftarrow Prof(x)$
- $Q_{rew}(x) \leftarrow AssistantProf(x)$

- Resulting query Q_{rew} is an UCQ and is called the perfect rewriting of Q
- $ans(Q_{rew}, DB(\mathcal{A})) = \{schroedinger, einstein\} = cert(Q, (\mathcal{T}, \mathcal{A}))$

Example (Query answering by rewriting)

- $AssistantProf \sqsubseteq Prof$
- $\exists teaches^{-} \sqsubseteq Course$
- $Prof \sqsubseteq \exists teaches$

- $Prof(schroedinger)$
- $teaches(schroedinger, csCats)$
- $Course(csCats)$
- $Prof(einstein)$

$Q(x) = \exists y.teaches(x, y) \land Course(y)$

- $Q_{rew}(x) \leftarrow teaches(x, y), Course(y)$
- $Q_{rew}(x) \leftarrow teaches(x, y), teaches(\_, y)$
- $Q_{rew}(x) \leftarrow teaches(x, y)$ (after unification/reduction)
- $Q_{rew}(x) \leftarrow teaches(x, \_)$ (after anonymization)
- $Q_{rew}(x) \leftarrow Prof(x)$
- $Q_{rew}(x) \leftarrow AssistantProf(x)$

- Resulting query Q_{rew} is an UCQ and is called the perfect rewriting of Q
- $ans(Q_{rew}, DB(\mathcal{A})) = \{schroedinger, einstein\} = cert(Q, (\mathcal{T}, \mathcal{A}))$

Example (Query answering by rewriting)

- $AssistantProf \sqsubseteq Prof$
- $\exists teaches^- \sqsubseteq Course$
- $Prof \sqsubseteq \exists teaches$
- $Prof(schroedinger)$
- $teaches(schroedinger, csCats)$
- $Course(csCats)$
- $Prof(einstein)$

$Q(x) = \exists y.teaches(x, y) \land Course(y)$

- $Q_{rew}(x) \leftarrow teaches(x, y), Course(y)$
- $Q_{rew}(x) \leftarrow teaches(x, y), teaches(\_, y)$
- $Q_{rew}(x) \leftarrow teaches(x, y)$ (after unification/reduction)
- $Q_{rew}(x) \leftarrow teaches(x, \_)$ (after anonymization)
- $Q_{rew}(x) \leftarrow Prof(x)$
- $Q_{rew}(x) \leftarrow AssistantProf(x)$

- Resulting query Q_{rew} is an UCQ and is called the perfect rewriting of Q
- $ans(Q_{rew}, DB(\mathcal{A})) = \{schroedinger, einstein\} = cert(Q, (\mathcal{T}, \mathcal{A}))$

Example (Query answering by rewriting)

- $AssistantProf \sqsubseteq Prof$
- $\exists teaches^- \sqsubseteq Course$
- $Prof \sqsubseteq \exists teaches$

- $Prof(schroedinger)$
- $teaches(schroedinger, csCats)$
- $Course(csCats)$
- $Prof(einstein)$

$Q(x) = \exists y.teaches(x, y) \wedge Course(y)$

- $Q_{rew}(x) \leftarrow teaches(x, y), Course(y)$
- $Q_{rew}(x) \leftarrow teaches(x, y), teaches(\_, y)$
- $Q_{rew}(x) \leftarrow teaches(x, y)$ (after unification/reduction)
- $Q_{rew}(x) \leftarrow teaches(x, \_)$ (after anonymization)
- $Q_{rew}(x) \leftarrow Prof(x)$
- $Q_{rew}(x) \leftarrow AssistantProf(x)$

- Resulting query Q_{rew} is an UCQ and is called the perfect rewriting of Q
- $ans(Q_{rew}, DB(\mathcal{A})) = \{schroedinger, einstein\} = cert(Q, (\mathcal{T}, \mathcal{A}))$

Example (Query answering by rewriting)

- $AssistantProf \sqsubseteq Prof$
- $\exists teaches^{-} \sqsubseteq Course$
- $Prof \sqsubseteq \exists teaches$

- $Prof(schroedinger)$
- $teaches(schroedinger, csCats)$
- $Course(csCats)$
- $Prof(einstein)$

$Q(x) = \exists y.teaches(x, y) \land Course(y)$

- $Q_{rew}(x) \leftarrow teaches(x, y), Course(y)$
- $Q_{rew}(x) \leftarrow teaches(x, y), teaches(\_, y)$
- $Q_{rew}(x) \leftarrow teaches(x, y)$ (after unification/reduction)
- $Q_{rew}(x) \leftarrow teaches(x, \_)$ (after anonymization)
- $Q_{rew}(x) \leftarrow Prof(x)$
- $Q_{rew}(x) \leftarrow AssistantProf(x)$

- Resulting query Q_{rew} is an UCQ and is called the perfect rewriting of Q
- $ans(Q_{rew}, DB(\mathcal{A})) = \{schroedinger, einstein\} = cert(Q, (\mathcal{T}, \mathcal{A}))$

Example (Query answering by rewriting)

- $AssistantProf \sqsubseteq Prof$
- $\exists teaches^{-} \sqsubseteq Course$
- $Prof \sqsubseteq \exists teaches$

- $Prof(schroedinger)$
- $teaches(schroedinger, csCats)$
- $Course(csCats)$
- $Prof(einstein)$

$Q(x) = \exists y.teaches(x, y) \land Course(y)$

- $Q_{rew}(x) \leftarrow teaches(x, y), Course(y)$
- $Q_{rew}(x) \leftarrow teaches(x, y), teaches(\_, y)$
- $Q_{rew}(x) \leftarrow teaches(x, y)$ (after unification/reduction)
- $Q_{rew}(x) \leftarrow teaches(x, \_)$ (after anonymization)
- $Q_{rew}(x) \leftarrow Prof(x)$
- $Q_{rew}(x) \leftarrow AssistantProf(x)$

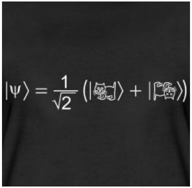
- Resulting query Q_{rew} is an UCQ and is called the perfect rewriting of Q
- $ans(Q_{rew}, DB(\mathcal{A})) = \{schroedinger, einstein\} = cert(Q, (\mathcal{T}, \mathcal{A}))$

Example (Query answering by rewriting)

- $AssistantProf \sqsubseteq Prof$
- $\exists teaches^{-} \sqsubseteq Course$
- $Prof \sqsubseteq \exists teaches$

- $Prof(schroedinger)$
- $teaches(schroedinger, csCats)$
- $Course(csCats)$
- $Prof(einstein)$

$Q(x) = \exists y.teaches(x, y) \land Course(y)$



Perfect Rewriting Algorithm PerfectRew(Q, TP)

```
Input  : Q = UCQ (in set notation), TP = DL-Lite_A PIs
Output: union of conjunctive queries PR
PR := Q;
repeat
    PR' := PR;
    forall the q ∈ PR' do
        forall the g ∈ q do
            forall the PI I ∈ TP do
                if I is applicable to g then
                    PR := PR ∪ {ApplyPI(q, g, I)}
                end
            end
        end
        forall the g1, g2 in q do
            if g1 and g2 unify then
                PR := PR ∪ {anon(reduce(q, g1, g2))};
            end
        end
    end
until PR' = PR;
return PR;
```

Procedure $ApplyPI(q, g, I)$

- Applicability condition
  - A PI I is applicable to atom $A(x)$, if I has A in rhs.
  - A PI I is applicable to atom $P(x1, x_2)$, if one of the following conditions holds:
    1. $x_2 = \_$ and rhs of I is $\exists P$ or
    2. $x_1 = \_$ and the rhs of I is $\exists P^-$; or
    3. I is a role inclusion assertion and rhs is either P or P^-
- Outcome of application

Atom g	PI I	$gr(g, I)$
$A(x)$	$A1 \sqsubseteq A$	$A1(x)$
$A(x)$	$\exists P \sqsubseteq A$	$P(x, \_)$
$A(x)$	$\exists P^- \sqsubseteq A$	$P(\_, x)$
$P(x, \_)$	$A \sqsubseteq \exists P$	$A(x)$
$P(x, \_)$	$\exists P1 \sqsubseteq \exists P$	$P1(x, \_)$
$P(x, \_)$	$\exists P1^- \sqsubseteq \exists P$	$P1(\_, x)$
$P(\_, x)$	$A \sqsubseteq \exists P^-$	$A(x)$
$P(\_, x)$	$\exists P1 \sqsubseteq \exists P^-$	$P1(x, \_)$
$P(\_, x)$	$\exists P1^- \sqsubseteq \exists P^-$	$P1(\_, x)$
$P(x_1, x_2)$	$\exists P1 \sqsubseteq P$ or $\exists P1^- \sqsubseteq P^-$	$P1(x_1, x_2)$
$P(x_1, x_2)$	$\exists P1 \sqsubseteq P^-$ or $\exists P1^- \sqsubseteq P$	$P1(x_2, x_1)$

- $ApplyPI(q, g, I) = q[g/gr(g, I)]$

Procedure $ApplyPI(q, g, I)$

- Applicability condition
  - A PI I is applicable to atom $A(x)$, if I has A in rhs.
  - A PI I is applicable to atom $P(x1, x_2)$, if one of the following conditions holds:
    1. $x_2 = \_$ and rhs of I is $\exists P$ or
    2. $x_1 = \_$ and the rhs of I is $\exists P^-$; or
    3. I is a role inclusion assertion and rhs is either P or P^-
- Outcome of application

Atom g	PI I	$gr(g, I)$
$A(x)$	$A1 \sqsubseteq A$	$A1(x)$
$A(x)$	$\exists P \sqsubseteq A$	$P(x, \_)$
$A(x)$	$\exists P^- \sqsubseteq A$	$P(\_, x)$
$P(x, \_)$	$A \sqsubseteq \exists P$	$A(x)$
$P(x, \_)$	$\exists P1 \sqsubseteq \exists P$	$P1(x, \_)$
$P(x, \_)$	$\exists P1^- \sqsubseteq \exists P$	$P1(\_, x)$
$P(\_, x)$	$A \sqsubseteq \exists P^-$	$A(x)$
$P(\_, x)$	$\exists P1 \sqsubseteq \exists P^-$	$P1(x, \_)$
$P(\_, x)$	$\exists P1^- \sqsubseteq \exists P^-$	$P1(\_, x)$
$P(x_1, x_2)$	$\exists P1 \sqsubseteq P$ or $\exists P1^- \sqsubseteq P^-$	$P1(x_1, x_2)$
$P(x_1, x_2)$	$\exists P1 \sqsubseteq P^-$ or $\exists P1^- \sqsubseteq P$	$P1(x_2, x_1)$

- $ApplyPI(q, g, I) = q[g/gr(g, I)]$

Procedure $ApplyPI(q, g, I)$

- Applicability condition
  - A PI I is applicable to atom $A(x)$, if I has A in rhs.
  - A PI I is applicable to atom $P(x1, x_2)$, if one of the following conditions holds:
    1. $x_2 = \_$ and rhs of I is $\exists P$ or
    2. $x_1 = \_$ and the rhs of I is $\exists P^-$; or
    3. I is a role inclusion assertion and rhs is either P or P^-
- Outcome of application

Atom g	PI I	$gr(g, I)$
$A(x)$	$A1 \sqsubseteq A$	$A1(x)$
$A(x)$	$\exists P \sqsubseteq A$	$P(x, \_)$
$A(x)$	$\exists P^- \sqsubseteq A$	$P(\_, x)$
$P(x, \_)$	$A \sqsubseteq \exists P$	$A(x)$
$P(x, \_)$	$\exists P1 \sqsubseteq \exists P$	$P1(x, \_)$
$P(x, \_)$	$\exists P1^- \sqsubseteq \exists P$	$P1(\_, x)$
$P(\_, x)$	$A \sqsubseteq \exists P^-$	$A(x)$
$P(\_, x)$	$\exists P1 \sqsubseteq \exists P^-$	$P1(x, \_)$
$P(\_, x)$	$\exists P1^- \sqsubseteq \exists P^-$	$P1(\_, x)$
$P(x_1, x_2)$	$\exists P1 \sqsubseteq P$ or $\exists P1^- \sqsubseteq P^-$	$P1(x_1, x_2)$
$P(x_1, x_2)$	$\exists P1 \sqsubseteq P^-$ or $\exists P1^- \sqsubseteq P$	$P1(x_2, x_1)$

- $ApplyPI(q, g, I) = q[g/gr(g, I)]$

Anonymization and Reduction

- Reduction $reduce(q, g1, g2)$
  - Input: $g1, g2$ atoms in boy of CQ q
  - Output: Returns a CQ $q?$ obtained by applying to q the most general unifier between $g1$ and $g2$
  - Required to generating possibly unbound variables

- Anonymization
  - Substitute variables that are not bound with _.
  - Variable is bound iff it is a distinguished variable or occurs at least twice in the body of a CQ

Properties of PerfectRew

- Termination
  - There are only finitely many different rewritings
- Correctness
  - Only certain answers are produced by the rewriting
  - Formally: $ans(Q_{rew}, \mathcal{A}) \subseteq cert(Q, (\mathcal{T}, \mathcal{A}))$
  - Clear, as PI applied correctly
- Completeness
  - All certain answers are produced by the rewriting
  - $ans(Q_{rew}, \mathcal{A}) \supseteq cert(Q, (\mathcal{T}, \mathcal{A}))$
  - How to prove this?
    $\Longrightarrow$ Our old friend, the chase, helps again

Chase Construction for DL

- The PIs of the TBox are read as (TGD) rules in the natural direction from left to right

- Resulting structure, the chase, also called canonical model here is universal
- Reminder: A universal model can be mapped homomorphically into any other model.

Theorem

Every satisfiable DL-Lite ontology has a canonical model

- Different from the approach in Date Exchange, one does not aim for finite chases (cannot be guaranteed)
- Chase used as tool for proving, e.g., completeness: Answering the rewritten query Q_{rew} on the minimal Herbrand model of the ABox is the same as answering Q on the chase.

Chase Construction for DL

- The PIs of the TBox are read as (TGD) rules in the natural direction from left to right
- Resulting structure, the chase, also called canonical model here is universal
- Reminder: A universal model can be mapped homomorphically into any other model.

Theorem

Every satisfiable DL-Lite ontology has a canonical model

- Different from the approach in Date Exchange, one does not aim for finite chases (cannot be guaranteed)
- Chase used as tool for proving, e.g., completeness: Answering the rewritten query Q_{rew} on the minimal Herbrand model of the ABox is the same as answering Q on the chase.

Satisfiability Check for Ontologies

- In case an ontology is unsatisfiable, answer set becomes trivial: An unsatisfiable ontology entails all assertions
- Hence to determine correct answers, a satisfiability check is needed

Theorem

Checking (un-)satisfiability of DL-Lite ontologies is FOL rewritable.

That means: For any TBox there is a Boolean query Q such that for all ABoxes $\mathcal{A}$: $(\mathcal{T}, \mathcal{A})$ is satisfiable iff Q is false.

- Unsatisfiability may be caused by an NI (negative inclusion) or by a functional declaration
- So the rewritten query asks for an object in the ABox violating an NI or a functional declaration

Satisfiability Check for Ontologies

- In case an ontology is unsatisfiable, answer set becomes trivial: An unsatisfiable ontology entails all assertions
- Hence to determine correct answers, a satisfiability check is needed

Theorem

Checking (un-)satisfiability of DL-Lite ontologies is FOL rewritable.

That means: For any TBox there is a Boolean query Q such that for all ABoxes $\mathcal{A}$: $(\mathcal{T}, \mathcal{A})$ is satisfiable iff Q is false.

- Unsatisfiability may be caused by an NI (negative inclusion) or by a functional declaration
- So the rewritten query asks for an object in the ABox violating an NI or a functional declaration

FOL Rewritability of Satisfiability

Example

TBox	*ABox*
$Prof \sqsubseteq \neg Student$	$Student(alice)$
$\exists mentors \sqsubseteq Prof$	$mentors(alice, bob)$
$(funct\ mentors^-)$	$mentors(andreia, bob)$

The ontology is made unsatisfiable by two culprits in the ABox:

- alice (via NI)

$$Q_1() \leftarrow \exists x(Prof(x) \wedge Student(x)) \vee \exists x, y mentors(x, y) \wedge Student(x))$$

- bob for the functional axiom

$$Q_2() \leftarrow \exists x, y, z(mentors^-(x, y) \wedge mentors^-(x, z) \wedge y \neq z)$$

FOL Rewritability of Satisfiability

Example

TBox	*ABox*
$Prof \sqsubseteq \neg Student$	$Student(alice)$
$\exists mentors \sqsubseteq Prof$	$mentors(alice, bob)$
$(funct\ mentors^{-})$	$mentors(andreia, bob)$

The ontology is made unsatisfiable by two culprits in the ABox:

- alice (via NI)

$$Q_1() \leftarrow \exists x(Prof(x) \wedge Student(x)) \vee \exists x, y mentors(x, y) \wedge Student(x))$$

- bob for the functional axiom

$$Q_2() \leftarrow \exists x, y, z(mentors^{-}(x, y) \wedge mentors^{-}(x, z) \wedge y \neq z)$$

FOL Rewritability of Satisfiability

Example

TBox	*ABox*
$Prof \sqsubseteq \neg Student$	$Student(alice)$
$\exists mentors \sqsubseteq Prof$	$mentors(alice, bob)$
$(funct\ mentors^-)$	$mentors(andreia, bob)$

The ontology is made unsatisfiable by two culprits in the ABox:

- alice (via NI)

$$Q_1() \leftarrow \exists x(Prof(x) \wedge Student(x)) \vee \exists x, y mentors(x, y) \wedge Student(x))$$

- bob for the functional axiom

$$Q_2() \leftarrow \exists x, y, z(mentors^-(x, y) \wedge mentors^-(x, z) \wedge y \neq z)$$

FOL Rewritability of Satisfiability

Example

TBox	*ABox*
$Prof \sqsubseteq \neg Student$	$Student(alice)$
$\exists mentors \sqsubseteq Prof$	$mentors(alice, bob)$
$(funct\ mentors^-)$	$mentors(andreia, bob)$

The ontology is made unsatisfiable by two culprits in the ABox:

- alice (via NI)

$$Q_1() \leftarrow \exists x(Prof(x) \wedge Student(x)) \vee \exists x, y mentors(x, y) \wedge Student(x))$$

- bob for the functional axiom

$$Q_2() \leftarrow \exists x, y, z(mentors^-(x, y) \wedge mentors^-(x, z) \wedge y \neq z)$$

Checking Inconsistency for NIs

- Every NI is separately transformed to a CQ asking for a counterexample object, e.g.,

$A \sqsubseteq \neg B$	becomes	$Q() \leftarrow \exists x.A \wedge B$
$\exists P \sqsubseteq \neg B$	becomes	$Q() \leftarrow \exists y, x.P(x, y) \wedge B(x)$

- Resulting CQs are rewritten separately with PerfectRew w.r.t. PIs in the TBox
  - Intuition closure: $A \sqsubseteq B$ and $B \sqsubseteq \neg C$ entails $A \sqsubseteq \neg C$
  - Intuition separability: No two NIs can interact.
- $Q_N :=$ union of these CQs

- For functionalities, it is enough to consider these alone

$(\mathit{funct}\ P)$ becomes $Q() \leftarrow \exists x, y, z.P(x, y) \wedge P(x, z) \wedge y \neq z$

- $Q_F :=$ union of these CQs
  - Intuition: No interaction of PI or NI with functionalities

Checking Inconsistency for NIs

- Every NI is separately transformed to a CQ asking for a counterexample object, e.g.,

$A \sqsubseteq \neg B$	becomes	$Q() \leftarrow \exists x.A \wedge B$
$\exists P \sqsubseteq \neg B$	becomes	$Q() \leftarrow \exists y, x.P(x, y) \wedge B(x)$

- Resulting CQs are rewritten separately with PerfectRew w.r.t. PIs in the TBox
  - Intuition closure: $A \sqsubseteq B$ and $B \sqsubseteq \neg C$ entails $A \sqsubseteq \neg C$
  - Intuition separability: No two NIs can interact.
- $Q_N :=$ union of these CQs

- For functionalities, it is enough to consider these alone

$(funct\ P)$ becomes $Q() \leftarrow \exists x, y, z.P(x, y) \wedge P(x, z) \wedge y \neq z$

- $Q_F :=$ union of these CQs
  - Intuition: No interaction of PI or NI with functionalities

Checking Inconsistency for NIs

- Every NI is separately transformed to a CQ asking for a counterexample object, e.g.,

$A \sqsubseteq \neg B$	becomes	$Q() \leftarrow \exists x.A \wedge B$
$\exists P \sqsubseteq \neg B$	becomes	$Q() \leftarrow \exists y, x.P(x,y) \wedge B(x)$

- Resulting CQs are rewritten separately with PerfectRew w.r.t. PIs in the TBox
  - Intuition closure: $A \sqsubseteq B$ and $B \sqsubseteq \neg C$ entails $A \sqsubseteq \neg C$
  - Intuition separability: No two NIs can interact.
- $Q_N :=$ union of these CQs

- For functionalities, it is enough to consider these alone

$(funct\ P)$ becomes $Q() \leftarrow \exists x, y, z.P(x,y) \wedge P(x,z) \wedge y \neq z$

- $Q_F :=$ union of these CQs
  - Intuition: No interaction of PI or NI with functionalities

Checking Inconsistency for NIs

- Every NI is separately transformed to a CQ asking for a counterexample object, e.g.,

$$A \sqsubseteq \neg B \quad \text{becomes} \quad Q() \leftarrow \exists x. A \wedge B$$
$$\exists P \sqsubseteq \neg B \quad \text{becomes} \quad Q() \leftarrow \exists y, x. P(x, y) \wedge B(x)$$

- Resulting CQs are rewritten separately with PerfectRew w.r.t. PIs in the TBox
  - Intuition closure: $A \sqsubseteq B$ and $B \sqsubseteq \neg C$ entails $A \sqsubseteq \neg C$
  - Intuition separability: No two NIs can interact.
- $Q_N :=$ union of these CQs

- For functionalities, it is enough to consider these alone

$$(\mathit{funct}\ P) \quad \text{becomes} \quad Q() \leftarrow \exists x, y, z. P(x, y) \wedge P(x, z) \wedge y \neq z$$

- $Q_F :=$ union of these CQs
  - Intuition: No interaction of PI or NI with functionalities

Rewritability

Theorem

Let $\mathcal{O} = (\mathcal{T}, \mathcal{A})$ be a DL-Lite$_{\mathcal{A}}$ ontology. Then:

$\mathcal{O}$ is satisfiable iff $Q_N \vee Q_F$ (which is a UCQ$^{\neq}$ and hence FOL query) is false.

- The separability has consequences for identifying culprits of inconsistency
  - At most two ABox axioms may be responsible for an inconsistency
  - This is relevant for ontology repair, version, change etc. (see next lectures)

Constructs Leading to Non-rewritability in DL-Lite

- DL-Lite$_{\mathcal{A}}$ is a maximal DL w.r.t. the allowed logical constructors under the FOL constraints
- Useful constructions such as qualified existentials, disjunction, non-restricted use of functional roles lead to non FOL-rewritability
- This can be proved using complexity theory and FOL (un-)definability arguments

Qualified existentials on Lhs

- Reachability in directed gaphs is known to be NLOGSPACE-complete
- X is FOL expressible $\leftrightarrow X \in AC^0 \subsetneq NLOGSPACE$
- Reachability reducible to QA

Reduction

$$
\begin{array}{rcl}
\text{Given:} & & \mathfrak{G}, \text{start } s, \text{ end } t \\
\mathcal{A}_{\mathfrak{G},t} & = & \{edge(v_1, v_2) \mid (v_1, v_2)\} \cup \{pathToTarget(t)\} \\
\mathcal{T} & = & \{\exists edge.PathToTarget \sqsubseteq PathToTarget\} \\
CQ & = & q() \leftarrow PathToTarget(s)
\end{array}
$$

- Fact: $\mathcal{T} \cup \mathcal{A}_{\mathfrak{G},t} \models q$ iff there is a path from s to t in $\mathfrak{G}$
- Fact: $\mathcal{T}, q$ do not depend on $\mathfrak{G}, t$
- Problem $\mathcal{T} \cup \mathcal{A}_{\mathfrak{G},t} \models q$ is NLOGSPACE hard

Limits of DL-Lite

- DL-Lite$_{\mathcal{A}}$ is not the maximal fragment of FOL allowing for rewritability

- Datalog$^{\pm}$ = Datalog with existentials in head = set of tuple generating (TGDs) rules (and EGDs)
  - Datalog$_0^{\pm}$ = "Linear fragment" of Datalog$^{\pm}$ containing rules whose body consists of one atom
  - Fact: Datalog$_0^{\pm}$ is strictly more expressive than DL-Lite.

Example

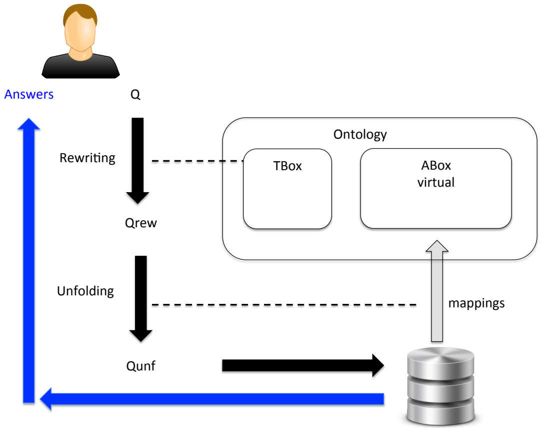
The rule

$$\forall x.manager(x) \rightarrow manages(x, x)$$

is in Datalog$_0^{\pm}$ but in no member of the DL-Lite family.

Unfolding

Connecting to the Real World: Mappings and Unfolding



Reminder: Mappings

- Mappings have an important role for OBDA

Schema of Mappings $\mathcal{M}$

m_1:	ontology template$_1$	$\longleftarrow$	data source template$_1$
m_2:	ontology template$_2$	$\longleftarrow$	data source template$_2$
	...		

- Lift data to the ontology level
  - Data level: (nearly) closed world
  - Ontology level: open world
- Mappings, described as rules, provide declarative means of implementing the lifting
  - User friendliness: users may built mappings on their own
  - Neat semantics: the semantics can be clearly specified and is not hidden in algorithms (as in direct mappings)
  - Modularity: mappings can be easly extended, combined etc.
  - Reuse of tools: Can be managed by (adapted) rule engines

The Burden of Mappings

- The data-to-ontology lift faces impedance mismatch
  - data values in the data vs.
  - abstract objects in the ontology world
  - Solved by Skolem terms $\vec{f}(\vec{x})$ below

Schema of Mappings

$$m : \psi(\vec{f}(\vec{x})) \longleftarrow \boxed{Q(\vec{x}, \vec{y})}$$

- $\psi(\vec{f}(\vec{x}))$: Query for generating ABox axioms
- $Q(\vec{x}, \vec{y})$: Query over the backend sources
- Function $\vec{f}$ translates backend instantiations of $\vec{x}$ to constants

- Mappings M over backend sources generates ABox $\mathcal{A}(M, DB)$.
- Use of mappings
  - as ETL (extract, transform, load) means: materialize ABox
  - as logical view means: ABox kept virtual (classical OBDA)

Example Scenario: Measurements

- Example schema for measurement and event data in DB

```
SENSOR(SID, CID, Sname, TID, description)
SENSORTYPE(TID, Tname)
COMPONENT(CID, superCID, AID, Cname)
ASSEMBLY(AID, AName, ALocation)
MEASUREMENT(MID, MtimeStamp, SID, Mval)
MESSAGE(MesID, MesTimeStamp, MesAssemblyID,  catID, MesEventText)
CATEGORY(catID, catName)
```

- For mapping

m: $Sens(f(SID)) \wedge name(f(SID), y) \longleftarrow$

```
SELECT SID,  Sname as y FROM SENSOR
```

- the row data in SENSOR table

SENSOR
(123, comp45, TC255, TempSens, 'A temperature sensor')

- generates facts

$Sens(f(123)), name(f(123), TempSens) \in \mathcal{A}(m, DB)$

Example Scenario: Measurements

- Example schema for measurement and event data in DB

```
SENSOR(SID, CID, Sname, TID, description)
SENSORTYPE(TID, Tname)
COMPONENT(CID, superCID, AID, Cname)
ASSEMBLY(AID, AName, ALocation)
MEASUREMENT(MID, MtimeStamp, SID, Mval)
MESSAGE(MesID, MesTimeStamp, MesAssemblyID,  catID, MesEventText)
CATEGORY(catID, catName)
```

- For mapping

m: $Sens(f(SID)) \wedge name(f(SID), y) \longleftarrow$

```
SELECT SID,  Sname as y FROM SENSOR
```

- the row data in SENSOR table

SENSOR
(123, comp45, TC255, TempSens, ‘A temperature sensor’)

- generates facts

$Sens(f(123)), name(f(123), TempSens) \in \mathcal{A}(m, DB)$

R2RML

- Very expressive mapping language couched in the RDF terminology
- Read only (not to allowed to write the RDFs view generated by the mappings)
- W3C standard (since 2012), http://www.w3.org/TR/r2rml/
- Defined for logical tables (= SQL table or SQL view or R2RML view)
  $\Longrightarrow$ they can be composed to chains of mappings
- Has means to model foreign keys (referencing object map)

Example (R2RML for Sensor Scenario)

```
@prefix rdf : <http ://www.w3.org/1999/02/22?rdf?syntax?ns#> .
@prefix rr : <http ://www.w3. org/ns/r2rml#> .
@prefix ex : <http ://www. example . org/> .

ex : SensorMap
    a rr:TriplesMap ;
    rr: logicalTable [ rr : tableName ‘‘Senso’’ ] ;
    rr : subjectMap [
        rr:template ‘‘http://www.sensorworld.org/SID’’ ;
        rr:class ex:Sensor
    ];
    rr: predicateObjectMap [
        rr:predicate ex:hasName;
        rr:objectMap [column ‘‘name’’]
    ] .
```

OBDA Semantics with Mappings

- Semantics canonically specified by using the generated ABox $\mathcal{A}(DB, \mathcal{M})$
- Ontology Based Data Access System (OBDAS)

$$\mathcal{OS} = (\underbrace{\mathcal{T}}_{\text{TBox}}, \overbrace{\mathcal{M}}^{\text{mappings}}, \underbrace{DB}_{\text{relational data base}})$$

Definition

An interpretation $\mathcal{I}$ satisfies an OBDAS $\mathcal{OS} = (\mathcal{T}, \mathcal{M}, DB)$, for short: $\mathcal{I} \models \mathcal{OS}$, iff $\mathcal{I} \models (\mathcal{T}, \mathcal{A}(DB, \mathcal{M}))$

An OBDAS is satisfiable iff it has a satisfying interpretation.

Unfolding

- Unfolding is the second but not to be underestimated step in classical OBDA QA
- Applies mappings in the inverse direction to produce query Q_{unf} over data sources which then becomes evaluated

Unfolding steps

1. Split mappings
   $atom_1 \wedge \cdots \wedge atom_n \longleftarrow Q$ becomes
   $atom_1 \longleftarrow Q, \ldots, atom_n \longleftarrow Q$
2. Introduce auxiliary predicates (views for SQL) for rhs queries
3. In Q_{rew} unfold the atoms (with unification) into a UCQ Q_{aux} using purely auxiliary predicates
4. Translate Q_{aux} into SQL
   - logical conjunction of atoms realized by a join
   - disjunction of queries realized by SQL UNION

Example (Unfolding for Measurement Scenario)

- DB with schema

```
SENSOR(SID, CID, Sname, TID, description)
MEASUREMENT1(MID, MtimeStamp, SID, Mval)
MEASUREMENT2(MID, MtimeStamp, SID, Mval) ...
```

- Mappings

m1: $Sens(f(SID)) \wedge name(f(SID), y) \longleftarrow$

```
SELECT SID, Sname as y FROM SENSOR
```

m2: $hasVal(f(SID), Mval) \longleftarrow$

```
SELECT SID, Mval FROM Measurement1
```

m3: $hasVal(f(SID), Mval) \longleftarrow$

```
SELECT SID, Mval FROM Measurement2
```

m4: $criticalValue(Mval) \longleftarrow$

```
SELECT Mval FROM MEASUREMENT1
WHERE Mval > 300
```

- Query

$Q(x) \longleftarrow Sens(x) \wedge hasVal(x, y) \wedge Critical(y)$

Example

Unfolding for Measurement Scenario

- Split mappings

  m1.1: $Sens(f(SID)) \longleftarrow$

  ```
  SELECT SID FROM SENSOR
  ```

  m1.2: $name(f(SID), y) \longleftarrow$

  ```
  SELECT SID, Sname as y FROM SENSOR
  ```

  m2: $hasVal(f(SID), Mval) \longleftarrow$

  ```
  SELECT SID, Mval FROM Measurement1
  ```

  m3: $hasVal(f(SID), Mval) \longleftarrow$

  ```
  SELECT SID, Mval FROM Measurement2
  ```

  m4: $criticalValue(Mval) \longleftarrow$

  ```
  SELECT Mval FROM MEASUREMENT1
  WHERE Mval > 300
  ```

- Query

$$Q(x) \quad \longleftarrow \quad Sens(x) \wedge hasVal(x, y) \wedge Critical(y)$$

Example

Unfolding for Measurement Scenario

- Split mappings

m1.1: $Sens(f(SID)) \longleftarrow$

`SELECT SID FROM SENSOR` =:Aux1(SID)

m1.2: $name(f(SID), y) \longleftarrow$

`SELECT SID, Sname as y FROM SENSOR` =:Aux2(SID,y)

m2: $hasVal(f(SID), Mval) \longleftarrow$

`SELECT SID, Mval FROM Measurement1` =:Aux3(SID,Mval)

m3: $hasVal(f(SID), Mval) \longleftarrow$

`SELECT SID, Mval FROM Measurement2` =:Aux4(SID,Mval)

m4: $criticalValue(Mval) \longleftarrow$

```
SELECT Mval FROM MEASUREMENT1
WHERE Mval > 300
```

=:Aux5(Mval)

- Query

$$Q(x) \quad \longleftarrow \quad Sens(x) \wedge hasVal(x, y) \wedge Critical(y)$$

Example (Unfolding for Measurement Scenario)

- Split mappings

m1.1: $Sens(f(SID)) \longleftarrow$

`SELECT SID FROM SENSOR` :=Aux(SID)

m1.2: $name(f(SID), y) \longleftarrow$

`SELECT SID, Sname as y FROM SENSOR` =:Aux2(SID,y)

m2: $hasVal(f(SID), Mval) \longleftarrow$

`SELECT SID, Mval FROM Measurement1` =:Aux3(SID,Mval)

m3: $hasVal(f(SID), Mval) \longleftarrow$

`SELECT SID, Mval FROM Measurement2` =:Aux4(SID,Mval)

m4: $criticalValue(Mval) \longleftarrow$

```
SELECT Mval FROM MEASUREMENT1
WHERE Mval > 300
```

=:Aux5(Mval)

- Query

$$Q(x) \quad \longleftarrow \quad Sens(x) \wedge hasVal(x, y) \wedge Critical(y)$$

- Query Q_{Aux} with Aux-views

$$Q_{Aux} \quad \longleftarrow \quad Aux1(SID), Aux3(SID, Mval), Aux5(Mval)$$
$$Q_{Aux} \quad \longleftarrow \quad Aux1(SID), Aux4(SID, Mval), Aux5(Mval)$$

Example

Unfolding for Measurement Scenario

```
SELECT 'Qunfold' || aux_1.SID || ')' FROM
    (SELECT  SID FROM SENSOR) as aux_1,
    ( SELECT  SID,  Mval  FROM Measurement1) as aux_3,
    (SELECT  Mval   FROM MEASUREMENT1 WHERE Mval > 300) as aux_5
    WHERE aux_1.SID = aux_3.SID AND aux_3.Mval = aux_5.Mval
UNION
SELECT 'Qunfold' || aux_1.SID || ')' FROM
    (SELECT  SID FROM SENSOR) as aux_1,
    ( SELECT  SID,  Mval  FROM Measurement2) as aux_4,
    (SELECT  Mval   FROM MEASUREMENT1 WHERE Mval > 300) as aux_5
    WHERE aux_1.SID = aux_4.SID AND aux_4.Mval = aux_5.Mval
```

- There are different forms of unfolding

Research on OBDA Mappings

- Recent research on classical OBDA reflects the insight of mappings' importance

- Adequateness conditions for mappings
  - consistency/coherency
  - redundancy

- Management of mappings
  - Repairing mappings (based on consistency notion)
  - Approximating ontologies and mappings

Lit: D. Lembo et al. Mapping analysis in ontology-based data access: Algorithms and complexity. In: ISWC 2015, volume 9366 of LNCS, pages 217–234, 2015.

Need for Opimizations

- UCQ-Rewritings may be exponentially larger than the original query
- Have to deal with this problem in practical systems
- Use different rewriting to ensure conciseness
- Use additional knowledge on the data: integrity constraints, (H)-completeness

- Have a look at OBDA framework ontop (`http://ontop.inf.unibz.it/`)
  - Open source
  - available as Protege plugin
  - implementing many optimizations

Solutions to Exercise 7 (14 points)

Exercise 7.1 (4 points)

1. Give a DL formalization of the following concept description "Father who has only children that are doctors or managers "
2. Give a DL formalization of the following assertion: "A busy female lecturer is a person who teaches at least three courses"

Solution

1. $\mathit{Father} \sqcap \forall \mathit{hasChild}.(\mathit{Doctor} \sqcup \mathit{Manager})$
2. $\mathit{BusyFemaleLecture} \equiv \mathit{Person} \sqcap (\geq 3\ \mathit{teaches}.\mathit{Courses})$

Exercise 7.2 (7 points)

Consider the following TBox $\mathcal{T}$

$$\begin{aligned} A &\sqsubseteq B \\ B &\sqsubseteq C \\ C &\sqsubseteq \exists R.D \\ D &\sqsubseteq \neg A \end{aligned}$$

1. In which DL is $\mathcal{T}$?
2. Is $\mathcal{T}$ satisfiable? If so, give a model, else argue why it is not satisfiable.
3. Is the concept D satisfiable w.r.t. $\mathcal{T}$, i.e., is there a model of $\mathcal{T}$ in which D is not interpreted by the empty set? If yes, give such a model else argue why it is not satisfiable
4. Is $D \sqcap A$ satisfiable w.r.t. $\mathcal{T}$? If so, give a model, else argue why it is not satisfiable.

Solution to Exercise 7.3 (7 points)

Consider the following TBox $\mathcal{T}$

$$\begin{array}{lcl} A & \sqsubseteq & B \\ B & \sqsubseteq & C \\ C & \sqsubseteq & \exists R.D \\ D & \sqsubseteq & \neg A \end{array}$$

1. DL-Lite
2. $\mathcal{T}$ is satisfiable. Take $\mathcal{I}$ with $A^{\mathcal{I}} = B^{\mathcal{I}} = C^{\mathcal{I}} = D^{\mathcal{I}} = R^{\mathcal{I}} = \emptyset$.
3. Concept D is satisfiable w.r.t. $\mathcal{T}$. Take $\mathcal{I}$ with $\Delta^{\mathcal{I}} = \{d\}$ and $D^{\mathcal{I}} = \{d\}$ and $A^{\mathcal{I}} = C^{\mathcal{I}} = B^{\mathcal{I}} = R^{\mathcal{I}}$
4. $D \sqcap A$ is not satisfiable w.r.t. $\mathcal{T}$. In any model where $D \sqcap A$ is satisfiable by say $d \in (D \sqcap A)^{\mathcal{I}}$, it must be $d \in (D)^{\mathcal{I}}$ and $d \in (A)^{\mathcal{I}}$, contradicting last axiom.

Exercise 7.3 (3 points)

Show that subsumption can be reduced to satisfiability tests (allowing the introduction of new constants). More concretely:

$C \sqsubseteq D$ w.r.t. $\mathcal{O}$ iff $(\sigma \cup \{b\}, \mathcal{T}, \mathcal{A} \cup \{C(b), \neg D(b)\})$ is not satisfiable (where b is a fresh constant).

Solution Assume $C \sqsubseteq D$ w.r.t $\mathcal{O}$.
$(\sigma \cup \{b\}, \mathcal{T}, \mathcal{A} \cup \{C(b), \neg D(b)\})$ were satisfiable by $\sigma \cup \{b\}$-interpretation $\mathcal{I}$, say, then could consider σ-interpretation $\mathcal{I}'$ which is the same as $\mathcal{I}$ for all denotations of elements in σ. Then $\mathcal{I}' \models \mathcal{O}$, so $\mathcal{I}' \models C \sqsubseteq D$. In particular as $\mathcal{I}(b) = \underline{b} \in C^{\mathcal{I}} = C^{\mathcal{I}'}$ we must also have $\underline{b} \in D^{\mathcal{I}'} = D^{\mathcal{I}}$. But $\underline{b} = \mathcal{I}(b) \notin D^{\mathcal{I}}$.
Now assume not $C \sqsubseteq D$ w.r.t $\mathcal{O}$. So there is $\mathcal{I} \models \mathcal{O}$ such that $\mathcal{I} \not\models C \sqsubseteq D$. So there is $\underline{b} \in \Delta^{\mathcal{I}}$ such that $\underline{b} \in C^{\mathcal{I}}$ and $\underline{b} \notin D^{\mathcal{I}}$.
Now can define new $\mathcal{I}'$ over $\sigma \cup \{b\}$ with $\mathcal{I}'(b) = \underline{b}$.

Exercise 8 (20 Bonus Points)

Exercise 8.1 (2 Bonus Points)

Prove that there are DL-Lite$_{\mathcal{F}}$ ontologies having only infinite models (using, e.g., the example mentioned in the lecture)

Exercise 8.2 (4 Bonus Points)

The anonymization function in the PerfRew algorithm is allowed to be applied only to un-bound variables. Unbound variables are those that occurs at most once in the body and that are not distinguished, i.e., that are not answer variables. Give an example showing that it makes sense to exclude distinguished variables from anonymization.

Exercise 8.3 (4 Bonus Points)

Explain the notion of reification, and show (with an example) why it is needed for (classical) OBDA.

Exercise 8.4 (4 Bonus Points)

Many relevant DL reasoning services can be reduced to ontology satisfiability in DL-Lite. Show that subsumption w.r.t. a DL-Lite TBox can be reduced to (un)satisfiability test of a DL-Lite ontology!

Hint: Use the general fact of entailment that $\psi \models \phi$ iff $\psi \wedge \neg\phi$ is unsatisfiable (or use Exercise 7.3) Then think of how the latter can be formulated in a DL-Lite ontology (introducing perhaps new symbols).

Exercise 8.5 (6 Bonus Points)

Inform yourself about modal logic (syntax and semantics) and give examples of how to translate $\mathcal{ALC}$ concepts into modal logic formulae.